

**ESTUDO DE SISTEMAS DE POLLING COM TEMPOS DE
SERVIÇO LIMITADOS**

Paulo Alcântara Saraiva Leão

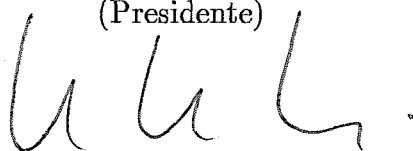
TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



Prof. Edmundo A. de Souza e Silva, Ph.D.

(Presidente)



Prof. Paulo H. Aguiar Rodrigues, Ph.D.



Prof. Valmir Carneiro Barbosa, Ph.D.



Prof. Aloísio de Castro P. Pedrosa, Ph.D.

LEÃO, PAULO ALCÂNTARA SARAIVA

Estudo de Sistemas de Polling com Tempos de Serviço Limitados [Rio de Janeiro] 1993

XII, 97 p., 29.7 cm, (COPPE/UFRJ, M. Sc., Engenharia de Sistemas e Computação, 1993)

TESE - Universidade Federal do Rio de Janeiro, COPPE

1. Sistemas de Polling.

2. Avaliação de Desempenho.

I. COPPE/UFRJ II. Título (série).

À Ana Lúcia.

AGRADECIMENTOS:

- Ao professor Edmundo de Souza e Silva, não pela praxe de agradecer, mas pelo reconhecimento da sua orientação, ajuda e dedicação na elaboração desta tese.
- Aos meus tios José e Norma por toda a ajuda, carinho e atenção que me deram.
- Aos meus pais, Pedro Henrique e Luiza, e à minha avó Eunice, por toda a força, amor e incentivo dados.
- Aos amigos antigos e novos, mas em especial a Evande, Erivaldo, José Maria, Lassance, Marcelo *Lorin*, Walmir *Big Mac*, Phillipe, Júlio, Neudson, Adriana, Ricardo Jorge, Lucídio, Odinaldo, Denise, Lilian, Sandra Isabel, Sônia, Natália, Sandra Regina, Porfírio, Plácido, Rochinha, Marcelo Dib, Nalvo, Sílvio, Edson, Nahri, Fernando *Roger*, Marcelo e João Carlos, pela amizade e pelos bons momentos juntos.
- Aos funcionários e professores da COPPE pelo apoio recebido.
- Ao CNPq e CAPES pela ajuda financeira.
- Ao Serviço de Processamento de Dados do Estado do Ceará (SEPROCE), e em especial a Francisco Augusto Andrade Maia, por todo o apoio concedido.

Resumo da tese apresentada à COPPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

ESTUDO DE SISTEMAS DE POLLING COM TEMPOS DE SERVIÇO
LIMITADOS

Paulo Alcântara Saraiva Leão
Janeiro de 1993

Orientador : Edmundo A. de Souza e Silva
Programa : Engenharia de Sistemas e Computação

A tendência atual em sistemas de comunicação é a integração, em uma rede local, de aplicações em tempo-real com aplicações que não sejam em tempo-real, cada uma com seus próprios requisitos de serviço. A incorporação do tráfego em tempo-real na rede requer que sejam projetados e implementados protocolos que atendam conjuntamente aos diferentes requisitos de serviço dos vários tipos de tráfego envolvidos. Nesta tese, estudamos um método de modelagem e análise para a avaliação de desempenho de sistemas de *polling* com tempos de serviço limitados. Esse tipo de sistema se presta para modelar alguns sistemas reais de transmissão, em um mesmo ambiente, de vários tipos de tráfego, onde são atendidos conjuntamente os seus requisitos de serviço. O trabalho realizado na tese compreendeu estudar a teoria do método, implementar, em computador, as suas equações, e fazer um estudo de casos para analisar alguns sistemas, quanto a certas medidas de desempenho de interesse.

Abstract of thesis presented to COPPE as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

A SURVEY ON POLLING SYSTEMS WITH LIMITED SERVICE TIMES

Paulo Alcântara Saraiva Leão

January, 1993

Thesis Supervisor : Edmundo A. de Souza e Silva
Department : System Engineering and Computing

Today, the trend in communication systems is the integration, in a local network, of real-time applications with non real-time applications, each one with its own service requirements. The inclusion of the real-time traffic in the network requires that protocols be designed and implemented attending jointly to the different service requirements of the many types of traffic involved. In this thesis, we study a analysis and modeling method designed to evaluate the performance of *polling* systems with limited service times. This kind of system is suited to model some real transmission systems wich deal, in the same environment, with many types of traffic, where the service requirements are jointly taken into account. The work accomplished in the thesis encompassed the study of the method's theory, the implementation in a computer of its equations, and a study of cases to analize some systems, regarding certain performance measures.

Índice

1	Introdução	1
1.1	O Multiplexador (T1,T2)	3
1.2	O Protocolo de Acesso ao Meio do Padrão IEEE 802.4 <i>Token Bus</i>	5
1.3	O Protocolo FDDI	7
1.4	Trabalhos Relacionados	8
2	O Método Proposto	15
2.1	Descrição do Modelo	15
2.2	Alguns Conceitos Básicos	17
2.2.1	Randomização	17
2.2.2	Cadeia de Markov com Recompensas	18
2.3	O Método de Modelagem e Análise	19
2.3.1	A Cadeia de Markov Embutida	19
2.3.2	Algumas Medidas de Desempenho	29
3	Aspectos Computacionais	38
3.1	Prevenindo Situações de <i>Underflow/Overflow</i>	39
3.2	Determinando $S^{(j)}$	41
3.3	Determinando $D^{(j)}$ para Modelos com <i>Buffer</i> Ilimitado	42
3.4	Determinando $D^{(j)}$ para Modelos com <i>Buffer</i> Limitado	44

3.5	Detalhes da Ferramenta Implementada	50
4	Resultados Numéricos	57
4.1	Influência do Tamanho do <i>Buffer</i>	57
4.2	Influência dos Intervalos de <i>Switch-over</i>	60
4.3	Influência das Quotas de Tempo de Serviço (T_1, T_2)	61
4.4	Influência da Utilização de uma Fila	78
4.5	Comparando com o Esquema TDM	84
4.6	Comparando com uma Fila M/M/1/K	85
4.7	Comparando com o Esquema HOL (<i>Head-Of-the-Line</i>)	85
4.8	Analisando a Convergência	90
5	Conclusão	92
A		94
	Referências Bibliográficas	96

Lista de Figuras

2.1	Ciclos de <i>Polling</i> e a Matriz C	20
2.2	Mini-ciclos de Serviço e Intervalos de <i>Switch-over</i>	21
2.3	Matrizes D e S	22
2.4	Cadeia de Markov para Serviço Exaustivo	24
2.5	Cadeia de Markov Randomizada para Serviço Exaustivo	25
2.6	Matrizes D, S, A e B	30
3.1	Cadeia de Markov Modificada	43
3.2	Recursão de Φ	46
3.3	Recursão de Δ e Ξ	48
3.4	Sub-estrutura para o Cálculo de $d_{s,s'}^{(j)}$	52
3.5	Sub-estrutura Após a Reorganização	53
4.1	Retardo X B	58
4.2	Prob. Bloqueio X B (Sistema Simétrico)	59
4.3	Retardo X B (Sistema Simétrico)	59
4.4	Retardo X Utilização da Mesma Fila	60
4.5	Influência dos Intervalos de <i>Switch-over</i>	61
4.6	Retardo X (T_1, T_2) (Sistema Simétrico)	62
4.7	Retardo Fila 1 X (T_1, T_2) (Sistema não Simétrico)	63
4.8	Retardo Fila 2 X (T_1, T_2) (Sistema não Simétrico)	63

4.9	Prob. Bloqueio Fila 1 X (T_1, T_2) (Sistema não Simétrico)	64
4.10	Prob. Bloqueio Fila 2 X (T_1, T_2) (Sistema não Simétrico)	64
4.11	Retardo Fila 2 X (T_1, T_2) (<i>Switch-Over</i> constante)	65
4.12	Retardos X (T_1, T_2)	66
4.13	<i>RC</i> X (T_1, T_2)	67
4.14	<i>PBC</i> X (T_1, T_2)	67
4.15	Retardo Fila 1 X (T_1, T_2) ($T_2 = 2T_1$)	68
4.16	Retardo Fila 2 X (T_1, T_2) ($T_2 = 2T_1$)	69
4.17	Retardo Fila 2 X (T_1, T_2) (<i>Switch-Over</i> constante)	69
4.18	Prob. Bloqueio Fila 1 X (T_1, T_2) ($T_2 = 2T_1$)	70
4.19	Prob. Bloqueio Fila 2 X (T_1, T_2) ($T_2 = 2T_1$)	71
4.20	Retardo Fila 1 X (T_1, T_2) ($\lambda_1 = \lambda_2$ e $\mu_1 = \mu_2$)	72
4.21	Retardo Fila 2 X (T_1, T_2) ($\lambda_1 = \lambda_2$ e $\mu_1 = \mu_2$)	72
4.22	Retardo Fila 1 X (T_1, T_2) ($T_1 = T_2$ e $\lambda_2 = 2\lambda_1$)	73
4.23	Retardo Fila 2 X (T_1, T_2) ($T_1 = T_2$ e $\lambda_2 = 2\lambda_1$)	73
4.24	Prob. Bloqueio Fila 2 X (T_1, T_2) ($T_1 = T_2$ e $\lambda_2 = 2\lambda_1$)	74
4.25	Retardo Fila 1 X T_2	74
4.26	Retardo Fila 2 X ρ_1	76
4.27	Prob. Bloqueio Fila 1 X ρ_2	76
4.28	Retardo Fila 1 X ρ_1	77
4.29	Retardo Fila 1 X ρ_2	77
4.30	Prob. Bloqueio Fila 1 X ρ_1	79
4.31	Retardo Fila 1 X ρ_2 (Influência de B)	79
4.32	Retardo X ρ_2	80
4.33	Prob. Bloqueio Fila 2 X ρ_1	81
4.34	Prob. Bloqueio Fila 2 X ρ_1 ($(T_1, T_2) = (8, 2)$)	81

4.35 Prob. Bloqueio Fila 2 X ρ_1 ($(T_1, T_2) = (3, 2)$)	82
4.36 (T_1, T_2) X TDM (Retardo)	83
4.37 (T_1, T_2) X M/M/1/K (Retardo)	84
4.38 Retardo X ρ ($(T_1, T_2, T_3) = (2, 4, 6)$)	85
4.39 (T_1, T_2, T_3) X HOL (Retardo Tráfego Prior. Baixa)	86
4.40 (T_1, T_2, T_3) X HOL (Retardo Tráfego Prior. Média)	87
4.41 (T_1, T_2, T_3) X HOL (Retardo Tráfego Prior. Alta)	88
4.42 (T_1, T_2, T_3) X HOL (Retardo)	88
4.43 Retardo X ρ ($(T_1, T_2, T_3) = (5, 5.5, 6)$)	89
4.44 Retardo X ρ ($(T_1, T_2, T_3) = (1, 5, 9)$)	89
4.45 Velocidade de Convergência X B	90
4.46 Velocidade de Convergência X (T_1, T_2)	91

Lista de Tabelas

3.1	Sistema com $M = 3$ e $B_i = 2$	47
3.2	Sistema com $M = 3$ e $B_i = 4$	48

Capítulo 1

Introdução

A atual tendência na área de sistemas de comunicação é a integração, em uma rede local, de aplicações em tempo-real (e. g. voz, controle de processos, sinalização de alarme) com aplicações que não sejam em tempo-real, como por exemplo tráfego de dados computacionais convencionais. Ao contrário da transmissão de dados computacionais, a transmissão de informações em tempo-real introduz requisitos de retardo, ou tempo de resposta, bastante rigorosos.

Os protocolos para redes locais convencionais não conseguem atender eficientemente tais requisitos por não terem sido projetados e desenvolvidos para tal fim. As redes locais foram inicialmente concebidas para suportar o tráfego de dados computacionais, que não têm requisitos de retardo tão rígidos. Eficientes protocolos para redes locais convencionais, como o *Ethernet* (CSMA/CD) e o *Token Ring*, embora permitindo a transmissão de diferentes tipos de dados, não são os mais adequados para uma solução integrada. Isto acontece, pois estes protocolos empregam técnicas simples de compartilhamento do meio de transmissão, semelhantes à multiplexação estatística. Na multiplexação estatística, ou multiplexação no tempo assíncrona, os pacotes de informação dos diversos tráfegos são transmitidos (servidos) sob uma política FCFS (*First Come First Served*), ou seja, os pacotes de um certo tráfego só têm acesso ao canal para transmissão quando todos os pacotes dos outros tráfegos, que chegaram primeiro, tiverem sido servidos. Em situações de sobrecarga, o tráfego em tempo-real poderia ser penalizado, chegando mesmo a sofrer retardos além do permitido. Portanto, a multiplexação estatística não dá ao tráfego em tempo-real a prioridade necessária, para uso do canal de transmissão, frente aos outros tráfegos, e conseqüentemente não garante seus rígidos requisitos em termos de tempo de resposta. Além disso, estes protocolos não têm um esquema de prioridades flexível entre os vários tipos de tráfego.

Com o crescente interesse pela integração de aplicações em tempo-real com aplicações que não sejam em tempo-real em redes locais, fez-se necessário o projeto e desenvolvimento de novos protocolos de comunicação. Estas novas propostas devem ser capazes de gerenciar a transmissão conjunta dos dois tipos de

tráfego de uma maneira eficiente e respeitando as suas características e exigências particulares.

A integração de serviços (e. g. dados convencionais, voz, imagem) traz várias vantagens. A redução dos custos de processamento em relação aos custos dos meios de transmissão nos últimos anos, bem como a disponibilidade de tecnologia de redes de alta velocidade a baixo custo com o advento da fibra óptica, tornaram vantajosa a utilização de uma única via de comunicação, para diferentes serviços e aplicações. Uma outra vantagem é advinda da observação de que, em muitos sistemas de comunicação de dados, existem normalmente capacidades de transmissão ociosas, que poderiam ser usadas com outros tipos de serviço. Além disso, os recentes desenvolvimentos na área de reconhecimento da voz humana e de imagens, por computador, sugerem, para um futuro próximo, um sistema altamente interativo entre homens e máquinas, para o qual, as redes integradas serão um veículo de comunicação adequado. A essas redes, que são uma evolução dos sistemas PABX (*Private Automatic Branch eXchange*) para integração de voz e dados, damos o nome de Redes Digitais com Integração de Serviços, ou em inglês, ISDN (*Integrated Service Digital Networks*). Portanto, as redes de comunicação do futuro deverão manipular uma grande variedade de tipos de tráfego de dados e aplicações, tais como: serviços de informação interativos, tráfegos de controle e alarme, aplicações de correio eletrônico, voz digitalizada, fax, transferência de arquivos, e transmissão de vídeo em banda larga, dentre outros. Uma rede integrada deverá também suportar, no mesmo ambiente, serviços de comunicação que utilizem as tecnologias de comutação de pacotes e comutação de circuito.

O objetivo desta tese foi estudar o método de modelagem e análise proposto em [1], para a análise de desempenho de sistemas de *polling* com tempos de serviço limitados. Estes sistemas de *polling* podem modelar alguns sistemas reais de transmissão, em um mesmo meio, de diferentes tipos de tráfego, como por exemplo, dados convencionais e voz digitalizada, onde são atendidos conjuntamente os diferentes requisitos de serviço dos vários tráfegos. O trabalho realizado na tese compreendeu estudar a teoria do método, desenvolver uma ferramenta visando implementar, em computador, as suas equações, e fazer um amplo estudo de casos, analisando o comportamento de alguns sistemas, quanto ao seu desempenho. A implementação das equações do método não é trivial, e foi feita paralelamente ao desenvolvimento da teoria. Inicialmente apresentaremos, em alto nível, três protocolos projetados para tratar diferentes tipos de tráfego, que podem ser parcialmente ou totalmente modelados e analisados, pelo método desenvolvido em [1], e estudado aqui. No final deste capítulo, comentaremos brevemente alguns importantes trabalhos relacionados ao tema em estudo. No capítulo 2, introduziremos o modelo e descreveremos detalhadamente o método de solução proposto. No capítulo 3, discutiremos os aspectos computacionais relacionados à resolução, em computador, das equações do método de modelagem e análise. Já no capítulo 4, analisaremos diversos sistemas de *polling*, com tempos de serviço limitados, e fazemos também estudos comparativos com outros tipos de sistemas. Finalmente, no capítulo 5, apresentamos as nossas conclusões relativas ao estudo realizado.

A seguir, apresentaremos o multiplexador (T1,T2), o protocolo de acesso ao meio do padrão IEEE 802.4 *Token Bus* e o protocolo FDDI. O protocolo adotado pelo multiplexador (T1,T2) pode ser modelado e analisado completamente pelo método desenvolvido em [1], quando dois tipos de tráfego são considerados. Os protocolos dos padrões IEEE 802.4 e FDDI também podem ser modelados pelo método, mas não será objeto do estudo feito neste trabalho.

1.1 O Multiplexador (T1,T2)

O multiplexador (T1,T2), proposto em [2], emprega um método de alocação de largura de banda, chamado esquema (T1,T2), para integrar tráfegos de voz e dados convencionais em um ambiente de transmissão de dados. Os pacotes de voz e de dados são armazenados separadamente em *buffers* (filas) e transmitidos alternadamente. Hipoteticamente, é imaginado um servidor que visita continuamente as filas. O momento no qual uma fila recebe permissão para usar o canal para transmissão é visto como a chegada do servidor àquela fila. T1 e T2 são limites máximos de tempo para transmissão de voz e dados, respectivamente, durante uma visita do servidor. Caso a fila que está sendo servida fique vazia antes que seu tempo máximo de serviço expire, o servidor passa imediatamente a servir a outra fila (outro tipo de tráfego). Além disso, caso não haja pacotes para serem transmitidos em uma fila, quando da chegada do servidor, este passa imediatamente à próxima fila.

No esquema, são garantidas larguras de banda mínimas para transmissão dos dois tipos de tráfego. As larguras de banda mínima garantidas aos tráfegos de voz e de dados são proporcionais aos seus limites de tempo T1 e T2, respectivamente. A alocação das larguras de banda é flexível pois, se uma fila estiver vazia, o outro tráfego pode utilizar a capacidade do canal para transmissão. Neste caso um dos tráfegos teve sua largura de banda aumentada dinamicamente. No esquema (T1,T2) de multiplexação de voz e dados, a alocação da largura de banda de transmissão é eficiente pois cada tipo de tráfego pode usar uma parte da largura de banda do canal que possa estar eventualmente disponível devido à inatividade do outro tipo de tráfego.

O multiplexador (T1,T2), tem na realidade três filas para armazenar os pacotes de três tipos diferentes de tráfego: sinalização, voz e dados. O servidor visita continuamente as três filas de uma maneira cíclica e em uma ordem pré-estabelecida. A fila de pacotes de sinalização é visitada primeiro, seguida das filas de voz e de dados nessa ordem. Quando não há interferência do tráfego de sinalização, que inclusive é desprezível em situações práticas, os pacotes de voz e de dados são servidos pelo esquema (T1,T2) da seguinte maneira: a fila de voz é servida até um tempo máximo de T1 unidades de tempo, ou até ela ficar vazia, o que acontecer primeiro. A seguir, a fila de dados é servida da mesma maneira, mas com um limite de tempo de serviço igual a T2 unidades de tempo. No entanto, qualquer serviço de voz ou dados pode ser interrompido pela chegada de pacotes de sinalização ao sistema, pois o tráfego de sinalização tem prioridade preemptiva sobre os outros dois

tipos de tráfego. Se pacotes de sinalização chegarem e o servidor estiver transmitindo pacotes de voz ou dados, o atendimento da fila será interrompido, após a conclusão da transmissão do pacote em serviço, e o servidor passará a servir os pacotes de sinalização. O temporizador da fila interrompida é suspenso e só é reativado quando a fila de sinalização ficar vazia novamente.

O desempenho mínimo requerido em termos de retardo e perda de pacotes pode ser alcançado para os dois tipos de tráfego. Além disso, o esquema também garante uma proteção para cada tipo de tráfego na ocorrência de sobrecarga do outro.

No método de alocação (T_1, T_2) , um esquema de controle de congestionamento é implementado visando reduzir a perda e o retardo de pacotes de voz e de dados em situações de sobrecarga de tráfego. Esse esquema é baseado na técnica de codificação para voz ADPCM (*Adaptive Differential Pulse Code Modulation*) embutida e no emprego de um controlador de congestionamento. A técnica ADPCM embutida codifica amostras de fontes de voz em bits e as organiza em pacotes. Em um pacote de voz os bits são reorganizados em blocos de acordo com a importância (significado) deles nas amostras. O controlador de congestionamento funciona da seguinte forma: antes de transmitir um pacote de voz, é calculada uma medida de congestionamento que é a soma ponderada do número de pacotes nas filas de voz e de dados. Esta medida dá uma noção do estado de carga do sistema. Caso o valor dessa medida ultrapasse certos limites, alguns blocos de bits menos significativos são descartados. O pacote de voz fica então com o tamanho reduzido, levando o multiplexador a servir mais rapidamente os pacotes de voz durante os períodos de congestionamento e assim, reduzindo a perda e o retardo dos pacotes de voz e de dados nas filas.

Os valores dos intervalos de transmissão T_1 e T_2 são da ordem de poucos milissegundos, ou melhor, múltiplos dos tempos de transmissão típicos de pacotes de voz e de dados. A escolha de T_1 e T_2 dessa ordem visa permitir a transmissão de vários pacotes por cada visita do servidor. Devido ao estado da atual tecnologia VLSI (*Very Large-Scale Integration*), os intervalos de *switch-over* (tempo durante o qual o servidor se desloca entre as filas) são da ordem de poucas dezenas de microssegundos, o que é bastante pequeno comparado com o tempo de transmissão típico de um pacote.

Como o tráfego de sinalização é muito menos intenso do que a soma dos tráfegos de voz e de dados, podemos dizer que o esquema (T_1, T_2) garante a alocação de uma largura de banda mínima de $\{T_1/(T_1+T_2)\}C$ para o tráfego de voz e de $\{T_2/(T_1+T_2)\}C$ para o tráfego de dados, onde C é a capacidade total de transmissão do canal. O tráfego de sinalização utiliza aqui apenas uma porção desprezível dessa capacidade. O esquema então, ao garantir uma largura de banda mínima para transmissão, protege um tipo de tráfego de uma eventual sobrecarga do outro.

Como já vimos, os valores dos intervalos de transmissão T_1 e T_2 determinam as larguras de banda mínimas alocadas para os dois tipos de tráfego.

Daí, o retardo por pacote é afetado pela seleção dos intervalos de transmissão T1 e T2. Por exemplo, se o intervalo de transmissão para voz T1 for muito maior que o intervalo de transmissão de dados T2, o retardo para pacotes de voz é baixo, às custas de um alto retardo para pacotes de dados. Por outro lado, se T2 for muito maior que T1, o contrário se verifica. A escolha dos valores dos intervalos de transmissão T1 e T2 visa então ajustar o sistema para atender requerimentos de retardo máximo por pacote e de largura de banda mínima para os tráfegos de voz e de dados.

1.2 O Protocolo de Acesso ao Meio do Padrão IEEE 802.4 *Token Bus*

O padrão IEEE 802.4 *Token Bus* especifica um protocolo de acesso ao meio (barra) por passagem de *token*. O protocolo vê a rede como um grupo de estações conectadas a uma barra formando um anel lógico por onde circula o *token* passando de uma estação para a sua sucessora. O conjunto de estações ativas na rede pode variar no tempo devido a desativações e reativações de estações realizadas dinamicamente. O protocolo implementa então um mecanismo de reconfiguração do anel lógico que é usado para a inclusão ou exclusão de uma estação. O protocolo também emprega um mecanismo para manter o anel lógico operacional mesmo quando ocorrem erros de canal.

O método de acesso ao meio proposto pelo padrão IEEE 802.4 comporta um esquema de prioridades opcional que estabelece 8 (oito) classes de serviço (de 0 a 7) para os tráfegos na camada lógica de enlace. Quanto maior o índice da classe, maior a prioridade associada a ela. As oito classes de serviço são mapeadas em quatro classes de acesso na camada MAC (*Media Access Control*) mantendo-se a mesma relação de prioridades entre elas. Esse mapeamento é feito desprezando-se os bits menos significativos da representação binária das oito classes de serviço. As quatro classes de acesso são nomeadas 0, 2, 4 e 6. Em cada estação, as classes de acesso são vistas como subestações virtuais por onde o *token* é passado internamente de uma classe a outra na ordem 6, 4, 2 e 0, ou seja, em ordem decrescente de prioridade. A classe 6, mais prioritária, é associada a um parâmetro temporizador chamado THT (*hi_pri-Token-Hold-Time*). Cada uma das outras três classes é associada a um outro parâmetro temporizador chamado TRT (*target Token Rotation Time*).

O funcionamento do protocolo é da seguinte forma: quando uma estação recebe o *token*, ela transmite quadros (*frames*) de dados de classe 6 até ela ter transmitido todos os quadros, ou um intervalo de tempo igual ao THT ter expirado, o que acontecer primeiro. Em seguida o *token* é passado às subestações virtuais correspondentes às classes 4, 2 e 0 sucessivamente, e depois à próxima estação no anel. Cada uma dessas subestações virtuais, ao receber o *token*, mede o tempo de ciclo do *token*, que é o tempo levado pelo *token* para retornar à subestação desde a sua última visita. Se o tempo de ciclo de *token* for menor do que seu parâmetro

TRT, então a subestação inicia a transmissão de quadros de dados de sua respectiva classe. A transmissão dura por um período igual a diferença entre seu TRT e o tempo de ciclo do *token* recém medido, ou até que todos os quadros de dados na subestação tenham sido transmitidos (isto é, a subestação tenha se esvaziado), o que acontecer primeiro. Caso a subestação esteja vazia no momento da chegada do *token*, este é passado imediatamente adiante. O tempo de serviço da própria subestação, obviamente, faz parte do próximo tempo de ciclo do *token* medido.

O protocolo é totalmente distribuído. Cada estação decide e controla o seu acesso ao meio para transmissão baseada somente em informações locais. O emprego dos temporizadores (relógios) é simples e barato. Esse mesmo mecanismo de acesso em barra pode ser aplicado em redes de anel com passagem de *token*, sistemas de *polling* e sistemas de *token* implícito. O padrão FDDI *Token Ring*, por exemplo, propõe um protocolo que implementa um mecanismo de acesso bastante similar.

Cada estação, ao receber o *token* em um ciclo, tem uma fatia de tempo igual ao THT, garantida para transmissão do tráfego mais prioritário (classe 6). No entanto, quadros de dados de classes menos prioritárias (classes 4, 2 e 0) só podem ter acesso ao canal para transmissão se o *token* não retornar à subestação ‘atrasado’, ou seja, se o tempo de ciclo do *token* for menor que o TRT da subestação correspondente.

Em situações nas quais o tempo total de transmissão de quadros de dados de classe 6 é maior que todos os TRT's, todas as classes menos prioritárias ficam impossibilitadas de usar o canal. Nessas situações somente o tráfego de classe 6 é transmitido pelo canal. Porém, à medida que a carga de tráfego de classe 6 for diminuindo, as classes menos prioritárias vão sucessivamente obtendo uso do canal em ordem decrescente de seus TRT's, refletindo portanto, as relações de prioridade entre as classes de acesso 4, 2 e 0. Estabelece-se então, uma relação entre as prioridades das classes e os valores dos seus TRT's, de tal forma que quanto maior o TRT, maior a prioridade da classe. A alocação da largura de banda entre as classes menos prioritárias é portanto, diretamente proporcional aos seus parâmetros TRT's.

O protocolo é apropriado para a transmissão, na mesma rede, de vários tipos de tráfego com características e requisitos diferentes. Um tráfego em tempo-real (e. g. voz empacotada) pode ser tratado como tráfego de classe de acesso 6, enquanto que, tráfegos relativamente menos prioritários (e. g. pacotes de dados convencionais) utilizam as outras classes de acesso. Assim, o tráfego em tempo-real tem reservada e assegurada uma certa porção da largura de banda do canal independente da carga dos outros tipos de tráfego. A fatia de tempo disponível para o tráfego em tempo-real é no mínimo igual à soma dos THT's das subestações virtuais de classe 6. Porém, dependendo da sua carga ofertada, o tráfego em tempo-real pode ocupar toda a largura de banda do canal para transmissão. Logo, requisitos restritos de tempo de resposta e vazão podem ser atingidos através de uma escolha conveniente dos parâmetros de temporização. O mecanismo de temporização empregado pelo protocolo pode suportar um grande número de estações e tem um excelente desempenho em termos de vazão (*throughput*). Essas características importantes

são atribuídas à adaptabilidade do protocolo, através do uso de informação passada (duração do ciclo anterior), para regular o acesso ao canal.

O método desenvolvido em [1] só pode modelar e analisar este protocolo, quando este opera em uma rede, onde as estações tratam apenas serviços de classe 6.

1.3 O Protocolo FDDI

O padrão FDDI (*Fiber Distributed Data Interface*) especifica um protocolo para controlar a transmissão de informação, a 100 Mbits/seg, em uma rede local em anel, utilizando fibra óptica como meio físico. O protocolo é baseado no método de acesso *Token Ring*. No início, o padrão FDDI foi proposto como uma rede de comutação de pacotes com duas aplicações: como uma interconexão de alto desempenho entre computadores de grande porte e seus equipamentos periféricos e sub-sistemas de memórias, e como uma rede *backbone* para interligar redes locais de baixa velocidade (e. g. IEEE 802.3, 802.4 e 802.5). O tráfego transmitido na rede pode ser de dois tipos. No primeiro, o tráfego de dados por pacotes ocorre em quantidade e tempo aleatórios, e é denominado como assíncrono. No segundo tipo, o tráfego se apresenta mais regular, ocorrendo em quantidades relativamente determinadas, e em um período regular de tempo. Este tipo de tráfego tem rígidos requisitos em termos de capacidade de transmissão e de retardo. Neste caso, denominamos o tráfego de síncrono. Para a classe de tráfego assíncrono, há uma sub-divisão em duas sub-classes: restritiva e não restritiva. Ambas competem pelo acesso ao anel nas mesmas condições, entretanto, uma vez o *token* tenha sido capturado para ser usado na transmissão de um tráfego restritivo, a largura de banda e o tempo de resposta são garantidos, como se fosse para o tráfego síncrono. A sub-classe não restritiva não tem largura de banda, nem tempo de resposta garantidos. O padrão FDDI sugere também, para a classe de acesso assíncrono, oito níveis diferentes de prioridades, possibilitando-se portanto, distinguir níveis de serviços. Neste caso, um tráfego interativo pode receber um serviço de prioridade maior, e portanto menos atraso, do que por exemplo, uma transferência de arquivo, que é um serviço menos prioritário.

A partir do FDDI original, foi desenvolvido o padrão FDDI-II, o qual adiciona ao padrão básico o tratamento de tráfego por comutação de circuito. Esta extensão visa atender aplicações em tempo real, tais como voz e vídeo, especificando uma rede multi-mídia. Portanto, uma rede FDDI-II pode suportar dois tipos de tráfego, que utilizam comutação de circuito ou comutação de pacotes.

Como foi apresentado em [3], descreveremos agora, de uma maneira geral, o comportamento do protocolo em uma rede local com dois tipos de tráfego: síncrono e assíncrono. Aqui, o tráfego síncrono, pelas suas características, e o tráfego assíncrono, correspondem no padrão FDDI, respectivamente, aos tráfegos assíncronos de sub-classe restritiva, e não restritiva. Este comportamento é similar

ao do protocolo especificado no padrão IEEE 802.4 *Token Bus*, visto anteriormente. Cada estação da rede pode transmitir tráfego síncrono e assíncrono. Quando o *token* chega a uma estação, esta passa a transmitir, se houver, tráfego síncrono por um intervalo de tempo já previamente estabelecido. Findo este intervalo, ou se o tráfego síncrono tiver acabado, a estação poderá transmitir o tráfego assíncrono, por um intervalo de tempo, que dependerá da duração do último ciclo de *token*, medida em relação àquela estação. Se o valor da duração ultrapassar um certo parâmetro de tempo chamado TTRT (*Target Token Rotation Time*), correspondente ao TRT do padrão IEEE 802.4 *Token Bus*, significa que o *token* está atrasado, e portanto a estação não transmitirá o tráfego assíncrono, passando o *token* imediatamente à próxima estação da rede. O protocolo de passagem de *token* com temporização do padrão FDDI apresenta duas importantes propriedades, que foram provadas em [3]. As propriedades são:

1. A duração média do ciclo de *token*, na ausência de falhas na rede, é no máximo igual ao valor do TTRT,
2. A duração máxima do ciclo de *token*, na ausência de falhas na rede, é no máximo duas vezes o valor do TTRT.

Aqui, apenas o comportamento do tráfego síncrono ou do tráfego assíncrono restritivo pode ser modelado e analisado pelo método desenvolvido em [1].

1.4 Trabalhos Relacionados

Nesta seção, alguns trabalhos significativos relacionados com o tema estudado são comentados. No entanto, além destes, ainda existem outros trabalhos não menos significativos com importantes resultados para o estudo da análise e modelagem de sistemas de *polling* com serviço limitado por temporização. Os resultados resumidos abaixo visam dar uma idéia geral de métodos utilizados para analisar protocolos dos tipos apresentados neste capítulo, onde temporizadores são utilizados para alocar capacidade a diferentes tipos de tráfego.

Em [4] uma análise de vazão (*throughput*) sob alta carga é feita para o protocolo apresentado pelo padrão IEEE 802.4 *Token Bus*. A análise também pode ser aplicada ao protocolo de passagem de *token* proposto pelo padrão FDDI *Token Ring*. O interesse de estudo é na modelagem do mecanismo de prioridades entre as várias classes de acesso do protocolo.

O modelo apresentado é baseado em um sistema de filas com um servidor que as atende em ordem cíclica. As filas representam as subestações virtuais que transmitem os diferentes tipos de tráfego, e o servidor representa o *token* circulante. No modelo existem dois tipos de filas: filas do tipo I que representam as subestações virtuais mais prioritárias com classe de acesso 6, e filas do tipo II que representam as outras. As filas podem ser distribuídas arbitrariamente no anel

lógico (a barra ou o anel físico no caso do FDDI) e podem ter diferentes parâmetros de temporização (THT e TRT).

Para poder se estudar o comportamento do protocolo, algumas suposições são feitas visando simplificar a modelagem. A primeira é que os tempos de *switch-over* (transmissão do *token* de uma fila para a sua sucessora) são constantes no tempo. Em vários sistemas de passagem de *token* isto acontece. A segunda suposição considera que em uma fila não há transmissões de partes de quadros após a sua fatia de tempo (THT ou TRT) ter expirado. Isso se aplica a sistema onde a fatia de tempo é múltiplo do tempo de transmissão do quadro. A terceira e última suposição é que o sistema está sob alta carga. Por alta carga entende-se que todas as estações estão sempre com quadros para transmitir e nunca ficam vazias, ou seja, o servidor sempre serve uma fila do tipo I por toda sua fatia de tempo THT, e serve uma fila de tipo II pela diferença entre seu TRT e a duração do ciclo de *token* anterior. A situação de alta carga é a mais crítica para o sistema. Numa configuração de carga estocástica (Poisson, por exemplo) as filas não estão sempre cheias, podendo inclusive em alguns momentos estarem vazias.

A modelagem do protocolo sujeito a configurações genéricas de carga é bastante difícil, pois é preciso estudar-se a dinâmica do mecanismo de temporização em conjunto com a dinâmica estocástica da carga oferecida. Então, a hipótese restritiva de alta carga é útil na análise pois permite estudar o mecanismo de temporização separado da dinâmica introduzida pelo processo estocástico de chegada de pacotes ao sistema. O comportamento do protocolo é modelado através de uma cadeia de Markov finita e determinística, onde o estado da cadeia é um vetor com os tempos de serviço das filas em um ciclo de *token*.

Um resultado obtido no artigo é que, sob alta carga, o tempo de ciclo do *token* é limitado superiormente pelo maior TRT. Então é possível controlar os tempos de ciclo de *token*, e conseqüentemente o retardo de transmissão, com uma escolha apropriada dos parâmetros TRT. Outro resultado importante é que estes tempos de ciclo só podem variar entre os parâmetros TRT de duas classes de prioridade sucessivas. Além do mais, algumas filas podem ficar sem receber serviço dependendo dos valores relativos dos parâmetros TRT. No artigo é mostrado que, em um sistema simétrico de filas do tipo II (ou seja, todas com o mesmo valor de TRT), todas as filas têm o mesmo tempo de ciclo médio e recebem o mesmo serviço médio. Também é mostrado no artigo que a alocação da largura de banda, sob alta carga, é independente da posição relativa das filas no anel lógico. Resultados analíticos para cálculo de vazão e tempos de ciclos de *token* foram derivados.

Embora os resultados obtidos não se apliquem ao caso geral onde as filas estão sujeitas, por exemplo, a uma carga Poisson, pode-se obter informações de vazão em momentos de sobrecarga do sistema. Também é possível estabelecer parâmetros do sistema (e. g. THT, TRT, número de filas) visando satisfazer certos requisitos de projeto.

Em [5] uma fila M/G/1 com férias e serviço *gated* e limitado por tempo é analisada e usada para modelar um caso particular do multiplexador (T1,

T2) proposto por Sriram em [2]. O serviço é limitado por tempo pois o servidor, quando de uma visita, pode servir a fila por um limite máximo de tempo pré-fixado. Após esse tempo o servidor tira férias e o serviço é interrompido, sendo reiniciado na próxima visita do servidor. Se todos o pacotes forem servidos antes deste limite, o servidor também tira férias. A característica *gated* do serviço se refere ao fato de que quando o servidor volta de férias, apenas os pacotes que se encontram na fila naquele instante serão servidos. Os pacotes que continuarem a chegar só serão servidos na próxima (ou em outra posterior) visita do servidor. O modelo utilizado é um modelo de férias múltiplas pois, se a fila estiver vazia quando o servidor voltar de férias, ele imediatamente tira outras férias.

Este modelo de fila M/G/1 com férias pode ser utilizado para se estudar aproximadamente vários sistemas de computação e comunicação onde temporizadores são empregados para alocar serviço a vários tipos de tráfego. Além de um caso particular do multiplexador (T1,T2), os padrões IEEE 802.4 e IEEE 802.5 para redes com passagem de permissão (*token*) e o protocolo FDDI são alguns exemplos. No caso da ausência de *overrun* (tempo excedente de posse do *token*, durante o qual a estação continua a transmitir após a sua quota de tempo expirar), estes sistemas podem ser modelados para estimar o retardo por pacote em uma estação particular, quando as outras estações servem seus pacotes pelo tempo máximo de retenção do *token* em cada visita do servidor.

No artigo, o modelo é utilizado para estudar um caso específico e limitante do esquema (T1,T2), onde a transmissão de pacotes é interrompida durante os períodos de *switch-over*, a disciplina de serviço emprega uma política *gated* e é assumido que um dos tipos de tráfego apresenta alta carga, ou seja, o servidor sempre serve a sua respectiva fila até um limite de tempo pré-fixado durante cada visita. Neste caso limitante, esta fila nunca está vazia. O retardo é estudado na fila que não está sob alta carga e o período de férias, aqui constante, representa o tempo que o servidor leva para servir a outra fila que abriga o tipo de tráfego que está sob alta carga.

O tempo médio de resposta (retardo) de um pacote é associado com o montante médio de trabalho visto por um pacote ao chegar à fila. Trabalho aqui é visto como tempo de serviço requerido por um pacote. Essa associação é obtida baseada na propriedade PASTA (*Poisson Arrivals See Time Averages*) e na decomposição estocástica, que permite que um sistema M/G/1 com férias seja estudado separadamente em duas partes, analisando o comportamento do sistema sem férias e o comportamento do sistema quando o servidor está em férias.

Uma equação funcional para o montante de trabalho, no instante da volta do servidor das férias (instante de *polling*) é derivada. Salvo no caso especial onde o limite de tempo de serviço da fila é infinito, essa equação tem uma solução analítica complexa. Uma técnica numérica de aproximação por soma ponderada de funções de Laguerre com coeficientes desconhecidos é então empregada. Resultados numéricos para o tempo médio de resposta de um pacote são obtidos para uma distribuição de tempo de serviço exponencial.

Um modelo de simulação, com eventos discretos é usado em [2] para estudar o comportamento do multiplexador (T1,T2). A carga de pacotes oferecida em uma fonte de voz é simulada com uma distribuição geométrica em intervalos regulares e iguais durante uma conversação (*talkspurt*). Entre cada conversação, um período de silêncio é simulado por uma distribuição exponencial. Para simular o tráfego de dados, os tempos entre chegadas de pacotes são considerados independentes e distribuídos geometricamente ou hiper-exponencialmente. O tamanho do pacote de dados é assumido fixo ou com distribuição exponencial. Variando os parâmetros da distribuição hiper-exponencial, tráfegos de dados com diferentes características podem ser simulados.

Os valores dos parâmetros de tráfego escolhidos para simulação foram baseados em sistemas reais como, por exemplo, o sistema IACS (*Integrated Access and Cross-connect System*) projetado pela AT&T *Bell Laboratories*. Em média, de 28% a 42% do tempo total de conexão é gasto com atividades (transmissões) de voz. Dois casos são considerados na simulação: atividade de voz de 35% e de 42%. Isto representa taxas ofertadas de voz de 22 e de 26,25 pacotes por segundo, respectivamente. O tamanho de um pacote de voz é de 74 bytes, incluindo um *header* de 10 bytes. O tamanho de um pacote de dados é distribuído geometricamente com média de 60 bytes ou, em alguns casos, é determinístico e igual a esta média.

Neste modelo, um pacote é perdido quando seu tempo de espera na fila ultrapassa um certo limite máximo permitido e ele é então descartado. Assim, a probabilidade de bloqueio de um pacote é estimada como a frequência com a qual essa perda acontece. Em situações práticas de carga, verifica-se que a probabilidade de bloqueio de pacotes de voz é desprezível devido ao mecanismo de *block-dropping* implementado que impede a perda de pacotes de voz mesmo em situações de sobrecarga.

Neste estudo, os *buffers* para voz e dados devem acomodar até 60 e 120 pacotes, respectivamente. A taxa de transmissão (capacidade) do canal usada é a de um *link* T1, ou seja, 1.544 Mbits/seg. Resultados para retardo e probabilidade de bloqueio de pacotes de dados e de voz, dentre outros, são apresentados para várias configurações de tráfego. O esquema (T1,T2) é , também, comparado com a disciplina de serviço FIFO (*First In First Out*).

Em [6], Takagi apresenta um modelo de avaliação de desempenho para analisar um protocolo simples de passagem de permissão (*token*) usando temporização. O objetivo do trabalho é estudar uma simplificação para o protocolo FDDI. No modelo apresentado, cada estação tem *buffer* unitário, ou seja, armazena somente uma mensagem por vez. As mensagens podem ser de dois tipos: prioritárias ou comuns. O direito à transmissão é diferenciado para uma mensagem quanto ao seu tipo. Uma mensagem prioritária é sempre transmitida, caso exista, quando um *token* chega à estação. Já uma mensagem comum só é transmitida se o tempo de rotação do *token* não tiver excedido o parâmetro de tempo pré-fixado TTRT (*Target Token Rotation Time*). O tempo de rotação do *token* é definido como o intervalo entre duas visitas sucessivas do *token* a uma estação. Os pacotes de voz e de dados, em uma rede integrada real, são representados no modelo pelas mensagens prioritárias

e comuns, respectivamente.

No protocolo simplificado modelado, o atraso na rotação do *token* não é acumulado, como no FDDI, mas considerado apenas no ciclo anterior. Como consequência da suposição de buffer de capacidade unitária, o tempo de posse do *token* por uma estação é o tempo gasto com a transmissão de uma única mensagem. O tempo de transmissão de uma mensagem qualquer é considerado constante. Seja b este tempo. Este comportamento é diferente no FDDI, já que neste o intervalo de transmissão para uma estação ordinária é igual à diferença entre o TTRT e o tempo de rotação do *token*. Todas as estações são assumidas estatisticamente idênticas, ou seja, as estações são simétricas. Isso simplifica as equações desenvolvidas no modelo. No entanto, uma extensão para sistemas assimétricos pode ser feita com algumas alterações.

No modelo estudado, uma rede em anel é representada por N estações com *buffer* unitário. As mensagens prioritárias e comuns são geradas por processos de chegada *Poisson* estacionários com taxas λ' e λ , respectivamente. As mensagens que, ao chegar, encontrarem o *buffer* ocupado são descartadas. O *token* representa o direito de transmissão, e circula no anel passando por cada estação a velocidade constante de tal forma que, quando não há nenhuma transmissão, consome um tempo R para dar um ciclo completo. Esse tempo representa alguns atrasos operacionais, tais como: períodos de *switch-over* entre as estações, retardo de propagação de transmissão, etc. O TTRT é definido como sendo $R + Mb$, onde M é um inteiro tal que $0 \leq M \leq N$. Dois casos particulares do protocolo podem ser observados. Primeiro, quando $M = N$, nenhuma distinção é feita entre as mensagens quanto ao tipo (prioritárias ou comuns), ou seja, a mensagem no *buffer* é sempre transmitida quando a estação obtém o direito de transmissão (um *token* chega). O segundo caso, quando $M = 0$, corresponde a uma rede onde as mensagens comuns só podem ser transmitidas se não houve transmissões, de qualquer tipo, no ciclo anterior.

O espaço de estados modelado é N -dimensional e tem 4^N estados, ou seja, é exponencial. As probabilidades de estado são calculadas a partir de várias equações lineares de transição de estado. A partir destas probabilidades, as diversas medidas de desempenho, estudadas no artigo, são calculadas através de fórmulas fechadas. Para se estudar o comportamento do sistema, pontos embutidos são selecionados no tempo. Estes pontos são os instantes nos quais um *token* chega a uma estação. As transições entre os estados ocorrem então entre as visitas do *token* a duas estações subsequentes.

Em redes integradas de voz e dados, pacotes de dados podem suportar um retardo longo, diferentemente dos pacotes de voz, que requerem um serviço rápido (tráfego *real-time*). Por outro lado, em alguns casos, os pacotes de voz podem suportar uma probabilidade de bloqueio maior do que os pacotes de dados. Visando estudar estes casos, uma variação no protocolo é proposta. Agora uma mensagem comum tem prioridade preemptiva (*push-out*) na ocupação do *buffer* sobre uma mensagem prioritária, ou seja, se uma mensagem comum for gerada e, ao chegar, encontrar o *buffer* ocupado por uma mensagem prioritária, esta é expulsa e a mensagem comum é armazenada em seu lugar. Entretanto se a mensagem prioritária

no *buffer* estiver sendo transmitida, ela não será expulsa. Nestes casos, mensagens comuns conseguem uma vazão maior do que as mensagens prioritárias. Por outro lado, as mensagens prioritárias têm retardos menores, o que aqui é o esperado.

Algumas medidas de desempenho como vazão, retardo médio para mensagens prioritárias e comuns, tempo médio de rotação do *token* e utilização do *buffer* são calculadas analiticamente. A influência do TTRT sobre o desempenho das mensagens prioritárias e comuns é demonstrada. A sua escolha controla o tempo médio de rotação do *token* e afeta a diferenciação entre as mensagens prioritárias e comuns quanto ao retardo médio. Pode-se observar que, no modelo sem prioridade preemptiva, a vazão e a utilização são praticamente insensíveis ao valor do TTRT.

Em [7] é apresentado um esquema de *token* com prioridades em cujo modelo uma rede em anel simétrica com passagem de *token* é considerada. O *token* circula ciclicamente entre as estações e o tempo gasto por ele para se mover de uma estação para a próxima é assumido constante. Este tempo é um parâmetro chave que afeta o desempenho do sistema. As mensagens que chegam a cada estação são divididas em duas classes de prioridade: classe 1 (alta prioridade) e classe 2 (baixa prioridade). Mensagens de alta prioridade têm requisitos rígidos de baixos retardos de acesso. Cada estação só pode armazenar no máximo uma única mensagem de alta prioridade em seu *buffer* por vez. O modelo de *buffer* unitário é usado para representar a estação como um gerador de mensagens de classe 1 com fonte finita, de modo que, no máximo uma única mensagem de alta prioridade (classe 1) esteja pronta para transmissão por vez, ou seja, no máximo uma conexão em tempo-real é estabelecida pela estação por vez. Também, este modelo é usado para representar um sistema de filas com *buffer* limitado onde uma mensagem de alta prioridade é bloqueada, caso chegue a uma estação cujo *buffer* esteja ocupado. Por outro lado, as mensagens de classe 2, menos prioritárias, têm requisitos para retardo mais relaxados, e podem ser armazenadas em cada estação, em um *buffer* de capacidade ilimitada. Os fluxos de mensagens chegando às estações para as duas classes de prioridade são assumidos como sendo processos Poisson independentes. Para simplificar, o comprimento (neste caso, o tempo de transmissão) de uma mensagem de uma classe é considerado constante, podendo ser diferente do da outra classe. Para as mensagens de alta prioridade é garantido acesso ao canal por um certo período de tempo pré-fixado. Já as mensagens de baixa prioridade são servidas apenas quando é determinado, através do uso de *tokens* circulantes, que não existe nenhuma mensagem de alta prioridade no momento aguardando para ser transmitida. O protocolo *Token-Ring* do padrão IEEE 802.5 emprega um método similar. Dois esquemas (políticas) de disciplina para o serviço das mensagens de baixa prioridade são analisados. Estes esquemas são chamados: exaustivo e limitado. Mais uma vez, o protocolo *Token-Ring* do padrão IEEE 802.5 utiliza um esquema de acesso ao meio similar. Portanto, a análise de desempenho do modelo do artigo também reflete seu desempenho quando operado sob condições semelhantes de tráfego e de política de serviço. Como resultados, foram derivadas fórmulas exatas para o cálculo de retardo médio e vazão para mensagens de alta e de baixa prioridade, e também de probabilidade de bloqueio para as mensagens de alta prioridade. Uma importante observação obtida dos experimentos é que o tráfego de classe 2 afeta o desempenho de retardo

e vazão das mensagens de classe 1, porém de uma forma limitada, assegurando mais prioridade destas últimas no acesso ao canal para transmissão.

Em [8] são derivadas, a partir de um modelo de *polling*, duas aproximações analíticas para o cálculo do retardo médio em uma rede com temporização especificada no protocolo de automação de manufatura MAP (*Manufacturing Automation Protocol*). O protocolo MAP é um emergente padrão industrial para comunicação de dados em indústrias modernas que empregam automação controlada por computador em suas linhas de produção. O protocolo é uma especificação para comunicações de dados em banda larga utilizando passagem de *token* em redes com topologia de barra, e é baseado no modelo de referência OSI (*Open System Interconnection*) da ISO (*International Organization for Standardization*). Ele é um sub-conjunto dos padrões estabelecidos para protocolos das sete camadas do modelo, acrescido de algumas outras normas específicas de manufatura para a camada de aplicação. No artigo, é estudada a sub-camada MAC (*Media Access Control*) da camada de enlace do MAP, que é igual ao protocolo de passagem de *token* do padrão IEEE 802.4 para redes em barra com transmissão em banda larga. A primeira aproximação, baseada no resultado de Fuhrmann para disciplina de serviço *k*-limitado, é para o caso simétrico de redes homogêneas, onde todas as estações têm a mesma carga de tráfego e valores iguais para os parâmetros THT e TRT. É assumido que a disciplina de serviço é não preemptiva. Daí, o tempo excedente de posse do *token* (*overrun*) é estudado. Também, a utilização máxima da rede, em função dos parâmetros THT e TRT, é obtida e usada nas expressões para aproximação do retardo médio por mensagem. A aproximação foi comparada com resultados de simulações e se demonstrou bastante precisa em casos práticos. A segunda aproximação é para o caso assimétrico, de redes não homogêneas, onde tem-se uma estação de baixa prioridade (limitada pelo TRT) saturada, e as demais estações da rede são de alta prioridade com serviço limitado pelo THT. Inicialmente é derivada uma expressão para estimar a redução da largura de banda disponível para a estação de baixa prioridade devido ao aumento da carga ofertada pela estações de alta prioridade. A partir daí, esta expressão é usada para se obter uma aproximação para o cálculo do retardo médio de mensagens nas estações de alta prioridade. Os resultados aqui também foram validados por simulações. As duas fórmulas são bastante úteis no projeto e operação de redes com o protocolo MAP, auxiliando na escolha dos parâmetros de temporização THT e TRT para satisfazer diversos requisitos de retardo para serviços com diferentes prioridades.

Capítulo 2

O Método Proposto

O método analítico de modelagem e análise usando sistemas de *polling* com tempos de serviço limitados (*time-outs*) estudado e implementado neste trabalho foi proposto em [1]. O objetivo é calcular algumas medidas de desempenho em estado estacionário, tais como: probabilidades para o número de pacotes em cada fila, tempo médio de espera na fila para um pacote, e probabilidades de bloqueio ou perda de pacotes no caso de *buffer* limitado. Para facilitar o entendimento dos capítulos seguintes, repetiremos o desenvolvimento de [1]. Inicialmente descreveremos o modelo adotado. A seguir introduzimos alguns conceitos básicos para o desenvolvimento da teoria. Finalmente, o método de modelagem e análise é apresentado.

2.1 Descrição do Modelo

O sistema de *polling* é modelado como um conjunto de M filas visitadas ciclicamente por um servidor. As filas representam as estações e o servidor a permissão para uso do canal (transmissão). Dizemos que uma estação recebe a permissão para transmitir quando esta recebe o *token*, ou seja, a visita do servidor. Cada fila i tem um *buffer* de tamanho B_i , onde ficam armazenadas as unidades de informação (e. g. pacotes, mensagens, *jobs*, quadros) a serem transmitidas. O modelo também é válido para o caso teórico de *buffer* infinito. Usando a nomenclatura da teoria das filas, podemos ver as unidades de informação como clientes que concorrem para receber o serviço de transmissão. Em cada fila i , clientes chegam de acordo com um processo Poisson de taxa λ_i e são servidos obedecendo um processo exponencial com taxa μ_i , que é a taxa de serviço da fila. O servidor visita as filas na seguinte ordem: da fila i para fila $(i \bmod M) + 1$. O tempo gasto pelo servidor para se locomover de uma fila i à fila seguinte é chamado de intervalo de *switch-over* e tem uma distribuição de probabilidade geral $S_i(\cdot)$ com média σ_i . Assumiremos neste trabalho, no entanto, que o intervalo de *switch-over* é determinístico, ou seja, constante. A extensão para outras distribuições é simples e será alvo de futuros estudos. Os dois processos

Poisson (chegada e serviço), bem como os tempos de *switch-over* são assumidos todos independentes entre si.

O sistema de *polling* com temporização estudado aqui emprega um temporizador (relógio) associado a cada fila (estação), que limita o tempo de permanência do *token* na fila. O tempo máximo de serviço para uma fila é chamado de quota de tempo de serviço. O que difere este tipo de sistema de um sistema de STD_M (*Synchronous Time-Division Multiplexing*), por exemplo, é que o servidor deixa imediatamente a fila, para servir a seguinte, se esta ficar vazia antes da sua quota de tempo expirar. Já em um sistema de STD_M, o servidor fica disponível para uma fila por todo um tempo de serviço pré-estabelecido, mesmo se a fila ficar vazia antes, ou até se já estiver vazia na chegada do servidor. Como pode ser observado, bastante recurso pode ser desperdiçado. Esse desperdício se dá pois o servidor pode eventualmente ficar dedicado a uma fila que não tenha informação para transmitir, enquanto que outras filas com informação estejam esperando recurso. A fatia ou quota de tempo associada a cada fila pode ser constante ou variável no tempo. Porém no sistema aqui estudado ela é assumida constante e igual a T_i para fila i . Note que se $T_i \rightarrow \infty$, o sistema se comporta igual ao TDM (*Time-Division Multiplexing*) estatístico, ou seja, o servidor só deixa uma fila, para ir servir outra, quando esta fica vazia, independentemente do tempo de serviço já gasto na fila. Quando por ocasião de um *time-out* (servidor deixar a fila), um cliente não tiver seu serviço completado, o protocolo pode adotar duas políticas diferentes: o cliente permanece na fila até a próxima visita do servidor, e daí continuará a ser servido a partir do ponto no qual foi interrompido (como na política *preemptive-resume* em um sistema de filas com prioridade), ou então é permitido ao cliente terminar o seu serviço (como os *overruns* no protocolo FDDI). Na segunda política, o servidor obviamente fica disponível à fila i um tempo maior que T_i . O método de modelagem e análise proposto em [1] se presta para modelar as duas políticas acima, porém a ferramenta desenvolvida neste trabalho e os estudos feitos só consideram a primeira. A política mais geral que pode ser considerada nesse tipo de sistema é a que não inicia um serviço de um cliente ‘‘perto’’ do *time-out* T_i expirar. Para definir melhor ‘‘perto’’, um parâmetro w_i é associado a cada fila i , tal que, se o serviço de um cliente terminar dentro de um intervalo de tempo w_i antes do final de T_i , o servidor deixa a fila para ir servir a próxima. Note que se $w_i = 0$, então o servidor serve a fila até o *time-out* T_i expirar (salvo se a fila ficar vazia antes), quando então, uma das duas políticas citadas anteriormente são aplicadas.

Cada fila pode ter uma disciplina de serviço diferente. As principais disciplinas consideradas em [1] são: exaustiva, *gated*, exaustiva-limitada (*E-limited*) e *gated*-limitada (*G-limited*). Na disciplina exaustiva, tanto os clientes presentes em uma fila i , quando da chegada do servidor, quanto outros clientes que eventualmente cheguem à fila durante a visita do mesmo, podem ser servidos. Na disciplina *gated*, somente os clientes que estejam na fila i , no instante de chegada do servidor, podem ser servidos. A disciplina exaustiva-limitada é similar à disciplina exaustiva, com a única diferença de que existe um certo número máximo de pacotes que podem ser servidos em uma visita do servidor. Por último, a disciplina *gated*-limitada é semelhante à disciplina *gated*, mas com a única diferença de que também há um número

máximo de clientes que podem ser servidos em uma visita do servidor. Obviamente, esse número só é relativo aos clientes que estejam na fila quando da chegada do servidor. Outras disciplinas, tais como a semi-exaustiva, também podem ser modeladas e analisadas através da mesma abordagem básica. A ferramenta desenvolvida e os estudos feitos neste trabalho só consideram a disciplina exaustiva, comum a todas as filas, ficando as demais para estudos futuros. Os clientes podem ser servidos em qualquer ordem desde que não haja preempção e a escolha do próximo cliente a ser servido seja independente do seu tempo de serviço requerido.

2.2 Alguns Conceitos Básicos

Antes de apresentarmos o método de modelagem e análise propriamente dito, daremos a seguir uma introdução sobre a teoria básica necessária ao seu desenvolvimento: randomização e cadeia de Markov com recompensa.

2.2.1 Randomização

A técnica de randomização ou uniformização é um método geral para computação de probabilidades transientes em processos Markovianos com espaço de estados finito. Porém, o método também pode ser usado em diversos processos Markovianos com espaço de estados infinito, quando as características da cadeia de Markov são previamente conhecidas, ou podemos aproximar o processo por um processo Markoviano com espaço de estados finito. A técnica de randomização permite uma interpretação probabilística que pode ser utilizada para limitar o erro de truncamento advindo das somas infinitas, conduzindo assim a uma modelagem mais eficiente e confiável. A técnica envolve a discretização de uma cadeia de Markov de tempo contínuo e é baseada na subordinação de uma cadeia de Markov a um processo Poisson [9].

Seja $Z = \{Z_n, n = 0, 1, \dots\}$ uma cadeia de Markov em um espaço de estados contável $S = \{S_i, i = 1, 2, \dots\}$ e com matriz de transição P . Seja $N = \{N(t), t \geq 0\}$ um processo de contagem Poisson com taxa Λ , ou seja, $N(t)$ indica o número de ocorrências no intervalo $[0, t]$. Assim, $P\{N(t) = n\} = e^{-\Lambda t}(\Lambda t)^n/n!$. Assumindo que Z e N são independentes, define-se o processo $U = \{U(t) = Z_{N(t)}, t \geq 0\}$. O processo U é um processo de Markov de tempo contínuo em S , com matriz geradora $Q = \Lambda(P - I)$ e com a mesma distribuição inicial do processo Z . Nesta construção, dizemos que a cadeia de Markov Z está subordinada ao processo Poisson N .

Por outro lado, seja $U = \{U(t), t \geq 0\}$ um processo Markoviano de tempo contínuo em um espaço de estados contável $S = \{S_i, i = 1, 2, \dots\}$ com matriz geradora Q . Seja α_i a taxa total de saída do estado S_i . Assumamos que U é uniformizável, ou seja, as taxas totais de saída dos estados são uniformemente limitadas. Isto é, existe uma taxa finita Λ tal que $\Lambda \geq \alpha_i$, para todo i . Então existe uma cadeia de Markov de tempo discreto $Z = \{Z_n, n = 0, 1, \dots\}$ em S , com matriz

de transição $P = Q/\Lambda + I$, e um processo Poisson $N = \{N(t), t \geq 0\}$ com taxa Λ , independentes entre si tais que, os processos $\{Z_{N(t)}, t \geq 0\}$ e $\{U(t), t \geq 0\}$ sejam equivalentes probabilisticamente. A discretização é feita adicionando-se, a cada estado S_i , retro-transições com taxa $\Lambda\alpha_i$, convertendo então o processo U em um processo equivalente com taxa total de saída de cada estado igual a Λ . Obviamente, apenas α_i/Λ das transições realmente deixam o estado.

Obtemos, através desta construção, uma fórmula simples e bastante útil para se calcular as probabilidades transientes de um processo Markoviano uniformizável. Condiçãoando no número de eventos do processo Poisson N no intervalo $[0, t]$, e usando o teorema da probabilidade total, temos:

$$\begin{aligned} \pi_S(t) &= P\{U(t) = S\} \\ &= P\{Z_{N(t)} = S\} \\ &= \sum_{n=0}^{\infty} P\{Z_{N(t)} = S \mid N(t) = n\}P\{N(t) = n\} \\ &= \sum_{n=0}^{\infty} P\{Z_n = S\}e^{-\Lambda t}(\Lambda t)^n/n!. \end{aligned}$$

Seja $\phi_S(n) = P\{Z_n = S\}$ e $\phi(n) = (\phi_1(n), \phi_2(n), \dots, \phi_S(n), \dots)$. Agora, usando esta definição na equação anterior e usando uma notação vetorial temos:

$$\pi(t) = \sum_{n=0}^{\infty} \phi(n)e^{-\Lambda t}(\Lambda t)^n/n!$$

onde $\pi(t) = (\pi_1(t), \pi_2(t), \dots, \pi_S(t), \dots)$. Verificando que $\pi(0) = \phi(0)$ e que P é a matriz de transição de $\{Z_n, n = 0, 1, 2, \dots\}$, podemos reescrever a equação anterior como

$$\pi(t) = \sum_{n=0}^{\infty} \pi(0)P^n e^{-\Lambda t}(\Lambda t)^n/n!.$$

2.2.2 Cadeia de Markov com Recompensas

Uma recompensa (*reward*) é uma variável aleatória que é associada a um processo de Markov. Ela representa uma determinada medida que estamos interessados em quantificar, durante a evolução de um sistema sendo modelado, em um certo intervalo de tempo. Formalizando, seja \mathcal{M} uma determinada medida de interesse. Sem perda de generalidade, seja também $\mathcal{Y} = \{Y_k; k = 0, 1, \dots\}$ uma cadeia de Markov embutida, com um espaço de estados $S = \{a_i; i = 1, \dots, M\}$, identificada nos instantes $\{\tau_k; k = 0, 1, \dots\}$ com $\tau_0 = 0$. Por exemplo, podemos definir a recompensa \mathcal{M}_i como sendo igual ao valor de \mathcal{M} em um intervalo (τ_{k-1}, τ_k) , dado que $Y_{k-1} = a_i$. A recompensa \mathcal{M}_i é portanto associada ao estado a_i da cadeia embutida de Markov \mathcal{Y} . Neste trabalho, as recompensas são independentes de k . Podemos obter valores de \mathcal{M} transientes e em estado estacionário.

Primeiro consideraremos resultados para estado estacionário. Seja

$\mathcal{M}(k)$ a recompensa total incondicional durante $(0, \tau_k)$. De [10], nós temos, independentemente do estado no instante 0,

$$\lim_{k \rightarrow \infty} \frac{\mathcal{M}(k)}{k} = \sum_{i=1}^M E[\mathcal{M}_i] \beta_i = \lim_{k \rightarrow \infty} \frac{E[\mathcal{M}(k)]}{k} \quad (2.1)$$

onde a primeira igualdade se verifica com probabilidade 1. Aqui β_i é a probabilidade em estado estacionário do estado a_i na cadeia de Markov embutida. M é o número de estados da cadeia, e o vetor $\beta = \langle \beta_1, \dots, \beta_M \rangle$ pode ser obtido através da solução da equação $\beta = \beta D$, onde D é a matriz de transição da cadeia de Markov embutida \mathcal{Y} .

Agora consideraremos resultados para medidas transientes no intervalo $(0, \tau_k)$. Para calcular $E[\mathcal{M}(k)]$, nós primeiro calculamos \mathcal{M}_i para todos os estados a_i , e então descondicionamos estes resultados baseado na distribuição de probabilidade de estado em τ_j , com $j = 0, 1, \dots, k-1$. Seja $\beta_i(j) = P(Y_j = a_i)$, e $\beta(j) = \langle \beta_1(j), \dots, \beta_M(j) \rangle$ o vetor de probabilidade de estado correspondente. Então, com $\beta(j) = \beta(j-1)D$, nós temos

$$E[\mathcal{M}(k)] = \sum_{i=1}^M E[\mathcal{M}_i] \sum_{j=0}^{k-1} \beta_i(j). \quad (2.2)$$

Uma vez calculados a matriz de transição D e os valores esperados $\{E[\mathcal{M}_i]; i = 1, \dots, M\}$, as equações 2.1 e 2.2 podem ser usadas para obtermos medidas transientes e em estado estacionário. Os conceitos e resultados recém apresentados serão usados neste trabalho para o cálculo das medidas de desempenho.

2.3 O Método de Modelagem e Análise

O método de modelagem e análise é composto de duas etapas: inicialmente determinaremos a distribuição de probabilidade de estado do sistema em estado estacionário. Na segunda etapa, com esta distribuição, calcularemos as medidas de interesse à luz de resultados da teoria de processos regenerativos e cadeias de Markov com recompensas. Para calcularmos a distribuição de probabilidade de estado, é preciso primeiro determinar uma cadeia de Markov que represente o comportamento do sistema. Como veremos a seguir, essa cadeia será uma cadeia de Markov embutida. A técnica de randomização ou uniformização é empregada no cálculo da matriz de transição da cadeia de Markov embutida e das medidas de interesse.

2.3.1 A Cadeia de Markov Embutida

Seja o processo estocástico $\mathcal{X} = \{X(t); t \geq 0\}$, onde $X(t) = \langle x_1(t), \dots, x_M(t) \rangle$ e $x_j(t)$ é o número de clientes do sistema na fila j no tempo t . O espaço de estados de

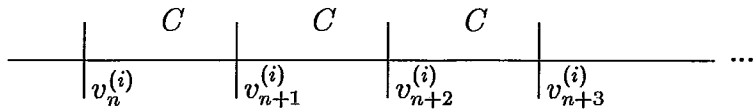


Figura 2.1: Ciclos de *Polling* e a Matriz C

\mathcal{X} é o conjunto de M -uplas de inteiros $S = \{ \langle q_1, \dots, q_M \rangle; 0 \leq q_j \leq B_j \}$, onde q_j indica o número de clientes na fila j , que tem um *buffer* de dimensão B_j . Embora a notação esteja considerando modelos com *buffer* limitado, a extensão para *buffer* infinito é simples. Devido ao fato de as quotas de tempo de serviço serem constantes, dentre outros fatores, \mathcal{X} não é um processo Markoviano. No entanto, pontos de regeneração podem ser identificados no tempo, como por exemplo, os instantes quando o servidor visita uma certa fila i e todas as filas do sistema estão vazias. Mesmo o processo \mathcal{X} não sendo um processo Markoviano, podemos identificar uma cadeia de Markov embutida nos instantes quando o servidor sucessivamente visita uma determinada fila i . Sejam os instantes $\{v_1^{(i)}, v_2^{(i)}, \dots\}$ esses pontos embutidos no tempo. Embora tais pontos no tempo não sejam pontos de regeneração para \mathcal{X} , a cadeia de Markov embutida de tempo discreto $\mathcal{Y}^{(i)} = \{Y_n^{(i)}; n = 1, 2, \dots\}$, dada por $Y_n^{(i)} = X(v_n^{(i)})$, é obtida verificando \mathcal{X} nesses pontos. Seja $\beta^{(i)} = \langle \beta_s^{(i)} \rangle$, com $s \in S$, o vetor de distribuição de probabilidade de estado do sistema, em estado estacionário, para a cadeia de Markov embutida $\mathcal{Y}^{(i)}$, onde $\beta_s^{(i)}$ é a probabilidade do estado s . Esta distribuição será depois usada para calcularmos as medidas de desempenho desejadas. Para obtermos $\beta^{(i)}$, a matriz de transição de $\mathcal{Y}^{(i)}$ deve ser antes determinada. É válido observar que existem tantas cadeias quantas são filas do sistema. Embora o processo \mathcal{X} não seja Markoviano durante todo tempo, ele o é durante um intervalo de tempo entre a chegada do servidor a uma determinada fila i e a sua partida, quer seja devido ao esvaziamento da fila ou devido à expiração da quota de tempo de serviço T_i . Isto é fácil de observar, pois os processos de chegada são Poisson e os tempos de serviço são exponenciais. Faz-se necessário então, o emprego de análise transiente para o cálculo das probabilidades de estado no final do intervalo. A técnica utilizada para a análise transiente é a randomização.

O servidor fica ciclicamente alternando intervalos de tempo, onde ele serve uma determinada fila com intervalos de tempo onde ele se desloca de uma fila para a seguinte durante um *switch-over*. Sem perda de generalidade, observaremos a cadeia embutida nos instantes de tempo onde uma certa fila fixada é visitada pelo servidor. Portanto daqui para frente eliminaremos o superescrito i .

Relembramos que o espaço de estados do sistema, representado pela cadeia de Markov embutida \mathcal{Y} , é dado por $S = \{ \langle q_1, \dots, q_M \rangle; 0 \leq q_j \leq B_j \}$. Portanto, um estado do sistema é representado pela M -upla $s = \langle q_1, \dots, q_M \rangle$ onde q_j é o número de clientes na fila j . A cadeia de Markov embutida \mathcal{Y} é de tempo discreto e C é a sua matriz de transição entre os estados do sistema. Devemos determinar C , para então obtermos $\beta = \langle \beta_s \rangle$, com $s \in S$, através da equação $\beta = \beta C$. A matriz

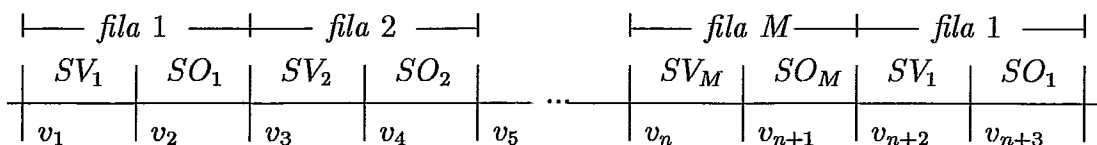
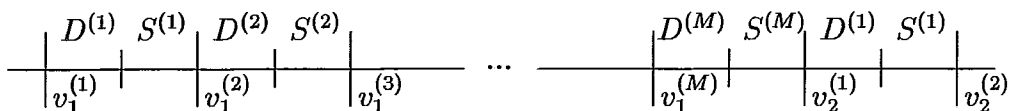


Figura 2.2: Mini-ciclos de Serviço e Intervalos de *Switch-over*

de transição C é determinada a partir do comportamento do sistema durante um ciclo de *polling*, que é definido aqui como o intervalo de tempo entre duas visitas consecutivas do servidor à fila fixada. Na figura 2.1, alguns ciclos de *polling* estão indicados no eixo do tempo, juntamente com alguns pontos embutidos. A matriz C contém as probabilidades de transição de estado entre dois pontos embutidos adjacentes. Como β é um vetor de probabilidades em estado estacionário, ele pode ser calculado pelo método de potência, iterativamente até convergir. O teste de convergência é feito nos pontos embutidos. No entanto, tratamos aqui o ciclo de *polling* como um conjunto de M ‘mini-ciclos’ de serviço e M intervalos de *switch-over*. Durante um mini-ciclo de serviço, o servidor está servindo uma certa fila e durante um intervalo de *switch-over* o servidor está se movendo de uma fila para a próxima.

Identificamos agora um novo conjunto de pontos embutidos. Além dos instantes de chegada do servidor em cada fila, também são considerados pontos embutidos, os instantes de início de intervalos de *switch-over*. Obviamente, estes instantes correspondem também aos instantes de partida do servidor destas filas. Sejam v_1, v_2, \dots tais pontos embutidos no tempo. Na figura 2.2, encontramos juntamente com os pontos embutidos, os mini-ciclos de serviço e intervalos de *switch-over* indicados no eixo do tempo. Os mini-ciclos indicados com SV_i e SO_i , com $i = 1, \dots, M$, representam, respectivamente, mini-ciclos de serviço e intervalos de *switch-over* para a fila i . Os pontos embutidos v_i , para i ímpar, são os instantes de chegada do servidor às filas. Já os pontos embutidos v_i , para i par, são os instantes de partida do servidor, ou início de intervalos de *switch-over*, onde ele se desloca para ir servir outra fila. Este tratamento é conveniente aqui, pois como já vimos, o processo que representa o comportamento do sistema só é Markoviano durante os intervalos de tempo onde o servidor serve uma determinada fila, que são justamente os mini-ciclos de serviço. Portanto, em face das características estocásticas do sistema de *polling* com *time-out* em questão, não mais obteremos C para todo o ciclo de *polling*, mas por partes, para os mini-ciclos de serviço e os intervalos de *switch-over* que o compõem.

Definindo mais formalmente, seja $s = \langle q_1, \dots, q_M \rangle \in S$, onde q_j é o número de clientes na fila j ($j = 1, \dots, M$), e também da mesma forma, seja $s' = \langle q'_1, \dots, q'_M \rangle \in S$. O mini-ciclo durante o qual a j -ésima fila é servida é chamado de mini-ciclo j , e o intervalo de *switch-over* durante o qual o servidor se desloca da fila j para a fila $(j \bmod M) + 1$ é chamado de intervalo de *switch-over* j . Seja $D^{(j)}$ a matriz de transição de estado para um mini-ciclo j , onde cada elemento

Figura 2.3: Matrizes D e S

(s, s') seu, dado por $d_{s,s'}^{(j)}$, indica a probabilidade de que o estado do sistema seja s' no final de um mini-ciclo j dado que o estado do sistema no início do mini-ciclo j era s . Da mesma forma, definimos $S^{(j)}$ como sendo a matriz de transição de estado para um intervalo de *switch-over* j . Sem perda de generalidade, seja a cadeia $\mathcal{Y}^{(i)}$ identificada na fila $i = 1$. Daí, é fácil ver, a partir de [1], que

$$C = \prod_{j=1}^M D^{(j)} S^{(j)}.$$

Em [1], C é calculada a partir das matrizes de transição de estado durante os mini-ciclos de serviço e os intervalos de *switch-over* para cada fila visitada em um ciclo de *polling*. Os instantes de início de um período de *switch-over*, que correspondem a partidas do servidor da fila precedente, serão considerados também pontos embutidos no tempo para efeito do cálculo do vetor de distribuição β . Agora, β não mais será obtido através da equação $\beta = \beta C$, e sim, passo a passo, nos pontos embutidos associados a cada fila e a cada período de *switch-over*. Assim, a partir de um vetor distribuição inicial, β sofrerá iterações, sendo sucessivamente computado pela equação $\beta(k) = \beta(k-1)D^{(j)}$ ou $\beta(k) = \beta(k-1)S^{(j)}$ (onde k indica um passo de iteração, e $\beta(k)$ representa o vetor β após esse passo), dependendo se é um mini-ciclo j ou um período de *switch-over* j , até completar um ciclo de *polling* para a fila fixada. Ao se completar um ciclo, um novo vetor β é obtido. Como β é um vetor de distribuição em estado estacionário, ele deverá ser repetidamente calculado até convergir, isto é, $\beta = \lim_{k \rightarrow \infty} \beta(k)$. Com este método para obter β através de mini-ciclos, é possível obter todos os vetores de distribuição β em todos os pontos embutidos, sem ter que resolver uma equação $\beta = \beta C$. Na figura 2.3, vemos no eixo do tempo, as matrizes D e S e os respectivos mini-ciclos de serviço e intervalos de *switch-over*. As matrizes D e S contêm as probabilidades de transição de estado entre dois pontos embutidos adjacentes. O teste de convergência de β é feito sempre em um certo ponto embutido fixado no eixo do tempo. Por exemplo, se fixarmos o instante de chegada do servidor à fila 1, então devemos fazer o teste de convergência nos pontos embutidos $v_1^{(1)}$, $v_2^{(1)}$, etc. Vamos agora ao cálculo das matrizes $S^{(j)}$ e $D^{(j)}$.

Cálculo de $S^{(j)}$

Calcularemos inicialmente a matriz de transição $S^{(j)}$, com $j = 1, \dots, M$, pois é a mais simples. Seja $S_{s,s'}^{(j)}$ a probabilidade de transição de $s = \langle q_1, \dots, q_M \rangle$ para $s' = \langle q'_1, \dots, q'_M \rangle$ durante o intervalo de *switch-over* j . Durante a transição

nenhuma fila recebe serviço algum, pois o servidor está se locomovendo da fila j para a próxima fila $(j \bmod M) + 1$. Portanto a mudança de estado se dá exclusivamente por chegadas de novos clientes às filas durante um intervalo de tempo fixo dado pelo *switch-over* j , ou seja, da fila j . Como os processos de chegada são independentes, então a probabilidade conjunta de chegadas a todas as filas é dada pelo produto das probabilidades de chegadas em cada fila individualmente. Também, uma fila não pode ter seu tamanho diminuído durante tal intervalo, pois clientes que já estejam na fila não podem sair sem que sejam servidos. Temos então

$$S_{s,s'}^{(j)} = \begin{cases} 0 & \text{se } q'_i < q_i \forall i, i = 1, \dots, M \\ \prod_{i=1}^M p(k_i, \lambda_i, \sigma_j) & \text{caso contrário} \end{cases} \quad (2.3)$$

onde $k_i = q'_i - q_i$ é o número de clientes que chegaram e foram armazenados na fila i , durante o intervalo de *switch-over* j , $p(k, \lambda, t)$ é a probabilidade de que k chegadas de um processo Poisson tenham sido armazenadas na fila com taxa λ durante um tempo t , e σ_j é a duração do intervalo de *switch-over* da fila j .

Para modelos de filas com *buffer* de capacidade infinita, temos que $p(k, \lambda, t) = e^{-\lambda t} (\lambda t)^k / k!$, pois neste caso, todas as chegadas à fila são sempre armazenadas. Já no caso de *buffer* limitado, uma pequena modificação no cálculo é feita. Se o estado $q'_i \in s'$ de uma certa fila i , no final de um intervalo de *switch-over* j , for igual ao tamanho do seu respectivo *buffer* B_i (o *buffer* está lotado de pacotes), então pode ter havido $q'_i - q_i$ ou mais chegadas à fila i , dentre as quais, apenas $q'_i - q_i$ chegadas foram armazenadas. Isso acontece pois, uma vez o *buffer* cheio, as demais chegadas não são mais armazenadas e não alteram o estado da fila. Obviamente, no caso de *buffers* ilimitados, todas as chegadas são aceitas. Portanto temos para o caso de *buffer* limitado esta nova definição:

$$p(k, \lambda, t) = \begin{cases} \frac{(\lambda t)^k}{k!} e^{-\lambda t} & \text{se } q'_i \neq B_i \\ \sum_{k=q'_i - q_i}^{\infty} \frac{(\lambda t)^k}{k!} e^{-\lambda t} & \text{se } q'_i = B_i \end{cases} \quad (2.4)$$

Visando eliminar o somatório infinito, obtemos

$$p(k, \lambda, t) = \begin{cases} \frac{(\lambda t)^k}{k!} e^{-\lambda t} & \text{se } q'_i \neq B_i \\ 1 - \sum_{k=0}^{q'_i - q_i - 1} \frac{(\lambda t)^k}{k!} e^{-\lambda t} & \text{se } q'_i = B_i. \end{cases} \quad (2.5)$$

Cálculo de $D^{(j)}$

Vamos agora determinar as matrizes de transição de estado $D^{(j)}$ para um mini-ciclo j . Relembremos que cada elemento (s, s') de $D^{(j)}$, indicado por $d_{s,s'}^{(j)}$, é a probabilidade de que o estado do sistema seja $s' = \langle q'_1, \dots, q'_M \rangle$ no final do mini-ciclo j dado que no início do mini-ciclo j era $s = \langle q_1, \dots, q_M \rangle$, onde q_i e q'_i representam os estados da fila i no início e no final do mini-ciclo j respectivamente. Quatro tipos de políticas de serviço para sistemas de *polling* são tratadas em [1]: exaustivo, *gated*, *gated*-limitado (G-limitado) e exaustivo-limitado (E-limitado). Apesar das regras serem distintas, é fácil a partir das equações para serviço exaustivo, obter as

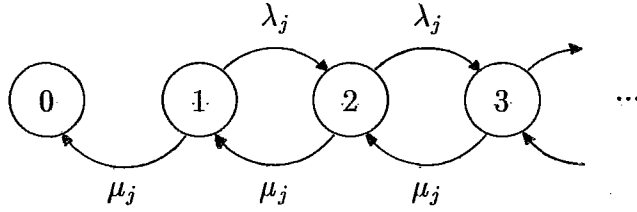


Figura 2.4: Cadeia de Markov para Serviço Exaustivo

equações para os outros tipos de política de serviço. Entretanto, abaixo apresentaremos apenas o desenvolvimento para serviço exaustivo, uma vez que os nossos resultados numéricos são obtidos para este tipo de serviço. Para modelos com *buffer* limitado, a obtenção das expressões segue um raciocínio similar ao desenvolvimento apresentado.

Serviço Exaustivo Seja um mini-ciclo j onde o estado do sistema no seu início é $s = \langle q_1, \dots, q_M \rangle$. O mini-ciclo poderá terminar, com $s' = \langle q'_1, \dots, q'_M \rangle$, de duas maneiras: pelo término da quota de tempo associada à fila j , durando neste caso T_j unidades de tempo, ou antes, por esvaziamento da fila antes de T_j . Portanto, no final do mini-ciclo, o estado da fila j , sendo servida, é $q'_j > 0$ no primeiro caso, ou $q'_j = 0$ no segundo caso. Vamos dividir o problema em dois casos, onde τ_j é o tamanho do mini-ciclo j :

1. $q'_j > 0$ e $\tau_j = T_j$
2. $q'_j = 0$ e $0 < \tau_j < T_j$

A alteração de estado nas outras filas é determinada apenas por novas chegadas Poisson durante o intervalo do mini-ciclo j . Como já vimos, algumas destas chegadas serão aceitas e outras rejeitadas, dependendo do espaço de armazenamento das filas. Vamos considerar então, durante um mini-ciclo j , a fila j sendo servida e as outras filas $i \neq j$, que têm comportamentos similares, sendo alvo apenas de novas chegadas de pacotes.

Para os dois casos, as filas $i \neq j$, que não estejam sendo servidas, não podem ter seus tamanhos reduzidos, pois clientes já na fila não podem sair sem que antes recebam serviço. No caso de existir i , tal que $q'_i < q_i$, então $d_{s,s'}^{(j)} = 0$. Os seus tamanhos podem apenas aumentar devido às novas chegadas $k_i = q'_i - q_i$, ou simplesmente não sofrerem alteração por ausência de chegadas.

Quando $q_j = 0$, ou seja, a fila j a ser servida já está vazia quando o servidor chega, o mini-ciclo j tem duração zero ($\tau_j = 0$). Como o mini-ciclo j não acontece, os estados de todas as filas devem permanecer inalterados. Não houve serviço na fila j , nem chegadas às outras filas $i \neq j$. Obviamente $q'_j = 0$, e $d_{s,s'}^{(j)} = 0$, se existir i , tal que $q'_i \neq q_i$.

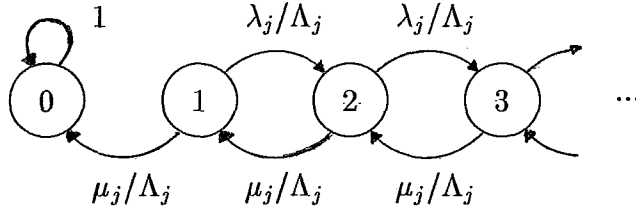


Figura 2.5: Cadeia de Markov Randomizada para Serviço Exaustivo

Em [1], o comportamento da fila j sendo servida durante o seu mini-ciclo j é modelado por uma cadeia de Markov $\mathcal{W}^{(j)}$ de tempo contínuo com o estado representando o número de clientes na fila. Sob serviço exaustivo, a cadeia de Markov $\mathcal{W}^{(j)}$ é um processo de nascimento e morte com parâmetros λ_j e μ_j , e com o estado 0 como um estado absorvente. A existência deste estado absorvente modela o fato do servidor abandonar a fila j quando ela se torna vazia. O diagrama de transição de estados para $\mathcal{W}^{(j)}$ com serviço exaustivo é ilustrado na figura 2.4. Para modelos com *buffer* finito, a cadeia é truncada no estado B_j , onde B_j é o tamanho do *buffer* da fila j .

A distribuição para o tamanho da fila j (número de clientes armazenados) no final do mini-ciclo j pode ser encontrada a partir do cálculo do comportamento transiente da cadeia $\mathcal{W}^{(j)}$ no intervalo de tempo $(0, T_j)$. Temos em [1] que, embora o mini-ciclo j possa durar menos do que T_j , é equivalente calcular a distribuição para o final do mini-ciclo j no instante T_j . Isso se dá pois o estado 0 é absorvente. O cálculo transiente, como já havíamos previsto, é feito usando a técnica da randomização. Partindo de um vetor inicial de distribuição $\pi^{(j)}(0)$ para o tamanho da fila j no início do seu mini-ciclo j , a distribuição no final do mini-ciclo é dada por

$$c^{(j)} = \sum_{n=0}^{\infty} e^{-\Lambda_j T_j} \frac{(\Lambda_j T_j)^n}{n!} \pi^{(j)}(n) \quad (2.6)$$

onde $\Lambda_j = \lambda_j + \mu_j$ e $\pi^{(j)}(n) = \pi^{(j)}(n-1)W^{(j)}$, com $W^{(j)}$ sendo a matriz de transição da cadeia de Markov randomizada obtida de $\mathcal{W}^{(j)}$, que está ilustrada na figura 2.5. Convém estabelecer aqui, para efeito de notação, que em um vetor de probabilidade v qualquer, v_i indica a sua i -ésima posição. O vetor inicial de distribuição $\pi^{(j)}(0)$ é dado por $\pi_{q_j}^{(j)}(0) = 1$ (ou equivalentemente $\pi^{(j)}(0) = e_{q_j}$), pois q_j é o estado inicial da fila j . Verifica-se ainda que $c_0^{(j)}$ é a probabilidade da fila j , condicionada a um estado inicial, ter-se esvaziado antes de T_j , levando conseqüentemente o mini-ciclo j a acabar também antes de T_j .

Antes de iniciarmos o cálculo das probabilidades de transição $d_{s,s}^{(j)}$, para os dois casos acima, derivaremos a função distribuição para a duração τ_j do mini-ciclo j . A distribuição para τ_j é obtida a partir de um vetor de distribuição inicial $\pi^{(j)}(0)$ do número de clientes na fila. Seja então $F(t) \stackrel{\text{def}}{=} P[\tau_j \leq t]$, com $0 \leq t < T_j$, a probabilidade da fila j , sendo servida, estar em um estado absorvente no instante de tempo t . Vamos definir também $\pi_{abs}^{(j)}(n)$ como sendo a probabilidade da fila estar em um estado absorvente após o n -ésimo passo da cadeia. Logo temos

que

$$\pi_{abs}^{(j)}(n) = \sum_i \pi_i^{(j)}(n); \quad \forall i \in S_{abs}$$

onde S_{abs} é o conjunto de estados absorventes da cadeia de Markov $\mathcal{W}^{(j)}$, e $\pi_i^{(j)}(\cdot)$ é a i -ésima posição do vetor $\pi^{(j)}(\cdot)$. Sob serviço exaustivo, como a cadeia tem somente o estado 0 como estado absorvente, $\pi_{abs}^{(j)}(n) = \pi_0^{(j)}(n)$. Em particular, para $t = 0$, temos que $F(0) = P[\tau_j = 0] = \pi_{abs}^{(j)}(0)$ é a probabilidade de que o servidor visite uma fila vazia j . Aqui, os estados no início e no final do mini-ciclo são iguais, ou seja, $s = s'$. Agora, aplicando os conceitos de randomização para cálculo de probabilidades transientes, temos para $0 < t < T_j$

$$F(t) = \sum_{n=0}^{\infty} e^{-\Lambda_j t} \frac{(\Lambda_j t)^n}{n!} \pi_{abs}^{(j)}(n) \quad (2.7)$$

onde $\pi_{abs}^{(j)}(n)$ é a probabilidade de estar no estado absorvente 0 no n -ésimo passo da cadeia de Markov randomizada $W^{(j)}$ dado $\pi^{(j)}(0)$ como distribuição inicial. A função densidade de probabilidade é dada pela derivada $F'(t)$. Em $t = 0$, $F'(0)$ é dada por um impulso de valor igual a $\pi_{abs}^{(j)}(0)$. Para $0 < t < T_j$ [1]:

$$F'(t) = \sum_{n=1}^{\infty} e^{-\Lambda_j t} \frac{(\Lambda_j t)^{n-1}}{(n-1)!} \Lambda_j \{ \pi_{abs}^{(j)}(n) - \pi_{abs}^{(j)}(n-1) \}. \quad (2.8)$$

Vemos que $P_{abs}^{(j)}(n) \stackrel{def}{=} \pi_{abs}^{(j)}(n) - \pi_{abs}^{(j)}(n-1)$, para $n > 0$, é a probabilidade da fila entrar em um estado absorvente exatamente no n -ésimo passo da cadeia de Markov randomizada. Chegamos então a

$$F'(t) = \begin{cases} \pi_{abs}^{(j)}(0) u_0(t) & \text{se } t = 0 \\ \sum_{n=1}^{\infty} e^{-\Lambda_j t} \frac{(\Lambda_j t)^{n-1}}{(n-1)!} \Lambda_j P_{abs}^{(j)}(n) & \text{se } 0 < t < T_j. \end{cases} \quad (2.9)$$

Claramente, no caso 2, $F'(t)$, para $0 < t < T_j$, é a função densidade para os eventos (q'_j, τ_j) com $q'_j = 0$ e $0 < \tau_j < T_j$.

Para o caso 1, onde o mini-ciclo j termina com $q'_j > 0$ e $\tau_j = T_j$, o cálculo de $d_{s,s'}^{(j)}$ é simples pois o mini-ciclo j é constante com duração T_j . A probabilidade de um evento (q'_j, T_j) com $q'_j > 0$, representativo deste caso, é igual a probabilidade da cadeia de Markov $\mathcal{W}^{(j)}$ estar no estado q'_j no instante T_j . Esta probabilidade é calculada por randomização usando a cadeia de Markov $W^{(j)}$ correspondente a $\mathcal{W}^{(j)}$. Além disto, devemos considerar também as outras filas $i \neq j$ que não estão sendo servidas. Nestas filas os seus estados são modificados apenas por novas chegadas $k_i = q'_i - q_i$ de clientes durante o intervalo constante T_j . Aqui, podemos tratar o comportamento de mudança de estado durante o mini-ciclo de uma forma independente. Daí

$$\begin{aligned} d_{s,s'}^{(j)} &= P[\mathcal{W}^{(j)}(T_j) = q'_j \mid \mathcal{W}^{(j)}(0) = q_j] \prod_{i \neq j} p(k_i, \lambda_i, T_j) \\ &= c_{q'_j}^{(j)} \prod_{i \neq j} p(k_i, \lambda_i, T_j) \end{aligned}$$

Substituindo $c_{q_j}^{(j)}$ na equação acima, obtemos finalmente

$$d_{s,s'}^{(j)} = \left\{ \sum_{n=0}^{\infty} e^{-\Lambda_j T_j} \frac{(\Lambda_j T_j)^n}{n!} \pi_{q_j}^{(j)}(n) \right\} \prod_{i \neq j} p(k_i, \lambda_i, T_j) \quad (2.10)$$

onde $\pi_{q_j}^{(j)}(0) = 1$ e $p(k, \lambda, t)$ é definido da mesma maneira que no cálculo da matriz de transição $S^{(j)}$ durante um *switch-over*, ou seja, como sendo a probabilidade de que k chegadas de um processo Poisson, com uma taxa λ , tenham sido armazenadas na fila durante um intervalo de tempo t .

Lembremos que $p(k, \lambda, t) = e^{-\lambda t} (\lambda t)^k / k!$ nos modelos de filas com *buffers* ilimitados, pois todas as chegadas são aceitas e armazenadas. Portanto, para *buffer* ilimitado temos

$$d_{s,s'}^{(j)} = \left\{ \sum_{n=0}^{\infty} e^{-\Lambda_j T_j} \frac{(\Lambda_j T_j)^n}{n!} \pi_{q_j}^{(j)}(n) \right\} \prod_{i \neq j} \frac{(\lambda_i T_j)^{k_i}}{k_i!} e^{-\lambda_i T_j}.$$

Reescrevendo esta equação com $\gamma = \sum_{i \neq j} \lambda_i + \Lambda_j$ e $k_j = \sum_{i \neq j} k_i$, é obtido em [1]

$$d_{s,s'}^{(j)} = \sum_{n=k_j}^{\infty} e^{-\gamma T_j} \frac{(\gamma T_j)^n}{n!} \prod_{i \neq j} \left(\frac{\lambda_i}{\gamma} \right)^{k_i} \left(\frac{\Lambda_j}{\gamma} \right)^{n-k_j} \frac{n!}{(n-k_j)! \prod_{i \neq j} k_i!} \pi_{q_j}^{(j)}(n-k_j). \quad (2.11)$$

De acordo com [1], a equação 2.11 tem uma interpretação probabilística. Consideremos a superposição de M processos Poisson com taxas Λ_j e λ_i , onde $i \neq j$, durante um intervalo de tempo T_j . A equação indica o cálculo da probabilidade de haver k_i transições do processo i , com $i \neq j$, durante o intervalo, e de que a cadeia de Markov $W^{(j)}$ esteja no estado q_j' após $n - k_j$ transições, condicionada na ocorrência de n eventos do processo composto, onde $n \geq k_j$. Depois, há o descondicionamento no número de eventos do processo composto.

No capítulo 3, daremos detalhes de como computar a equação 2.11 de uma maneira mais fácil e eficiente, avaliando também os custos computacionais de processamento e armazenamento envolvidos. Para modelos de filas com *buffer* finito, a função $p(k, \lambda, t)$ na equação 2.10 para cada fila i , com $i \neq j$, é calculada através da equação 2.4. Nestes modelos não existe uma fórmula ‘fechada’ como a equação 2.11. Porém, conseguimos computar $d_{s,s'}^{(j)}$, de uma forma recursiva a partir das transições da cadeia de Markov randomizada correspondente à fila j .

Para o caso 2, onde o mini-ciclo j termina com $q_j' = 0$ e $0 < \tau_j < T_j$, o cálculo de $d_{s,s'}^{(j)}$, é baseado no fato de que houve absorção (a fila j sendo servida se esvaziou, no caso exaustivo) antes de terminar sua fatia de tempo T_j , e de que nas outras filas somente houve chegadas durante o mini-ciclo. A alteração do estado nas filas não sendo servidas se dá pelas novas chegadas aceitas $k_i = q_i' - q_i$, para $i \neq j$. A alteração do estado da fila j sendo servida é de q_j para $q_j' = 0$ devido à absorção. Note que o mini-ciclo j aqui não mais dura T_j , e sim menos. Mais ainda, este intervalo é variável. A expressão para $d_{s,s'}^{(j)}$, neste caso deverá representar

a probabilidade de k_i chegadas às filas i , tal que $i \neq j$, e de que o servidor deixe a fila j antes que a sua quota de tempo para serviço expire, ou seja, o mini-ciclo j dure menos do que a quota T_j , ou seja, $\tau_j < T_j$. Portanto, inicialmente calcularemos as probabilidades de k_i chegadas às filas $i \neq j$, condicionadas na duração τ_j , do mini-ciclo j , constante e igual a t . Depois descondicionamos utilizando a função densidade de probabilidade $F'(t)$ para a duração do mini-ciclo, condicionada no estado q_j da fila j no início do seu mini-ciclo. Lembrando que $F'(t) = P[t - dt \leq \tau_j \leq t + dt]$, temos

$$d_{s,s'}^{(j)} = \int_0^{T_j} \prod_{i \neq j} p(k_i, \lambda_i, t) \{F'(t) | q_j\} dt \quad (2.12)$$

onde $p(k, \lambda, t)$ é mais uma vez definido como a probabilidade de que k chegadas de um processo Poisson, com uma taxa λ durante um tempo t , tenham sido armazenadas na fila.

Para modelos com *buffer* ilimitado, onde $p(k, \lambda, t) = e^{-\lambda t} (\lambda t)^k / k!$, e substituindo $F'(t)$ da equação 2.9, temos [1]

$$\begin{aligned} d_{s,s'}^{(j)} &= \sum_{n=k_j+1}^{\infty} e^{-\gamma T_j} \frac{(\gamma T_j)^n}{n!} \\ &\times \sum_{m=k_j+1}^n \prod_{i \neq j} \left(\frac{\lambda_i}{\gamma}\right)^{k_i} \left(\frac{\Lambda_j}{\gamma}\right)^{m-k_j} \frac{(m-1)!}{(m-k_j-1)! \prod_{i \neq j} k_i!} P_{abs}^{(j)}(m-k_j) \end{aligned} \quad (2.13)$$

onde $\pi_{q_j}^{(j)}(0) = 1$, $\gamma = \sum_{i \neq j} \lambda_i + \Lambda_j$ e $\Lambda_j = \lambda_j + \mu_j$.

Esta equação também pode ser interpretada probabilisticamente a partir da superposição de M processos Poisson representando as chegadas às filas $i \neq j$ e transições da cadeia de Markov randomizada $W^{(j)}$ associada à fila j . O índice n indica o número de transições do processo composto durante um intervalo igual a T_j . A transição na qual houve a absorção na fila j é indicada por m . Dois tipos de transições podem ocorrer na cadeia: chegadas às filas $i \neq j$ (tipo i) e transições na própria fila j (tipo j). Portanto, ocorreram k_i transições do tipo i , para $i \neq j$, e $m - \sum_{i \neq j} k_i$ transições do tipo j . A última das transições foi do tipo j e significou a transição para um estado absorvente, que no caso de serviço exaustivo é a transição para o estado 0, ou seja, a fila se tornou vazia.

Veremos no capítulo 3 como computar a equação 2.13 eficientemente em termos de custos computacionais e de armazenamento. Lembramos que esta equação só se aplica no caso de filas com *buffer* ilimitado. Para modelos com filas com *buffer* finito, $p(k, \lambda, t)$, para cada fila $i \neq j$, é definido na equação 2.12 como na equação 2.4. É claro ver que não há uma expressão ‘fechada’ como a equação 2.13. No próximo capítulo veremos como calcular $d_{s,s'}^{(j)}$, no caso 2, para estes modelos com *buffer* finito, a partir da evolução da cadeia de Markov randomizada correspondente à fila j . Usaremos um raciocínio baseado em recursões semelhante ao feito para o caso 1. Uma vez determinadas as matrizes $S^{(j)}$ e $D^{(j)}$, podemos computar β e depois as medidas de desempenho desejadas.

2.3.2 Algumas Medidas de Desempenho

Na seção anterior, uma cadeia de Markov embutida foi identificada nos instantes nos quais o servidor visita uma determinada fila i . Calculamos a matriz de transição C da cadeia de Markov embutida $\mathcal{Y}^{(i)}$, para depois obtermos o vetor de distribuição em estado estacionário $\beta^{(i)}$. Para tal, poderíamos solucionar a equação $\beta^{(i)} = \beta^{(i)}C$, mas vimos que é mais conveniente dividirmos o ciclo de *polling* em mini-ciclos para então calcular C e, conseqüentemente $\beta^{(i)}$, por partes. Assumindo que o vetor $\beta^{(i)}$ foi obtido eficientemente, vejamos agora como usá-lo para calcular algumas medidas de desempenho de interesse com o auxílio da teoria de cadeias de Markov com recompensas. Nesta seção, inicialmente mostraremos como calcular probabilidades de estado com média no intervalo de tempo $[0, \infty)$, e não mais pontualmente no tempo, como o vetor $\beta^{(i)}$. Isso é equivalente a determinar probabilidades de estado nos instantes de chegada de clientes ao sistema, pois pela propriedade PASTA (*Poisson Arrivals See Time Averages*), um cliente ao chegar vê o estado do sistema como sendo uma média no tempo. A partir destas probabilidades, algumas medidas de desempenho podem ser calculadas, tais como: comprimento médio das filas (número médio de pacotes nas estações), retardo médio nas filas e probabilidade de perda de pacotes devido a *buffer* cheio em modelos com *buffer* finito. Analogamente à seção anterior, faremos o desenvolvimento segundo [1], para facilitar a compreensão dos capítulos seguintes. As equações serão obtidas basicamente para modelos com *buffer* infinito, porém serão indicadas, quando necessário, as devidas alterações para o caso de *buffer* limitado.

Seja $P_{\mathcal{R}}$ a probabilidade limitante (durante todo o tempo) que o processo $\mathcal{X} = \{X(t); t \geq 0\}$ esteja em um certo subconjunto $\mathcal{R} \subset S$ dos estados de \mathcal{X} , ou seja, $P_{\mathcal{R}} = \lim_{t \rightarrow \infty} P[X(t) \in \mathcal{R}]$. Por exemplo, \mathcal{R} pode ser o subconjunto de estados, onde o comprimento de uma determinada fila i tem um certo valor, ou seja, $\mathcal{R} = \{s = \langle q_1, \dots, q_i, \dots, q_M \rangle; s \in S \text{ e } q_i = q\}$. É assumido que as probabilidades limitantes existem, garantindo assim a ergodicidade do processo regenerativo \mathcal{X} . Usando o teorema para processos regenerativos ergódicos [10], temos

$$P_{\mathcal{R}} = \lim_{t \rightarrow \infty} \frac{[\text{tempo total gasto em } \mathcal{R} \text{ durante } (0, t)]}{t} \quad (2.14)$$

ou ainda

$$P_{\mathcal{R}} = \frac{E[\text{tempo gasto em } \mathcal{R} \text{ durante um ciclo de renovação}]}{E[\text{duração de um ciclo de renovação}]}$$

Embora os instantes $\{v_n^{(i)}; n \geq 1\}$ de visita do servidor a uma determinada fila i não representem pontos de renovação para o processo \mathcal{X} , é possível expressar a probabilidade limitante $P_{\mathcal{R}}$ em função de um ciclo de *polling* [1], ou seja, durante o intervalo entre duas visitas consecutivas do servidor. Para tal, definamos r_s como uma recompensa igual ao tempo total gasto no subconjunto de estados \mathcal{R} pelo processo, durante um ciclo de *polling*, dado que o sistema estava no estado s no início deste ciclo. É fácil observar que se $\mathcal{R} = S$, o conjunto de todos os estados, então r_s é igual a duração do próprio ciclo de *polling* iniciado no estado s , representada pela variável randomica L_s .

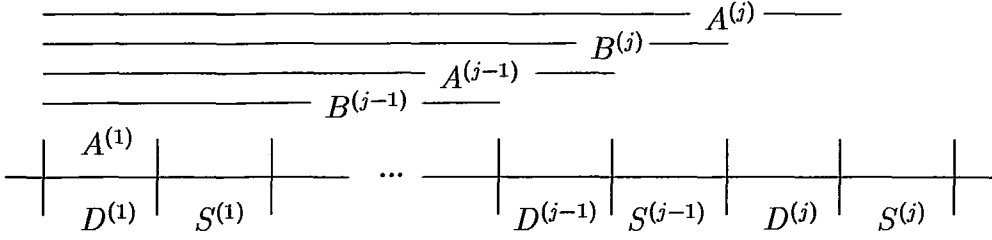


Figura 2.6: Matrizes D , S , A e B

Tomemos então, o k -ésimo ciclo de *polling* para a fila i como sendo o intervalo de tempo $(v_{k-1}^{(i)}, v_k^{(i)})$. Seja $r_s(k)$ a recompensa igual ao tempo total gasto em \mathcal{R} pelo processo \mathcal{X} durante o intervalo $(0, v_k^{(i)})$, dado que $Y_0^{(i)} = s$, ou seja, $X(v_0^{(i)}) = s$. O sub-escrito 0 aqui indica o instante inicial 0 antes que a primeira transição de $Y^{(i)}$ aconteça. De [10], temos para $\forall s' \in S$

$$\lim_{k \rightarrow \infty} \frac{r_{s'}(k)}{k} = \sum_{s \in S} E[r_s] \beta_s^{(i)} = \lim_{k \rightarrow \infty} \frac{E[r_{s'}(k)]}{k} \quad (2.15)$$

onde $\beta_s^{(i)}$ é o s -ésimo elemento do vetor de distribuição de probabilidade $\beta^{(i)}$, que satisfaz a equação $\beta^{(i)} = \beta^{(i)}C$. A partir de agora, omitiremos o super-escrito i , visando simplificar a notação. Seja também $M(t)$ como o número de transições da cadeia embutida de Markov $\mathcal{Y}^{(i)}$ ocorridas até o instante de tempo t . De [1], é possível mostrar que, independentemente do estado inicial s'

$$\lim_{t \rightarrow \infty} \frac{r(v_{M(t)})}{M(t)} = \sum_{s \in S} E[r_s] \beta_s$$

e, usando $\mathcal{R} = S$, obtemos

$$\lim_{t \rightarrow \infty} \frac{v_{M(t)}}{M(t)} = \sum_{s \in S} E[L_s] \beta_s.$$

Finalmente, chegamos a

$$P_{\mathcal{R}} = \frac{\sum_{s \in S} E[r_s] \beta_s}{\sum_{s \in S} E[L_s] \beta_s}. \quad (2.16)$$

Portanto, a partir da equação 2.14, é possível expressar a probabilidade limitante $P_{\mathcal{R}}$ em termos de medidas envolvendo o comportamento da cadeia embutida durante um ciclo de *polling*. Sem perda de generalidade, todas as medidas a seguir serão calculadas durante o ciclo de *polling* iniciado com a chegada do servidor à fila 1, ou seja, a cadeia embutida de Markov $\mathcal{Y}^{(i)}$ é identificada na fila $i = 1$. No entanto, já vimos que é mais conveniente trabalharmos com mini-ciclos e intervalos de *switch-over*. No final então, somamos os valores esperados individuais.

Seja então $U_s^{(j)}$ o tempo gasto pelo processo \mathcal{X} no subconjunto \mathcal{R} durante um mini-ciclo j , dado que o estado do sistema era s no início do mini-ciclo.

Seja também $V_s^{(j)}$ o tempo gasto em R durante um intervalo de *switch-over* j , dado que o estado no início do intervalo era s . Usando estas definições e condicionamentos, $E[r_s]$ é expressado em [1] como sendo

$$E[r_s] = \sum_{j=1}^M \sum_{s' \in S} E[U_{s'}^{(j)}] B_{s,s'}^{(j)} + \sum_{j=1}^M \sum_{s' \in S} E[V_{s'}^{(j)}] A_{s,s'}^{(j)}$$

onde $B^{(j)}$ é a matriz de probabilidades de transição do início do ciclo de *polling* (assumido aqui como o instante da chegada do servidor à fila 1) ao início do mini-ciclo j , e $A^{(j)}$ é a matriz de transição do início do ciclo de *polling* ao início do intervalo de *switch-over* j . Verifica-se que $B^{(1)} = I$ (a matriz identidade), $B^{(j)} = A^{(j-1)} S^{(j-1)} = \prod_{i=1}^{j-1} D^{(i)} S^{(i)}$, para $j \in \{2, \dots, M\}$, e $A^{(j)} = B^{(j)} D^{(j)}$, para $j \in \{1, \dots, M\}$. Na figura 2.6, temos vários mini-ciclos de serviço e intervalos de *switch-over*, dispostos no eixo do tempo, e associadas a eles as matrizes $D^{(j)}, S^{(j)}, A^{(j)}$ e $B^{(j)}$. Reescrevendo o numerador da equação 2.16, temos [1]

$$\sum_{s \in S} E[r_s] \beta_s = \sum_{j=1}^M \sum_{s \in S} E[U_s^{(j)}] \beta_s^{(j)} + \sum_{j=1}^M \sum_{s \in S} E[V_s^{(j)}] \alpha_s^{(j)}$$

onde $\beta^{(j)}$ e $\alpha^{(j)}$ são os vetores de distribuição de probabilidade, em estado estacionário, para as cadeias embutidas de Markov identificadas respectivamente no instante de início de um mini-ciclo j , e no instante de início de um intervalo de *switch-over* j . Verifica-se que $\beta^{(1)} = \beta$, $\beta^{(j)} = \alpha^{(j-1)} S^{(j-1)}$, para $j \in \{2, \dots, M\}$, e $\alpha^{(j)} = \beta^{(j)} D^{(j)}$, para $j \in \{1, \dots, M\}$. Lembre que os instantes de início de intervalos de *switch-over* são também pontos embutidos no tempo.

Definamos agora a variável aleatória $\Gamma_s^{(j)}$ como sendo a duração de um mini-ciclo j , dado que, quando este começou, o sistema estava no estado s . Seja também $\Delta_s^{(j)}$ a duração de um intervalo de *switch-over* j , dado que o estado inicial era s . Então, analogamente ao numerador, o denominador da equação 2.16 pode ser dado por [1]

$$\sum_{s \in S} E[L_s] \beta_s = \sum_{j=1}^M \sum_{s \in S} E[\Gamma_s^{(j)}] \beta_s^{(j)} + \sum_{j=1}^M \sum_{s \in S} E[\Delta_s^{(j)}] \alpha_s^{(j)}.$$

As medidas $E[\Gamma_s^{(j)}]$ e $E[\Delta_s^{(j)}]$ são independentes do subconjunto \mathcal{R} . Como se vê, expressamos $P_{\mathcal{R}}$ agora em função de medidas envolvendo mini-ciclos de serviço e intervalos de *switch-over*, ao invés de ciclos de *polling*, o que torna mais fácil a análise. Portanto, de [1], ficamos com

$$P_{\mathcal{R}} = \frac{\sum_{j=1}^M \sum_{s \in S} E[U_s^{(j)}] \beta_s^{(j)} + \sum_{j=1}^M \sum_{s \in S} E[V_s^{(j)}] \alpha_s^{(j)}}{\sum_{j=1}^M \sum_{s \in S} E[\Gamma_s^{(j)}] \beta_s^{(j)} + \sum_{j=1}^M \sum_{s \in S} E[\Delta_s^{(j)}] \alpha_s^{(j)}}. \quad (2.17)$$

Inicialmente calcularemos o denominador da equação 2.17. Conforme assumimos, os intervalos de *switch-over* são constantes e portanto, independentes do estado em que se encontra o sistema no início do intervalo, ou seja, $E[\Delta_s^{(j)}] = \sigma_j$,

para $j \in \{1, \dots, M\}$, independente de s . Logo, o segundo termo no denominador da equação 2.17 se torna

$$\sum_{j=1}^M \sum_{s \in S} E[\Delta_s^{(j)}] \alpha_s^{(j)} = \sum_{j=1}^M \sigma_j$$

que é o tempo total gasto em períodos de *switch-over* em um ciclo de *polling*.

Para o cálculo de $E[\Gamma_s^{(j)}]$, usaremos o fato de que a duração de um mini-ciclo j depende unicamente do estado inicial da fila j , quando chega o servidor, não dependendo portanto, do estado das demais filas. Portanto, $E[\Gamma_s^{(j)}] = E[\Gamma_{s'}^{(j)}]$, se $q_j = q'_j$, onde $s = \langle q_1, \dots, q_j, \dots, q_M \rangle$ e $s' = \langle q'_1, \dots, q'_j, \dots, q'_M \rangle$. Definamos então probabilidades marginais a partir das probabilidades conjuntas em $\beta^{(j)}$ para refletir este fato. Seja $b^{(j)}$ o vetor de probabilidades marginais obtido do vetor $\beta^{(j)}$, agregando-se os estados $s \in S$ com o mesmo j -ésimo elemento (q_j). Seja também $S_j = \{q; q \geq 0\}$, o espaço de estados da cadeia de Markov randomizada associada a $\mathcal{W}^{(j)}$. Já para modelos com *buffer* limitado, temos $S_j = \{q; 0 \leq q \leq B_j\}$, onde B_j é o tamanho do *buffer* da fila j . Então para $q \in S_j$, seja

$$b_q^{(j)} = \sum_{\substack{s = \langle q_1, \dots, q_j, \dots, q_M \rangle \in S \\ q_j = q}} \beta_s^{(j)}.$$

Com isto, podemos reescrever o primeiro termo do denominador da equação 2.17 como

$$\sum_{j=1}^M \sum_{s \in S} E[\Gamma_s^{(j)}] \beta_s^{(j)} = \sum_{j=1}^M \sum_{q=0}^{\infty} E[\tau_j | q] b_q^{(j)}$$

onde τ_j é a duração de um mini-ciclo j . Para o caso de *buffer* limitado, o somatório interno no lado direito da equação anterior é limitado em $q = B_j$. Para o cálculo de $E[\tau_j | q]$, vamos inicialmente condicioná-lo à ocorrência de n transições da cadeia de Markov randomizada associada a $\mathcal{W}^{(j)}$. Da teoria de processos Poisson temos: se n eventos Poisson ocorrerem em um intervalo de tempo fixo I , então este intervalo será dividido em $n + 1$ sub-intervalos, representando os intervalos de tempo entre eventos. As variáveis aleatórias que representam as durações destes sub-intervalos são cambiáveis, e portanto, o valor esperado da duração de qualquer sub-intervalo é $I/(n + 1)$. Observando que neste caso, o intervalo considerado é a quota de serviço T_j da fila j , temos

$$E[\tau_j | q, n] = \sum_{m=0}^n \frac{T_j}{n+1} (1 - \pi_{abs}^{(j)}(m))$$

e descondicionando no número de transições [1]

$$E[\tau_j | q] = T_j \sum_{n=0}^{\infty} e^{-\Lambda_j T_j} \frac{(\Lambda_j T_j)^n}{n!} \left\{ \frac{\sum_{m=0}^n (1 - \pi_{abs}^{(j)}(m))}{n+1} \right\} \quad (2.18)$$

onde $\pi^{(j)}(m) = \pi^{(j)}(m-1)W^{(j)}$, com $W^{(j)}$ sendo a matriz de transição da cadeia de Markov randomizada associada a $\mathcal{W}^{(j)}$, e $\pi_q^{(j)}(0) = 1$. Na equação acima, $\pi_{abs}^{(j)}(m)$ é a probabilidade da cadeia randomizada estar em um estado absorvente no passo m .

Uma vez que já calculamos a função distribuição $F(t)$ para a duração τ_j de um mini-ciclo j , dado que a fila j tem q clientes no início do mini-ciclo, a equação 2.18 pode ser obtida a partir da fórmula $E[\tau_j | q] = \int_0^{T_j} [1 - F(t | q)] dt$. O desenvolvimento para tal é semelhante ao feito para derivar a equação 2.13. É fácil ver que temos um caso particular quando a fila j está vazia ($q = 0$) quando da chegada do servidor. Neste caso, o servidor parte imediatamente para ir servir a próxima fila e o mini-ciclo j não ocorre, ou seja, $E[\tau_j | q = 0] = 0$, pois $\pi_{abs}^{(j)}(m) = 1$ para todo $m \geq 0$.

O primeiro termo do denominador da equação 2.17, como já vimos, nos dá o valor esperado da duração de um ciclo de *polling* sem contar os períodos de *switch-over*. Para obter tal medida, poderíamos calcular a equação 2.18, em cada mini-ciclo j , para cada valor de q individualmente, e depois fazer um descondicionamento com $b_q^{(j)}$. No entanto, basta calcularmos a equação 2.18 com uma distribuição inicial $b^{(j)}$ em cada mini-ciclo j . Então temos

$$\sum_{q=0}^{\infty} E[\tau_j | q] b_q^{(j)} = T_j \sum_{n=0}^{\infty} e^{-\Lambda_j T_j} \frac{(\Lambda_j T_j)^n}{n!} \left\{ \frac{\sum_{m=0}^n (1 - \pi_{abs}^{(j)}(m))}{n+1} \right\} \quad (2.19)$$

onde $\pi^{(j)}(m) = \pi^{(j)}(m-1)W^{(j)}$, porém agora a distribuição inicial é dada por $\pi^{(j)}(0) = b^{(j)}$. Portanto, de [1], o primeiro termo do denominador da equação 2.17 é dado por

$$\sum_{j=1}^M \sum_{s \in \mathcal{S}} E[\Gamma_s^{(j)}] \beta_s^{(j)} = \sum_{j=1}^M T_j - \sum_{j=1}^M T_j \sum_{n=0}^{\infty} e^{-\Lambda_j T_j} \frac{(\Lambda_j T_j)^n}{n!} \left\{ \frac{\sum_{m=0}^n \pi_{abs}^{(j)}(m)}{n+1} \right\} \quad (2.20)$$

onde $\sum_{j=1}^M T_j$ é a soma das quotas de serviço de todas as filas, ou seja, a máxima duração de um ciclo de *polling*, sem contar os períodos de *switch-over*, e assumindo *time-outs* preemptivos. O termo entre chaves na equação 2.19 pode ser calculado recursivamente a partir da fórmula

$$f^{(j)}(n+1) = \frac{n+1}{n+2} f^{(j)}(n) + \frac{1 - \pi_{abs}^{(j)}(n+1)}{n+2}$$

onde

$$f^{(j)}(n+1) = \frac{\sum_{m=0}^n (1 - \pi_{abs}^{(j)}(m))}{n+1}.$$

Os termos envolvidos no cálculo do numerador da equação 2.17 são dependentes do sub-conjunto \mathcal{R} , cuja probabilidade estamos interessados em calcular.

Agora calcularemos o numerador da equação 2.17. Neste trabalho assumiremos, por simplicidade, que \mathcal{R} é o sub-conjunto de estados de \mathcal{X} onde uma dada fila i tem um certo tamanho l_i , ou seja, l_i clientes nela. No entanto, para outros sub-conjuntos \mathcal{R} , a expressão para a probabilidade pode ser derivada sem maiores problemas, à luz do caso particular derivado aqui. Inicialmente, calcularemos o termo $E[V_s^{(j)}]$ que representa, como já vimos, o valor esperado do tempo gasto pelo processo \mathcal{X} , ou seja, pelo sistema, em \mathcal{R} durante um intervalo de *switch-over* j , dado que o estado no início do intervalo era s . Como a duração σ_j do intervalo de *switch-over* j é independente dos estados das filas no início do intervalo, temos

que, o tempo gasto pela fila i com l_i clientes durante o intervalo de *switch-over* j depende apenas do estado da fila i no início do intervalo e do número de chegadas que ocorreram nesta mesma fila, com taxa λ_i . Seja portanto q_i , o estado da fila i no início do intervalo de *switch-over* j . Se $q_i > l_i$, então nunca haverá l_i clientes na fila i durante o intervalo de *switch-over*, pois partidas da fila não podem ocorrer. Neste caso $E[V_s^{(j)}] = 0$. Para o caso onde $q_i \leq l_i$, é necessário que ocorram pelo menos $l_i - q_i$ chegadas à fila i , durante o intervalo de *switch-over*, para que em algum instante haja l_i clientes na fila i . Vamos usar também o fato de que n chegadas Poisson, durante um intervalo de *switch-over* j de duração σ_j , gera neste intervalo $n+1$ sub-intervalos com duração média $\sigma_j/(n+1)$. Portanto, se ocorrerem $n \geq l_i - q_i$ chegadas, então o tempo gasto pela fila i , com l_i clientes durante o intervalo de *switch-over* j , é dado pela duração de um sub-intervalo, ou seja, $\sigma_j/(n+1)$. Descondicionando no número de chegadas, temos de [1]

$$E[V_s^{(j)}] = \sum_{n=l_i-q_i}^{\infty} e^{-\lambda_i \sigma_j} \frac{(\lambda_i \sigma_j)^n}{n!} \left(\frac{\sigma_j}{n+1} \right). \quad (2.21)$$

Seja $a^{(i,j)}$, similarmente a $b^{(j)}$, o vetor de probabilidades marginais obtido do vetor $\alpha^{(j)}$, agregando-se os estados $s \in \alpha^{(j)}$ com o mesmo i -ésimo elemento (q_i). Da mesma forma, seja $S_j = \{q; q \geq 0\}$, o espaço de estados da cadeia de Markov randomizada associada a \mathcal{W}^j . Para modelos com *buffer* limitado, temos $S_j = \{q; 0 \leq q \leq B_j\}$, onde B_j é o tamanho do *buffer* da fila j . Então, para $q \in S_j$, seja

$$a_q^{(i,j)} = \sum_{\substack{s = \langle q_1, \dots, q_i, \dots, q_M \rangle \in S \\ q_i = q}} \alpha_s^{(j)}.$$

Portanto, usando $a^{(i,j)}$ e descondicionando no estado inicial s , bem como reconhecendo a distribuição de Erlang com $k+1$ estágios dada por $E_{k,\lambda}(t) = 1 - \sum_{n=0}^k e^{-\lambda t} (\lambda t)^n / n!$, obtemos o segundo termo do numerador da equação 2.17 como sendo

$$\sum_{j=1}^M \sum_{s \in S} E[V_s^{(j)}] \alpha_s^{(j)} = \sum_{j=1}^M \frac{1}{\lambda_i} \sum_{q_i=0}^{l_i} E_{l_i-q_i, \lambda_i}(\sigma_j) a_{q_i}^{(i,j)}. \quad (2.22)$$

Para modelos de *buffer* finito, temos algumas modificações, pois a expressão 2.22 não resolve dois casos particulares. No primeiro caso, se $l_i = q_i = B_i$, então $E[V_s^{(j)}] = \sigma_j$, ou seja, como não pode haver partidas da fila i , o tempo gasto por ela com seu *buffer* cheio, dado que ela já estava cheia no início do intervalo de *switch-over* j , é igual a própria duração deste intervalo. No segundo, quando $l_i = B_i \neq q_i$, estamos querendo calcular o tempo gasto pela fila i com seu *buffer* cheio, dado que ela não estava cheia no início do intervalo de *switch-over* j . Neste caso, uma vez a fila ficando cheia, as subseqüentes chegadas de clientes não afetarão o seu estado. Calcularemos $E[V_s^{(j)}]$ de uma forma complementar, ou seja, o tempo gasto pela fila i com seu *buffer* cheio é igual à duração do intervalo de *switch-over* j menos o tempo gasto com o *buffer* não cheio, este podendo ser calculado diretamente

pela equação 2.21. Formalizando, temos

$$E[V_s^{(j)}] |_{l_i=B_i} = E[\Delta_s^{(j)}] - \sum_{l=0}^{B_i-1} E[V_s^{(j)}] |_{l_i=l}.$$

No modelo estudado aqui, onde os intervalos de *switch-over* são constantes, ficamos com

$$E[V_s^{(j)}] |_{l_i=B_i} = \sigma_j - \sum_{l=0}^{B_i-1} E[V_s^{(j)}] |_{l_i=l}. \quad (2.23)$$

Os demais casos podem ser resolvidos pela aplicação direta da equação 2.21.

Finalmente, calcularemos o primeiro termo do numerador da equação 2.17, que representa o valor esperado do tempo gasto pelo sistema em \mathcal{R} durante um mini-ciclo j , ou ainda, o tempo gasto pela fila i com l_i clientes no seu *buffer* durante um mini-ciclo j . Por conveniência, vamos dividir o tratamento em duas partes: quando $j = i$ ou $j \neq i$. Como no cálculo da expressão 2.22, se $l_i < q_i$, então $E[U_s^{(j)}] = 0$, pois só pode haver chegadas à fila i . Vamos começar com o caso onde $j = i$. Aqui, estamos interessados em calcular o valor esperado do tempo gasto pela fila i com l_i clientes, durante um mini-ciclo i , ou seja, durante o mini-ciclo de serviço da própria fila i . Neste caso, podemos estudar o comportamento da fila i isoladamente das demais filas. Seja q_i o estado (número de clientes) da fila i no instante de início do mini-ciclo i . Considerando casos particulares, note que se a fila i estiver vazia no início do seu mini-ciclo i ($q_i = 0$), este não acontecerá, com o servidor passando imediatamente à próxima fila. Neste caso portanto, $E[U^{(i)} | q_i = 0] = 0$. Convém notar também que, se $l_i = 0$, então $E[U_s^{(i)}] = 0$, pois o servidor abandona imediatamente uma fila quando esta se torna vazia. Para prosseguir o desenvolvimento, vamos considerar portanto, que $l_i > 0$. Empregando a técnica de randomização, com a cadeia de Markov randomizada correspondente a $\mathcal{W}^{(i)}$, o tempo gasto pela fila i com l_i clientes pode ser calculado através de uma análise transiente durante um intervalo de tempo de duração T_i . Como já vimos, isto é possível porque, embora o mini-ciclo i possa ter uma duração menor do que T_i , o estado 0 da cadeia $\mathcal{W}^{(i)}$ é absorvente. Observe que, se ocorrerem n transições neste intervalo, então pode haver vários sub-intervalos durante os quais existem l_i clientes na fila i , com cada sub-intervalo durando, como já vimos, $T_i/(n+1)$. Formalizando, a seguinte expressão, condicionada em q_i , o estado no início do mini-ciclo, pode ser obtida [1]:

$$E[U^{(i)} | q_i] = \sum_{n=0}^{\infty} e^{-\Lambda_i T_i} \frac{(\Lambda_i T_i)^n}{n!} \cdot \left[\frac{T_i}{n+1} \cdot \sum_{m=0}^n \pi_{l_i}^{(i)}(m) \right] \quad (2.24)$$

onde $\pi(m) = \pi(m-1)W^{(i)}$ e $\pi_{q_i}(0) = 1$. Se, como na equação 2.19, empregarmos o vetor $b^{(i)}$ como vetor distribuição inicial da cadeia uniformizada, ficamos com

$$\sum_{s \in \mathcal{S}} E[U_s^{(i)}] \beta_s^{(i)} = T_i \sum_{n=0}^{\infty} e^{-\Lambda_i T_i} \frac{(\Lambda_i T_i)^n}{n!} \cdot \left[\frac{\sum_{m=0}^n \pi_{l_i}^{(i)}(m)}{n+1} \right] \quad (2.25)$$

onde $\pi^{(i)}(m) = \pi^{(i)}(m-1)W^{(i)}$ e $\pi^{(i)}(0) = b^{(i)}$. Para modelos de *buffer* finito, o tratamento é idêntico.

Para o caso onde $j \neq i$, temos chegadas à fila i durante um mini-ciclo de outra fila j . Seja q_i o estado (número de clientes) da fila i no início do mini-ciclo j . Seja também q_j o número de clientes na fila j no início do mini-ciclo j . O tempo gasto pela fila i com l_i clientes durante um mini-ciclo j ($E[U^{(j)}]$) depende de q_i , mas também de q_j , pois este último controla a duração do mini-ciclo. Como caso particular, note que se a fila j estiver vazia no início do seu mini-ciclo j ($q_j = 0$), este não acontecerá, e o servidor passará imediatamente à próxima fila. Portanto neste caso $E[U_s^{(j)}] = 0$. Inicialmente, calcularemos o valor esperado do tempo gasto pela fila i com l_i clientes durante um mini-ciclo j . Condicionando o valor esperado em q_i, q_j , obtemos de [1]

$$\begin{aligned}
E[U^{(j)} | q_i, q_j] &= \frac{1}{\lambda_i} \sum_{n=l_i-q_i+1}^{\infty} \left\{ \sum_{k=n+1}^{\infty} e^{-(\lambda_i+\Lambda_j)T_j} \frac{[(\lambda_i+\Lambda_j)T_j]^k}{k!} \right. \\
&\times \sum_{m=n+1}^k \left(\frac{\lambda_i}{\lambda_i+\Lambda_j} \right)^n \left(\frac{\Lambda_j}{\lambda_i+\Lambda_j} \right)^{m-n} \binom{m-1}{n} P_{abs}(m-n) \\
&+ \sum_{k=n}^{\infty} e^{-(\lambda_i+\Lambda_j)T_j} \frac{[(\lambda_i+\Lambda_j)T_j]^k}{k!} \\
&\times \left. \left(\frac{\lambda_i}{\lambda_i+\Lambda_j} \right)^n \left(\frac{\Lambda_j}{\lambda_i+\Lambda_j} \right)^{k-n} \binom{k}{n} [1 - \pi_{abs}(k-n)] \right\} \quad (2.26)
\end{aligned}$$

onde $\Lambda_j = \lambda_j + \mu_j$, $\pi(m) = \pi(m-1)W^{(j)}$ e $\pi_{q_j}(0) = 1$. Por fim, descondicionamos em q_i e q_j . Como no cálculo da equação 2.20, não é preciso realizar computações com a cadeia de Markov randomizada $W^{(j)}$ na equação 2.26 para cada par (q_i, q_j) . Definamos então, $b_q^{(i,j)}$ como sendo um vetor de probabilidades obtido de $\beta^{(j)}$ pelo agregamento de estados com o mesmo j -ésimo elemento e com o i -ésimo elemento igual a q . Por exemplo, se o sistema tem 3 filas com tamanho de *buffer* 2, o vetor $\beta^{(j)}$ é dado por $\{P(0,0,0), P(0,0,1), \dots, P(2,2,2)\}$, onde $P(q_1, q_2, q_3)$ significa a probabilidade de, no instante de início do mini-ciclo j , o sistema estar no estado (q_1, q_2, q_3) , ou seja, estar com q_1, q_2 e q_3 clientes respectivamente nas filas 1, 2 e 3. Com $i = 2$ e $j = 1$, é fácil ver que $b_1^{(2,1)} = \{P(0,1,0) + P(0,1,1) + P(0,1,2), P(1,1,0) + P(1,1,1) + P(1,1,2), P(2,1,0) + P(2,1,1) + P(2,1,2)\}$. Neste caso $b_1^{(2,1)}$ tem 3 elementos indicando as probabilidades marginais para q_j (0, 1 ou 2) e $q_i = 1$. De volta ao desenvolvimento, descondicionando a equação 2.26 em q_i e q_j , e utilizando o vetor $b_q^{(i,j)}$, chegamos a [1]

$$\begin{aligned}
\sum_{s \in S} E[U_s^{(j)}] \beta_s^{(j)} &= \frac{1}{\lambda_i} \sum_{q_i=0}^{l_i} \sum_{n=l_i-q_i+1}^{\infty} \left\{ \sum_{k=n+1}^{\infty} e^{-(\lambda_i+\Lambda_j)T_j} \frac{[(\lambda_i+\Lambda_j)T_j]^k}{k!} \right. \\
&\times \sum_{m=n+1}^k \left(\frac{\lambda_i}{\lambda_i+\Lambda_j} \right)^n \left(\frac{\Lambda_j}{\lambda_i+\Lambda_j} \right)^{m-n} \binom{m-1}{n} P_{abs}^{(i,j)}(m-n) \\
&+ \sum_{k=n}^{\infty} e^{-(\lambda_i+\Lambda_j)T_j} \frac{[(\lambda_i+\Lambda_j)T_j]^k}{k!} \\
&\times \left. \left(\frac{\lambda_i}{\lambda_i+\Lambda_j} \right)^n \left(\frac{\Lambda_j}{\lambda_i+\Lambda_j} \right)^{k-n} \binom{k}{n} [1 - \pi_{abs}^{(i,j)}(k-n)] \right\} \quad (2.27)
\end{aligned}$$

onde $\Lambda_j = \lambda_j + \mu_j$, $\pi^{(i,j)}(m) = \pi^{(i,j)}(m-1)W^{(j)}$ e $\pi^{(i,j)}(0) = b_{q_i}^{(i,j)}$. Salvo os casos

particulares, o primeiro termo do numerador da equação 2.17 é dado portanto pelas equações 2.25 e 2.27.

Para modelos de *buffer* finito, algumas modificações devem ser feitas, pois a expressão 2.27 não resolve dois casos particulares: No primeiro caso, se $l_i = q_i = B_i$, então $E[U_s^{(j)}]$ será igual ao valor esperado do mini-ciclo j , condicionado no estado inicial q_j da fila j . Isto é, como não pode haver partidas da fila i , o tempo gasto por ela com seu *buffer* cheio durante um mini-ciclo j , dado que ela já estava cheia no início deste mini-ciclo, é igual a própria duração do mini-ciclo j . No segundo caso, quando $l_i = B_i \neq q_i$, estamos interessados em calcular o tempo gasto pela fila i com seu *buffer* cheio durante um mini-ciclo j , dado que ela não estava cheia no início deste mini-ciclo j . Neste caso, uma vez a fila ficando cheia, as subsequentes chegadas de clientes não afetarão o seu estado. Como no cálculo de $E[V_s^{(j)}]$, calcularemos portanto $E[U_s^{(j)}]$ de uma forma complementar, ou seja, o tempo gasto pela fila i com seu *buffer* cheio é igual à diferença entre a duração esperada do mini-ciclo j , condicionado no estado inicial q_j da fila j , e o tempo gasto com o *buffer* não cheio. Este tempo, por sua vez, pode ser calculado diretamente pela equação 2.26. Formalizando, escrevemos assim:

$$E[U^{(j)} | q_i, q_j] |_{l_i=B_i} = E[\tau_j | q_j] - \sum_{l=0}^{B_i-1} E[U^{(j)} | q_i, q_j] |_{l_i=l} \quad (2.28)$$

Os demais casos podem ser resolvidos pela aplicação direta da equação 2.26.

Finalmente, uma vez calculadas as probabilidades de estado com média no intervalo de tempo $[0, \infty)$, ou seja, em estado estacionário, os tamanhos médios das filas durante todo o tempo podem ser obtidos facilmente. Também, usando o resultado de Little, os tempos médios de espera (retardos médios), por pacote, para cada fila podem ser determinados. Para os modelos de *buffer* finito, probabilidades de bloqueio podem ser avaliadas diretamente das probabilidades em equilíbrio, explorando-se a suposição de chegadas Poisson (PASTA).

Capítulo 3

Aspectos Computacionais

Neste capítulo, discutiremos vários aspectos relacionados à resolução, em computador, do método de modelagem e análise proposto em [1]. Inicialmente, estudaremos as situações onde podem ocorrer problemas de *underflow/overflow*, quando do cálculo das equações apresentadas no capítulo 2. Depois, apresentaremos um método para determinar eficientemente as matrizes $S^{(j)}$ e $D^{(j)}$. A eficiência do método objetiva reduzir os custos de processamento e de armazenamento envolvidos com as matrizes. O método, aplicável em modelos com *buffer* ilimitado ou limitado, é baseado na política de serviço exaustivo. Contudo, para as outras políticas, o raciocínio é análogo. Finalmente mostraremos detalhes da ferramenta desenvolvida neste trabalho, incluindo como ela implementa o método em questão, bem como ela computa os vetores de distribuição de probabilidade de estado β , em estado estacionário, além de diversas medidas de desempenho de interesse estudadas neste trabalho.

O espaço de estados do modelo para um sistema com M filas, onde o tamanho do *buffer* de cada uma delas é igual a B , contém $(B + 1)^M$ estados. Portanto, a matriz $S^{(j)}$ ou $D^{(j)}$, possui $[(B + 1)^M]^2$ entradas. Lembrando que, para cada fila estão associados um mini-ciclo de serviço e um intervalo de *switch-over*, temos um gasto total com o armazenamento das matrizes da ordem de $2M[(B+1)^M]^2$ elementos. O espaço de armazenamento cresce, portanto, exponencialmente com o tamanho do sistema a ser modelado.

Um método para determinarmos $S^{(j)}$ e $D^{(j)}$ seria calcular todos os seus elementos individualmente e armazená-los. Porém, para modelos de sistemas reais, estas matrizes são bastante grandes e este método se torna ineficiente em termos de armazenamento, pois requer que a matriz inteira seja guardada. Felizmente, não é preciso guardar todos os elementos das matrizes. Inclusive, muitos deles são zero, indicando transições impossíveis de ocorrer. O método para computação das matrizes $S^{(j)}$ e $D^{(j)}$ apresentados a seguir é eficiente em termos de armazenamento, pois não guarda informação redundante, bem como em termos de processamento, pois não emprega demasiado esforço computacional.

3.1 Prevenindo Situações de *Underflow/Overflow*

A computação das equações 2.11 e 2.13, bem como as diversas medidas de desempenho (e. g. equação 2.21), envolve o cálculo de

$$\eta = \sum_{n=n'}^{\infty} e^{-Rt} \frac{(Rt)^n}{n!}$$

onde R é a taxa Poisson do processo randomizado composto, e n' é o valor inicial de n a ser considerado. Seja $\eta(n) = e^{-Rt} (Rt)^n / n!$, ou seja, $\eta(n)$ é o $(n + 1 - n')$ -ésimo termo em η . Como a expressão η acima envolve uma série infinita de termos de uma distribuição Poisson, e não é possível considerar todos, temos que trabalhar com uma certa tolerância de erro ϵ nos cálculos. Mais adiante, veremos como determinar N , como sendo o maior n , tal que $\eta(n)$ ainda seja considerado na computação das fórmulas, de acordo com a precisão exigida. Se η é avaliado diretamente; tal qual é apresentado acima, é bastante provável que nos deparemos com problemas de *underflow/overflow*, mesmo para valores moderados de Rt . Dependendo do valor de Rt , alguns termos $\eta(n)$ iniciais da distribuição, para valores de n pequenos, podem ser muito pequenos e causarem portanto, situações de *underflow* em cálculos em computador. Portanto, iremos encontrar o menor valor de n , tal que $\eta(n)$ é grande o bastante para não causar *underflow*, e também que a soma dos valores de $\eta(m)$, para $0 \leq m \leq n$, seja desprezível em relação ao erro ϵ exigido. Para evitar tais problemas, empregamos um algoritmo bastante simples [11], e que será discutido a seguir.

Com base no fato de que a distribuição Poisson pode ser aproximada por uma distribuição normal, para grandes valores de Rt , podemos escolher um valor de n igual a N_{min} , tal que a cauda inferior da distribuição tenha um valor bastante menor que ϵ , mas é ainda maior do que o limite de *underflow* existente. Por exemplo, usando

$$N_{min} = \max(0, Rt - 10\sqrt{Rt})$$

o valor da cauda inferior da distribuição está na ordem de 10^{-30} , o que é desprezível em comparação com valores razoáveis de ϵ . Além disto, $\eta(N_{min})$ ainda é maior que o menor número possível (limite para *underflow*) representado em FORTRAN77, que é da ordem de 10^{-70} . Temos portanto que $\eta(N_{min})$ é computável. Já qualquer termo $\eta(n)$, para $0 \leq n < N_{min}$, é muito pequeno, e pode causar *underflow*. Uma vez identificado N_{min} , calculamos a seguir o termo $\eta(N_{min})$. Embora este termo seja computável, o seu cálculo, de uma forma direta, também pode gerar uma situação de *underflow*. Para pequenos valores de Rt , temos $N_{min} = 0$, e portanto $\eta(N_{min}) = e^{-Rt}$. No caso de $N_{min} > 0$, usamos o seguinte algoritmo: inicialmente, definimos $E = Rte^{-Rt/(N_{min}+1)}$. Note que para grandes valores de Rt , temos $N_{min} \approx Rt$, e então $E \approx Rt/e$. Agora, usando isto, temos

$$\eta(N_{min}) = e^{-Rt/(N_{min}+1)} \prod_{i=1}^{N_{min}} \frac{E}{i}.$$

A idéia do algoritmo é calcular o produto acima, de tal forma que o resultado intermediário fique tão perto de 1 quanto possível. Para tal, nós escolhemos i , tal

que $E/i \approx 1$. Multiplicando-se então este termo por E/j , para $j < i$ ou $j > i$, há uma tendência de o resultado intermediário aumentar ou diminuir, respectivamente. Nós usamos este fato para escolher o valor de j apropriado, tal que E/j seja o próximo termo a ser multiplicado pelo resultado intermediário. Como $\eta(N_{min})$ é maior que o valor de *underflow*, o resultado final certamente é computável. O algoritmo propriamente dito é apresentado no apêndice A. Como acabamos de ver, nas equações que envolvam o cálculo de η , o primeiro termo que pode ser computado é $\eta(N_{min})$. Porém, quando em uma equação, o valor inicial n' de n for menor que N_{min} , devemos desconsiderá-lo e realizar a computação considerando somente os termos $\eta(n)$ para valores de n a partir de $n = N_{min}$. Por outro lado, se $n' > N_{min}$, começamos a partir do valor de n igual ao valor inicial n' na equação. Por último, se o valor inicial n' for igual ao próprio N_{min} , nada precisa ser alterado na computação da equação. Resumindo, se n' é o valor inicial de n em uma dada fórmula, o primeiro valor efetivamente considerado na computação dessa fórmula será n^+ tal que

$$n^+ = \max(N_{min}, n').$$

Os termos restantes $\eta(n)$, para $N_{min} < n \leq N$, são calculados pela recursão

$$\eta(n) = \eta(n-1) \cdot \frac{Rt}{n}.$$

Também, nas diversas equações do modelo, nos deparamos com séries infinitas de termos envolvendo as distribuições Poisson, como em η , e multinomial. Além do problema das séries serem infinitas, nos deparamos, como no caso do N_{min} , com problemas de *underflow/overflow*. No entanto, desta vez, o problema surge no cômputo de termos $\eta(n)$ muito pequenos, para valores grandes de n . Tomemos então, por exemplo, as equações 2.11 e 2.13, usadas para o cálculo de $d_{s,s'}^{(j)}$. Como não é possível computar todos os termos, temos que trabalhar, como já vimos, com um certo erro nos cálculos. A partir de uma dada tolerância de erro ϵ , podemos calcular antecipadamente o número mínimo de termos da distribuição Poisson a serem considerados, para que a probabilidade $d_{s,s'}^{(j)}$, calculada tenha no máximo um erro de precisão ϵ em relação ao valor exato. Obviamente, o valor exato seria obtido se todos os termos Poisson fossem computados. Seja $P(n) = e^{-\gamma T_j} (\gamma T_j)^n / n!$ o n -ésimo termo da distribuição Poisson, e N , tal que $P(N)$ é o último termo, a partir de $n = N_{min}$, considerado nos cálculos. Aqui nós consideramos a partir de N_{min} , pois $\sum_{n=0}^{N_{min}-1} P(n)$ é desprezível em relação a ϵ . O erro, determinado pelo cálculo de $d_{s,s'}^{(j)}$, considerando os termos Poisson $P(n)$ com $n \in [N+1, \infty)$, não deve ultrapassar a tolerância de erro ϵ dada. A expressão $\sum_{n=N+1}^{\infty} P(n)$ é um limitante superior para o erro. Portanto

$$\sum_{n=N+1}^{\infty} P(n) \leq \epsilon.$$

Usando probabilidade total, temos

$$\sum_{n=N_{min}}^N P(n) \geq 1 - \epsilon. \quad (3.1)$$

Obviamente, quanto menor for ϵ , mais próxima de 1 é a soma na equação 3.1, e portanto mais precisa é a probabilidade computada.

Então, com uma dada tolerância de erro ϵ , determinamos o número de termos a serem considerados no cômputo das equações. Para a equação 2.11, considerando os termos Poisson $P(n)$ com $n \in [\max(N_{min}, k_j), N]$, temos

$$d_{s,s'}^{(j)} = \sum_{n=\max(N_{min}, k_j)}^N P(n) \prod_{i \neq j} \left(\frac{\lambda_i}{\gamma}\right)^{k_i} \cdot \left(\frac{\Lambda_j}{\gamma}\right)^{(n-k_j)} \cdot \frac{n!}{(n-k_j)! \prod_{i \neq j} k_i!} \pi_{q'_j}^{(j)}(n-k_j)$$

ou, fazendo $\alpha_i = \lambda_i/\gamma$ e $\beta = \Lambda_j/\gamma$,

$$d_{s,s'}^{(j)} = \sum_{n=\max(N_{min}, k_j)}^N P(n) \times \prod_{i \neq j} \alpha_i^{k_i} \beta^{n-k_j} \cdot \frac{n!}{(n-k_j)! \prod_{i \neq j} k_i!} \pi_{q'_j}^{(j)}(n-k_j) \quad (3.2)$$

e, para a equação 2.13, com $n \in [\max(N_{min}, k_j + 1), N]$, temos

$$d_{s,s'}^{(j)} = \sum_{n=\max(N_{min}, k_j+1)}^N P(n) \times \sum_{m=k_j+1}^n \prod_{i \neq j} \left(\frac{\lambda_i}{\gamma}\right)^{k_i} \cdot \left(\frac{\Lambda_j}{\gamma}\right)^{m-k_j} \frac{(m-1)!}{(m-k_j-1)! \prod_{i \neq j} k_i!} P_{abs}^{(j)}(m-k_j)$$

ou, mais uma vez com $\alpha_i = \lambda_i/\gamma$ e $\beta = \Lambda_j/\gamma$,

$$d_{s,s'}^{(j)} = \sum_{n=\max(N_{min}, k_j+1)}^N P(n) \times \sum_{m=k_j+1}^n \prod_{i \neq j} \alpha_i^{k_i} \beta^{m-k_j} \frac{(m-1)!}{(m-k_j-1)! \prod_{i \neq j} k_i!} P_{abs}^{(j)}(m-k_j). \quad (3.3)$$

Lembre que, nas equações 3.2 e 3.3 acima, $P(n) = e^{-\gamma T_j} (\gamma T_j)^n / n!$.

3.2 Determinando $S^{(j)}$

A matriz $S^{(j)}$, como já vimos, é determinada pelo cálculo de seus elementos $S_{s,s'}^{(j)}$, através da equação 2.3, adaptada ao tipo de modelo estudado. Ao invés de computarmos cada elemento individualmente, e armazenarmos a matriz inteira, basta computarmos e guardarmos todas as probabilidades Poisson de possíveis chegadas efetivamente armazenadas nas filas durante os períodos de *switch-over*. Para modelos com *buffer* infinito, o número máximo de possíveis chegadas, indicado por N , é determinado pela distribuição Poisson a partir do erro tolerado nos cálculos. Discutiremos mais sobre isto ainda neste capítulo. Em modelos com *buffer* finito e igual a B , a estrutura de dados armazenada é um vetor tridimensional possuindo $B(M)^2$ elementos, o que é bem melhor do que guardar todas as matrizes $S^{(j)}$ inteiras, com $M[(B+1)^M]^2$ elementos no total. O primeiro índice deste vetor indica o número de

chegadas, o segundo índice indica a fila a qual estão chegando os clientes, e o terceiro índice indica de qual fila é o período de *switch-over*. Quando, durante o cálculo de β , precisarmos de uma probabilidade de transição $S_{s,s'}^{(j)}$, temos apenas que resolver a equação 2.3, adaptada ao tipo de modelo, utilizando as probabilidades Poisson já calculadas e armazenadas anteriormente.

3.3 Determinando $D^{(j)}$ para Modelos com *Buffer* Ilimitado

Determinamos $D^{(j)}$ através do cálculo das suas probabilidades de transição $d_{s,s'}^{(j)}$. Veremos então agora, como computar eficientemente as equações 2.11 e 2.13, válidas para modelos com *buffer* ilimitado e política de serviço exaustivo.

Uma entrada $d_{s,s'}^{(j)}$ na matriz $D^{(j)}$ representa a probabilidade de transição do estado s para o estado s' em um mini-ciclo, quando a fila j está sendo servida. Isto é, a probabilidade de o sistema estar no estado s' no final do mini-ciclo j , dado que no início do mini-ciclo, quando o servidor chega à fila j , o estado do sistema era s . Sem perda de generalidade, nós assumimos, daqui para frente, que $j = 1$. Para simplificar a notação, vamos supor que $N_{min} = 0$ no desenvolvimento a seguir, e na próxima seção. Porém, deve ficar claro que, na implementação de uma fórmula em computador, o primeiro valor de n considerado, deve ser o máximo entre N_{min} , e o valor inicial para n , indicado no somatório da fórmula. Logo, a modificação para $N_{min} \neq 0$ é trivial. Para computar eficientemente a equação 3.2, que é a equação 2.11 truncada, definimos então

$$\Upsilon[k_2, \dots, k_M; v] = \sum_{n=k_1}^v P(n) \Omega[k_2, \dots, k_M; n] \pi_{q_1}^{(1)}(n - k_1) \quad (3.4)$$

onde $v \geq k_1 = \sum_{i=2}^M k_i$, e $P(n) = e^{-\gamma T_1} (\gamma T_1)^n / n!$ é o n -ésimo termo da distribuição Poisson. Quando $v = N$, temos portanto a própria equação 3.2, com $d_{s,s'}^{(1)} = \Upsilon[k_2, \dots, k_M; N]$, onde $s' = (j_1, j_2, \dots, j_M)$, $s = (i_1, i_2, \dots, i_M)$, e $k_c = j_c - i_c$, para $c = 2, \dots, M$. Aqui, no caso de *buffer* ilimitado, temos

$$\Omega[k_2, \dots, k_M; n] = \alpha_2^{k_2} \dots \alpha_M^{k_M} \beta^{n-k_1} \binom{n}{k_2 \dots k_M} \quad (3.5)$$

onde o coeficiente multinomial é dado por

$$\binom{n}{k_2 \dots k_M} = \frac{n!}{k_2! \dots k_M! (n - k_1)!}$$

com $k_2 + \dots + k_M \leq n$. Então, $\Omega[k_2 + \dots + k_M; n]$ é a probabilidade de ocorrerem conjuntamente k_i eventos do tipo i na fila i com probabilidade $\alpha_i = \lambda_i / \gamma$ ($i = 2, \dots, M$), dado que o número total de eventos é igual a n , e $\gamma = \sum_{i \neq j} \lambda_i + \Lambda_j$ é a taxa global. Lembre que $\beta = \Lambda_j / \gamma$ é a probabilidade de ocorrer uma transição (chegada ou partida de cliente) na própria cadeia $\mathcal{W}^{(j)}$, com $\Lambda_j = \lambda_j + \mu_j$ para

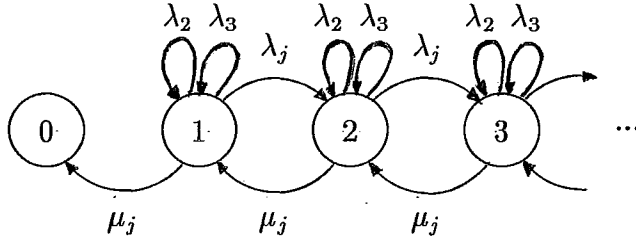


Figura 3.1: Cadeia de Markov Modificada

serviço exaustivo. Podemos calcular esta probabilidade de uma forma recursiva, através da evolução da cadeia de Markov subjacente, para o mini-ciclo j , um pouco modificada [1]. Para tal, nós adicionamos para cada estado da cadeia, com exceção do absorvente, transições retroativas representando chegadas de clientes às outras filas i ($i = 2, \dots, M$), que não estão sendo servidas. Observe que tais transições não alteram a cadeia no que tange ao seu comportamento estocástico. Um exemplo desta cadeia para $M = 3$ é ilustrado na figura 3.1. Então, para $n \geq k_1$, obtemos

$$\begin{aligned} \Omega[k_2, \dots, k_M; n] &= \sum_{i=2}^M \alpha_i \Omega[k_2, \dots, k_i - 1, \dots, k_M; n - 1] \mathcal{I}(k_i \geq 1) \\ &+ \beta \Omega[k_2, \dots, k_M; n - 1] \mathcal{I}(n - 1 \geq k_1) \end{aligned} \quad (3.6)$$

onde $\mathcal{I}(E)$ é uma função que indica se E se verifica (é verdade) ou não, isto é

$$\mathcal{I}(E) = \begin{cases} 1 & \text{se } E \text{ se verifica} \\ 0 & \text{se } E \text{ não se verifica.} \end{cases}$$

O segundo termo da equação 3.6 considera as transições na cadeia relativas à própria fila j sendo servida, ou seja, chegadas ou partidas de clientes da fila j . Já o primeiro termo diz respeito às transições relativas às chegadas às demais filas. A base da equação recursiva 3.6 é dada por $\Omega[0, \dots, 0; 0] = 1$, o que é trivial. Finalmente temos, para $N \geq v \geq k_1$,

$$\begin{aligned} \Upsilon[k_2, \dots, k_M; v] &= \Upsilon[k_2, \dots, k_M; v - 1] \mathcal{I}(v - 1 \geq k_1) \\ &+ P(v) \Omega[k_2, \dots, k_M; v] \pi_{q_1}^{(1)}(v - k_1) \end{aligned} \quad (3.7)$$

onde $P(v)$ e $\pi_{q_j}^{(1)}(v - k_1)$ são calculados recursivamente das suas respectivas expressões para $v - 1$.

O maior esforço computacional aqui é no cálculo da probabilidade $\pi_{q_j}^{(1)}(v - k_1)$. No entanto, este esforço não é tão grande, pois $\pi_{q_j}^{(1)}$ pode ser obtida a partir de uma simples multiplicação de um vetor por uma matriz, onde a matriz representa a cadeia de Markov randomizada correspondente a $\mathcal{W}^{(1)}$, que é pequena. Além do mais, todos os termos $\Omega[k_2, \dots, k_M; n]$, para $i \neq 1$ e $n = 0, \dots, N$, podem ser calculados em paralelo sem grandes custos computacionais extras. Observe que as recursões 3.6 e 3.7 são numericamente estáveis, uma vez que envolvem apenas operações de adição e multiplicação com as probabilidades.

De maneira similar, para computar eficientemente a equação 3.3, que é a mesma equação 2.13, só que truncada, definimos

$$\Theta[k_2, \dots, k_M; v] = \sum_{n=k_1+1}^v P(n) \Gamma[k_2, \dots, k_M; n] \quad (3.8)$$

onde $v \geq k_1 + 1$. Também aqui, quando $v = N$, temos portanto a própria equação 3.3, com $d_{s,s'}^{(1)} = \Theta[k_2, \dots, k_M; N]$, onde $s' = (j_1, j_2, \dots, j_M)$, $s = (i_1, i_2, \dots, i_M)$, e $k_c = j_c - i_c$, para $c = 2, \dots, M$. Aqui, no caso de *buffer* ilimitado, temos

$$\Gamma[k_2, \dots, k_M; n] = \sum_{m=k_1+1}^n \alpha_2^{k_2} \dots \alpha_M^{k_M} \beta^{m-k_1} \binom{m-1}{k_2 \dots k_M} P_{\text{abs}}^{(1)}(m-k_1) \quad (3.9)$$

onde $n \geq k_1 + 1$ e $k_1 = \sum_{i=2}^M k_i$. Lembrando, $\Gamma[k_2, \dots, k_M; n]$ é a probabilidade de k_i eventos do tipo i ($i = 2, \dots, M$) ocorrerem, e de haver absorção (fila 1 ficar vazia) na m -ésima transição da cadeia, dado que o número total de eventos é n . Podemos obter Γ recursivamente com [1]

$$\begin{aligned} \Gamma[k_2, \dots, k_M; n] &= \Gamma[k_2, \dots, k_M; n-1] \mathcal{I}(n-1 \geq k_1+1) \\ &+ \beta \Omega[k_2, \dots, k_M; n-1] P_{\text{abs}}^{(1)}(n-k_1) \end{aligned} \quad (3.10)$$

onde $n \geq k_1 + 1$. Aqui $P_0(n-k_1) = \pi_0(n-k_1) - \pi_0(n-k_1-1)$ é facilmente computado recursivamente. Finalmente temos, para $N \geq v \geq k_1 + 1$,

$$\begin{aligned} \Theta[k_2, \dots, k_M; v] &= \Theta[k_2, \dots, k_M; v-1] \mathcal{I}(v-1 \geq k_1+1) \\ &+ P(v) \Gamma[k_2, \dots, k_M; v]. \end{aligned} \quad (3.11)$$

3.4 Determinando $D^{(j)}$ para Modelos com *Buffer* Limitado

Indicaremos agora como determinar a matriz $D^{(j)}$ para o caso de modelos com *buffer* limitado e política de serviço exaustivo. Para tal, vamos derivar as expressões para o cálculo das suas probabilidades de transição $d_{s,s'}^{(j)}$. Aqui, diferentemente do caso de *buffer* ilimitado, chegadas às filas i ($i = 2, \dots, M$) são aceitas ou rejeitadas, dependendo se o respectivo *buffer* está cheio ou não. Como já vimos no capítulo 2, no caso de *buffer* limitado, não há expressões simples como as equações 3.5 e 3.9. Porém, com agregação de estados, será mostrado como desenvolver expressões similares, e também recursões, como nas equações 3.6 e 3.10, visando permitir o cálculo das probabilidades $d_{s,s'}^{(j)}$, de um modo eficiente.

Para tal, seja B_i o tamanho do *buffer* da fila i . Conseqüentemente, isto limita o número de chegadas que podem ser aceitas na fila i durante um mini-ciclo. Obviamente, o número máximo de chegadas aceitas depende do estado inicial da fila. Isto é, se a fila i tem q_i clientes no início do mini-ciclo, apenas as primeiras $B_i - q_i$ chegadas serão aceitas, e as eventuais chegadas posteriores serão rejeitadas.

Independente do estado inicial da fila i , qualquer chegada após as B_i primeiras não alterarão o estado da fila i , pois serão certamente rejeitadas. Por exemplo, todos os vetores do tipo (k_2, \dots, k_M) , representando os totais de chegadas às filas que não estão sendo servidas, que satisfazem $k_i \geq B_i$ ($i = 2, \dots, M$), identificam o mesmo vetor (B_2, \dots, B_M) de estado das filas, independente do estado inicial. Portanto, baseado neste fato, faremos uma agregação de estados para determinarmos equações similares às equações 3.5 e 3.9.

Seja $l = (l_2, \dots, l_M)$ um vetor, tal que $l_i \leq B_i$ para todo i , onde l_i representa o mínimo entre o tamanho do *buffer* e o número total de chegadas à fila i , ou seja, $l_i = \min(B_i, k_i)$. Seja $a \geq \sum_{i=2}^M l_i$, o número total de chegadas às filas que não estão sendo servidas. Note que $a = \sum_{i=2}^M k_i$. Seja também, o conjunto de vetores $K(l, a) = \{(k_2, \dots, k_M); a = \sum_{i=2}^M k_i, \text{ e } k_i = l_i \text{ se } l_i < B_i, \text{ ou } k_i \geq l_i \text{ se } l_i = B_i, \text{ para } i = 2, \dots, M\}$. Por exemplo, para $M = 3$ e $B_2 = B_3 = 2$, temos que $K((0, 0), 0) = \{(0, 0)\}$, $K((0, 0), 1) = \{\}$, $K((2, 1), 3) = \{(2, 1)\}$, $K((2, 1), 4) = \{(3, 1)\}$, $K((2, 2), 5) = \{(3, 2), (2, 3)\}$ e $K((2, 2), 6) = \{(3, 3), (4, 2), (2, 4)\}$.

Primeiramente, consideremos o caso onde o servidor só deixa a fila j , quando a respectiva quota de serviço T_j expira, ou seja, a duração τ_j do mini-ciclo j é igual a T_j , e não há absorção. Este é o mesmo caso 1, para serviço exaustivo, no capítulo 2. Definamos então [1],

$$\Phi[l_2, \dots, l_M, a; n] = \sum_{k \in K(l, a)} \alpha_2^{k_2} \dots \alpha_M^{k_M} \beta^{n-a} \binom{n}{k_2 \dots k_M} \quad (3.12)$$

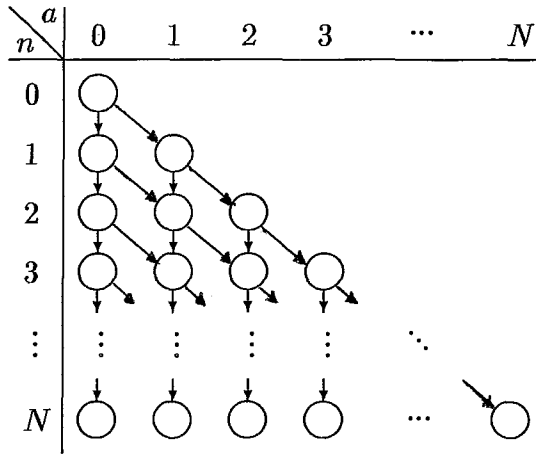
onde $n \geq a \geq \sum_{i=2}^M l_i$, como sendo a probabilidade de haver um total de a chegadas às filas $2, \dots, M$, com exatamente l_i chegadas à fila i (se $l_i < B_i$), ou pelo menos l_i chegadas à fila i (se $l_i = B_i$), dado que o número total de eventos é n . A equação 3.12 acima corresponde à equação 3.5, usada para modelos com *buffer* ilimitado. Correspondendo à equação 3.4 para modelos com *buffer* ilimitado, seja então, para $v \geq a \geq \sum_{i=2}^M l_i$, [1]

$$\Psi[l_2, \dots, l_M, a; v] = \sum_{n=a}^v P(n) \Phi[l_2, \dots, l_M, a; n] \pi_{q_1}^{(1)}(n-a) \quad (3.13)$$

uma soma parcial da probabilidade Φ descondicionada. Mais uma vez estamos considerando $N_{min} = 0$, para efeito de apresentação das fórmulas, visando simplificar a notação. Note também que, nesta equação, estamos considerando o estado da fila j sendo servida, através do termo $\pi_{q_1}^{(1)}(n-a)$, com $j = 1$.

É possível encontrar recursões similares às das equações 3.6 e 3.7, para computar eficientemente Φ e Ψ . Da mesma forma que antes, através da evolução no tempo da cadeia de Markov subjacente, modificada como na figura 3.1, podemos escrever uma fórmula recursiva para Φ como sendo [1]

$$\begin{aligned} \Phi[l_2, \dots, l_M, a; n] &= \sum_{i=2}^M \alpha_i \Phi[l_2, \dots, l_i - 1, \dots, l_M, a - 1; n - 1] \mathcal{I}(0 \leq l_i - 1 < B_i) \\ &+ \sum_{i=2}^M \alpha_i \Phi[l_2, \dots, l_i, \dots, l_M, a - 1; n - 1] \mathcal{I}(l_i = B_i) \end{aligned}$$

Figura 3.2: Recursão de Φ

$$+ \beta \Phi[l_2, \dots, l_M, a; n-1] \mathcal{I}(n-1 \geq a). \quad (3.14)$$

A primeira soma leva em conta a situação onde uma chegada à fila i foi aceita e armazenada no *buffer*. Isto só ocorre se $0 \leq l_i - 1 < B_i$, ou seja, o *buffer* da fila i não poderia estar anteriormente cheio. O evento de uma chegada à fila i ocorre com probabilidade α_i . Note que, tanto o número total de chegadas a , quanto o número total de eventos n sofrem um acréscimo. Já a segunda soma considera o caso onde uma chegada à fila i foi rejeitada devido ao *buffer* da fila estar cheio, com B_i clientes. Obviamente, só temos esta situação se $l_i = B_i$. O último termo representa o caso onde houve uma transição relativa à própria fila 1 sendo servida, ou seja, uma chegada ou uma partida. Uma transição deste tipo ocorre com probabilidade β . É fácil ver que esta situação só acontece se $n-1 \geq a$. Note também que somente o número total de eventos n sofre um acréscimo, já que não houve nenhuma chegada às filas $2, \dots, M$. A base da recursão na equação 3.14 é dada por $\Phi[0, \dots, 0, 0; 0] = 1$, o que é trivial. Correspondendo à recursão 3.7, temos aqui, para $N \geq v \geq a$, [1]

$$\begin{aligned} \Psi[l_2, \dots, l_M, a; v] &= \Psi[l_2, \dots, l_M, a; v-1] \mathcal{I}(v-1 \geq a) \\ &+ P(v) \Phi[l_2, \dots, l_M, a; v] \pi_{q_1}^{(1)}(v-a). \end{aligned} \quad (3.15)$$

Usando a noção de probabilidade total, sejam

$$\Phi[l_2, \dots, l_M; n] = \sum_{a=0}^n \Phi[l_2, \dots, l_M, a; n]$$

$$\Psi[l_2, \dots, l_M; v] = \sum_{a=0}^v \Psi[l_2, \dots, l_M, a; v]$$

as expressões para Φ e Ψ , independentes do número total de chegadas às filas que não estão sendo servidas $2, \dots, M$.

Na figura 3.2 temos uma representação gráfica da recursão de Φ . Lembre que, embora n varie de 0 a N , ao calcularmos $\Psi[l_2, \dots, l_M, a; N]$, só consideramos $N \geq n \geq \max(N_{min}, a)$. Cada círculo da figura 3.2 é chamado de célula, e

a	Duplas Válidas
0	(0,0)
1	(0,1), (1,0)
2	(0,2), (1,1), (2,0)
3	(0,2), (1,2), (2,0), (2,1)
4	(0,2), (1,2), (2,0), (2,1), (2,2)
5	(0,2), (1,2), (2,0), (2,1), (2,2)
⋮	⋮
N	(0,2), (1,2), (2,0), (2,1), (2,2)

Tabela 3.1: Sistema com $M = 3$ e $B_i = 2$

representa um conjunto de probabilidades $\Phi[l_2, \dots, l_M, a; n]$, para as várias $(M - 1)$ -uplas (l_2, \dots, l_M) válidas em relação a a . Cada célula é portanto um vetor de probabilidades. Note que para um certo valor de a , tal que $a \leq n$, temos apenas algumas $(M - 1)$ -uplas (l_2, \dots, l_M) válidas. Uma $(M - 1)$ -upla (l_2, \dots, l_M) é válida, dado a , se as três condições abaixo forem satisfeitas:

- $\sum_{i=2}^M l_i \leq a$
- $0 < l_i \leq B_i$, para $2 \leq i \leq M$
- se $l_i \neq B_i$, para $2 \leq i \leq M$, então $\sum_{i=2}^M l_i = a$

Por exemplo, em um sistema com $M = 3$ e $B_i = 4$, para $i = 1, 2, 3$, e com a fila 1 sendo servida, a dupla $(l_2, l_3) = (0, 4)$ é válida para $a = 5$, pois é possível que 5 clientes cheguem à fila 3, embora só 4 fiquem armazenados, devido ao tamanho do *buffer*. A dupla $(3, 2)$ também é válida para $a = 5$, pois todas as chegadas às filas 2 e 3 foram aceitas, com nenhum *buffer* ficando cheio. Já a dupla $(3, 1)$ não é válida para $a = 5$, pois dado que houve 5 chegadas no total, não pode ter havido somente três chegadas à fila 2 e uma chegada à fila 1, já que nenhum *buffer* ficou cheio. Vemos também que a dupla $(0, 5)$ não é válida para $a = 5$, pois por definição, o valor de l_3 pode ser no máximo igual a $B_3 = 4$. Nas tabelas 3.1 e 3.2, damos dois exemplos, onde apresentamos todas as $(M - 1)$ -uplas válidas para os vários valores de a . Em ambos exemplos, temos um sistema com 3 filas, com o mesmo tamanho de *buffer*, onde a fila servida é a de número 1. Nas tabelas 3.1 e 3.2, temos respectivamente, $B_i = 2$ e $B_i = 4$, para $i = 1, 2, 3$. É fácil ver que para um dado valor de a , as $(M - 1)$ -uplas (l_2, \dots, l_M) válidas são tais que, $\sum_{i=2}^M l_i = a$, se $a \leq B_i$, ou $B_i \leq \sum_{i=2}^M l_i \leq a$, se $a > B_i$.

Portanto, se $a > n$, ou (l_2, \dots, l_M) não é válida em relação a a , tem-se que $\Phi[l_2, \dots, l_M, a; n] = 0$. Note também que

$$\sum_{\substack{0 \leq l_i \leq B_i \\ 2 \leq i \leq M}} \sum_{a=0}^n \Phi[l_2, \dots, l_M, a; n] = 1.$$

a	Duplas Válidas
0	(0,0)
1	(0,1), (1,0)
2	(0,2), (1,1), (2,0)
3	(0,3), (1,2), (2,1), (3,0)
4	(0,4), (1,3), (2,2), (3,1), (4,0)
5	(0,4), (1,4), (2,3), (3,2), (4,0), (4,1)
6	(0,4), (1,4), (2,4), (4,0), (4,1), (4,2)
7	(0,4), (1,4), (2,4), (3,4), (4,0), (4,1), (4,2), (4,3)
8	(0,4), (1,4), (2,4), (3,4), (4,0), (4,1), (4,2), (4,3), (4,4)
9	(0,4), (1,4), (2,4), (3,4), (4,0), (4,1), (4,2), (4,3), (4,4)
⋮	⋮
N	(0,4), (1,4), (2,4), (3,4), (4,0), (4,1), (4,2), (4,3), (4,4)

Tabela 3.2: Sistema com $M = 3$ e $B_i = 4$

Por exemplo, em um sistema com $M = 3$, $B_i = 2$, onde $(i = 1, 2, 3)$, e com a fila 1 sendo servida, temos que $\Phi[0, 0, 0; 1] + \Phi[0, 1, 1; 1] + \Phi[1, 0, 1; 1] = \beta + \alpha_3 + \alpha_2 = 1$.

Finalmente consideremos o caso onde há absorção, quando a fila se esvazia antes de expirar a sua quota de serviço. Este é o mesmo caso 2 para serviço exaustivo no capítulo 2. Semelhante à Γ , definamos $\Delta[l_2, \dots, l_M, a; n]$ como sendo a probabilidade de haver um total de a chegadas às filas $2, \dots, M$, com exatamente l_i chegadas à fila i (se $l_i < B_i$), ou pelo menos l_i chegadas à fila i (se $l_i = B_i$), e também haver absorção (fila 1 ficar vazia) na m -ésima transição da cadeia, dado que houve n eventos no total. Portanto, para $n \geq a + 1$, [1]

$$\Delta[l_2, \dots, l_M, a; n] = \sum_{k \in K(l, a)} \sum_{m=a+1}^n \alpha_2^{k_2} \dots \alpha_M^{k_M} \beta^{m-a} \binom{m-1}{k_2 \dots k_M} P_{\text{abs}}^{(1)}(m-a). \tag{3.16}$$

Esta equação corresponde à equação 3.9 usada para modelos com *buffer* ilimitado. Da mesma forma que na derivação de Θ (equação 3.8), podemos descondicionar Δ

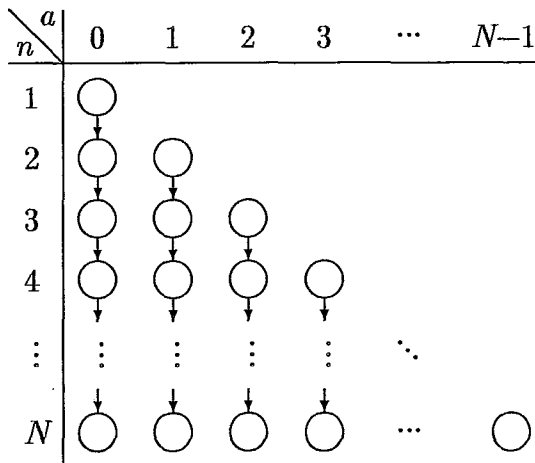


Figura 3.3: Recursão de Δ e Ξ

e obter, para $v \geq a + 1$, a seguinte soma parcial:

$$\Xi[l_2, \dots, l_M, a; v] = \sum_{n=a+1}^v P(n) \Delta[l_2, \dots, l_M, a; n]. \quad (3.17)$$

Lembre que, aqui também, assumimos $N_{min} = 0$. Finalmente, diretamente das expressões para Δ e Ξ nas equações 3.16 e 3.17, nos é possível escrever as equações recursivas [1]

$$\begin{aligned} \Delta[l_2, \dots, l_M, a; n] &= \Delta[l_2, \dots, l_M, a; n-1] \mathcal{I}(n-1 \geq a+1) \\ &+ \beta \Phi[l_2, \dots, l_M, a; n-1] P_{abs}^{(1)}(n-a) \end{aligned} \quad (3.18)$$

para $N \geq n \geq a + 1$, e

$$\begin{aligned} \Xi[l_2, \dots, l_M, a; v] &= \Xi[l_2, \dots, l_M, a; v-1] \mathcal{I}(v-1 \geq a+1) \\ &+ P(v) \Delta[l_2, \dots, l_M, a; v] \end{aligned} \quad (3.19)$$

para $N \geq v \geq a + 1$. Na figura 3.3, da mesma forma que para Φ , temos uma representação gráfica da recursão de Δ e Ξ . Mais uma vez, usando a noção de probabilidade total, sejam

$$\begin{aligned} \Delta[l_2, \dots, l_M; n] &= \sum_{a=0}^{n-1} \Delta[l_2, \dots, l_M, a; n] \\ \Xi[l_2, \dots, l_M; v] &= \sum_{a=0}^{v-1} \Xi[l_2, \dots, l_M, a; v] \end{aligned}$$

as expressões para Δ e Ξ , independentes do número total de chegadas às filas que não estão sendo servidas $2, \dots, M$.

Obviamente ao calcularmos $d_{s,s'}^{(j)}$, aqui no caso de *buffer* limitado, não podemos fazê-lo com a simples aplicação direta de uma fórmula, como nos modelos com *buffer* infinito, onde usamos diretamente as equações 3.4 e 3.8, com $v = N$. Ao invés, levando em conta o estado inicial das filas, devemos somar, através das probabilidades Ψ ou Ξ , dependendo do caso, todos os eventos possíveis que levem à transição de estado de s para s' . Lembramos que, como obtemos estas probabilidades por um processo de descondicionamento parcial, então no cálculo de $d_{s,s'}^{(j)}$, devemos utilizar as probabilidades $\Psi[l_2, \dots, l_M, a; N]$ ou $\Xi[l_2, \dots, l_M, a; N]$, onde $N = \text{máx}\{n, \text{tal que } P(n) \text{ é computável}\}$. Formalizando, ao calcularmos a probabilidade de transição $d_{s,s'}^{(j)}$, onde $s' = (j_1, j_2, \dots, j_M)$, $s = (i_1, i_2, \dots, i_M)$ e $j = 1$, devemos somar todas as probabilidades $\Psi[l_2, \dots, l_M, a; N]$ ou $\Xi[l_2, \dots, l_M, a; N]$, com $l_c = j_c - i_c$ (se $j_c \neq B_c$), ou $B_c - i_c \leq l_c \leq B_c$ (se $j_c = B_c$), para $c = 2, \dots, M$. Portanto, para uma dada transição de s para s' , temos vários vetores (l_2, \dots, l_M) representando os possíveis eventos na transição. Por exemplo, em um modelo com 4 filas, seja $d_{s,s'}^{(1)}$ a probabilidade de transição de $s = (3, 4, 0, 1)$ para $s' = (0, 4, 2, 4)$. Com $B_i = 4$, para $i = 1, \dots, 4$, então

$$d_{s,s'}^{(1)} = \sum_{l=(l_2, l_3, l_4) \in L} \Xi[l_2, l_3, l_4; N]$$

onde $L = \{(l_2, l_3, l_4); 0 \leq l_2 \leq 4, l_3 = 2 \text{ e } 3 \leq l_4 \leq 4\}$.

Agora, quando durante o cálculo de β , precisarmos de uma probabilidade de transição $d_{s,s'}^{(j)}$, temos apenas que somar as probabilidades $\Psi[l_2, \dots, l_M, a; N]$ ou $\Xi[l_2, \dots, l_M, a; N]$ adequadas, já calculadas e armazenadas anteriormente.

Com este método, ao invés de computarmos cada elemento $d_{s,s'}^{(j)}$ de $D^{(j)}$ individualmente, e armazenarmos a matriz inteira, apenas mantemos em memória uma estrutura de dados que nos permita computar todas as probabilidades $d_{s,s'}^{(j)}$ a partir das probabilidades $\Upsilon[k_2, \dots, k_M; N]$ e $\Theta[k_2, \dots, k_M; N]$, para modelos com *buffer* ilimitado, ou $\Psi[l_2, \dots, l_M, a; N]$ e $\Xi[l_2, \dots, l_M, a; N]$ para modelos com *buffer* limitado. Na próxima seção, discutiremos detalhes de como esta estrutura de dados foi projetada, modelada e implementada na ferramenta desenvolvida neste trabalho, bem como o seu custo de armazenamento. Contudo, podemos adiantar que este custo é bem menor que o custo para armazenar uma matriz $D^{(j)}$ completa, que é, como já vimos, exponencial com o número de estados do sistema modelado ($O(B^{2M})$).

3.5 Detalhes da Ferramenta Implementada

A ferramenta desenvolvida neste trabalho tem como objetivo calcular, empregando o método de solução de [1], detalhado no capítulo 2, diversas medidas de desempenho, em estado estacionário, para um sistema de *polling* com temporização contendo M filas, onde todas têm um *buffer* limitado de tamanho B . Por exemplo, podemos calcular o retardo médio de clientes nas várias filas, além de probabilidades de bloqueio, em modelos com *buffer* limitado. Para tal, a distribuição de probabilidade do número de clientes armazenados nas filas, em estado estacionário, deve ser calculada primeiro. Os parâmetros de entrada para a ferramenta são as taxas de chegada de pacotes λ , as taxas de serviço μ e as quotas de serviço T , para as várias filas do sistema sendo modelado. A ferramenta foi desenvolvida em linguagem C e em uma estação de trabalho SUN SPARC-2.

Inicialmente, o vetor de probabilidades de estado do sistema β é obtido. Relembrando, o estado do sistema é modelado por uma M -upla (q_1, q_2, \dots, q_M) , onde M é o número de filas no sistema e q_i é o número de clientes armazenados na fila i ($i = 1, \dots, M$) naquele instante. Em outras palavras, q_i representa o estado da fila i . O vetor β é implementado diretamente como um vetor de dimensão $(B+1)^M$, que é o número de estados de um sistema com M filas e tamanho de *buffer* igual a B . Os estados do sistema são organizados em β de acordo com uma mudança de base, entre as bases decimal e a base $B+1$, gerando uma ordem lexicográfica. Dado um estado do sistema, representado em base $B+1$, conseguimos, através de mudança de base, obter a sua posição em base decimal, no vetor β . Por outro lado, a partir de uma posição, representada em base decimal, obtemos da mesma forma o seu estado correspondente, em base $B+1$. Por exemplo, para $M = 3$ e $B = 2$, o estado $(0, 1, 0)$ está na 3ª posição de β . Portanto, identificamos aqui a seguinte

relação: (posição do estado em β)₁₀ = (estado)_{B+1}.

O vetor β é calculado pela ferramenta da maneira como foi discutida no capítulo 2. Relembrando, β é obtido passo a passo, nos pontos embutidos, onde é sucessivamente multiplicado por uma matriz de transição de estado ($D^{(j)}$ ou $S^{(j)}$, dependendo do caso) até alcançar a convergência. Na implementação, a convergência, tomada em relação a um ponto embutido fixo, é alcançada quando o erro em β for menor ou igual a 10^{-3} . No vetor β inicial, antes da primeira multiplicação, cada elemento é igual a $1/(B+1)^M$, ou seja, o vetor β inicial é uniformemente distribuído. A medida que ele vai sendo multiplicado, a sua distribuição definitiva vai se formando até convergir.

A obtenção de um novo vetor β após um intervalo de *switch-over* é implementada pela ferramenta exatamente como foi discutido anteriormente na seção 3.2. Assim, ao invés das matrizes $S^{(j)}$ serem armazenadas inteiras, uma outra estrutura de dados é mantida, com uma grande redução no custo de armazenamento. Quando for preciso obter um novo vetor β após um intervalo de *switch-over*, usamos esta estrutura mediante certas operações de cálculo. Da mesma forma, a obtenção de um novo vetor β após um mini-ciclo de serviço é realizada pela ferramenta empregando o método que foi discutido anteriormente na seção 3.4. A tolerância de erro ϵ , admitida para os cálculos, é igual a 10^{-10} . Para o cálculo das medidas de desempenho, a mesma tolerância é adotada.

A estrutura de dados que substitui as matrizes $D^{(j)}$ inteiras é composta de duas subestruturas. A primeira é utilizada para o cálculo das probabilidades $d_{s,s'}^{(j)}$ para transições da caso 1, onde não há absorção. Por restrições de tempo no desenvolvimento da ferramenta, não mantivemos, na implementação do método de solução, uma estrutura de dados para armazenar as probabilidades $\Psi[l_2, \dots, l_M, a; n]$ no cálculo de $d_{s,s'}^{(j)}$ para o caso 1. Ao invés disso, a subestrutura implementada pela ferramenta está modelada com dois vetores tridimensionais. O primeiro armazena as várias probabilidades de chegadas Poisson às filas durante os mini-ciclos de serviço. O primeiro índice deste vetor indica o número de chegadas, o segundo indica a fila a qual estão chegando os clientes, e o terceiro índice indica a fila onde está ocorrendo o mini-ciclo de serviço. Este vetor contém $B(M)^2$ elementos. O segundo vetor armazena as várias probabilidades transientes de estado de uma fila, a partir de um estado inicial, calculadas usando randomização de um processo de Markov. Neste vetor, o primeiro índice informa a fila sendo servida, o segundo informa o estado final da fila e o terceiro índice informa o estado inicial da fila. Este segundo vetor contém $M(B+1)B$ elementos. Ambos os vetores são preenchidos antes do processo de iteração de β ter início. Como no cálculo de um elemento $S_{s,s'}^{(j)}$, ao precisarmos de uma probabilidade $d_{s,s'}^{(j)}$, temos que resolver a equação 2.10, utilizando as probabilidades de chegadas Poisson e as probabilidades transientes de estado já calculadas e armazenadas. A segunda subestrutura é utilizada para o cálculo de $d_{s,s'}^{(j)}$ para transições do caso 2, com absorção. Esta subestrutura está modelada com um conjunto de vetores de ponteiros, para efeito de indexação, mais um conjunto de vetores para armazenamento das probabilidades $\Phi[l_2, \dots, l_M, a; n]$, $\Delta[l_2, \dots, l_M, a; n]$ e $\Xi[l_2, \dots, l_M, a; n]$. Para cada fila j , temos um vetor de ponteiros onde cada elemento,

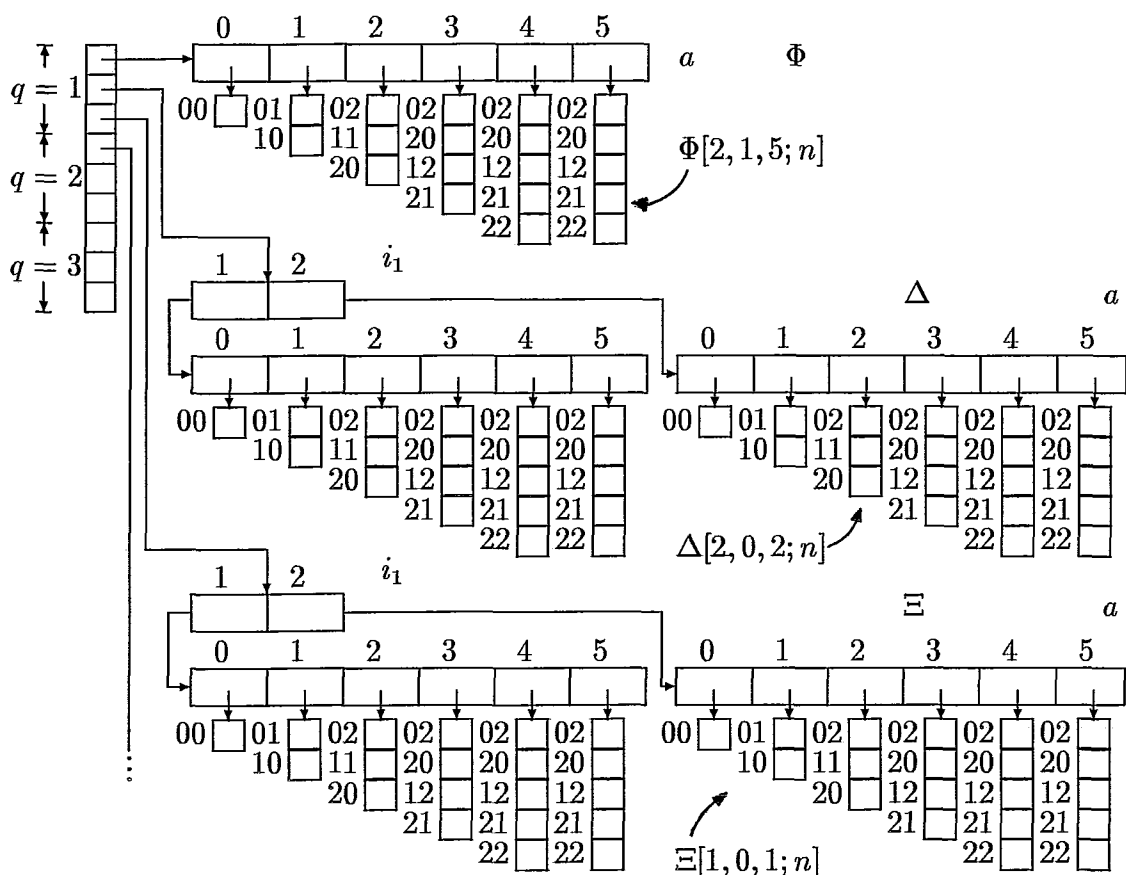


Figura 3.4: Sub-estrutura para o Cálculo de $d_{s,s'}^{(j)}$

associado a um valor de a , aponta para um vetor de probabilidades $\Phi[l_2, \dots, l_M, a; n]$ com dimensão igual ao número de $(M-1)$ -uplas (l_2, \dots, l_M) válidas para a . Portanto, cada elemento deste vetor de probabilidades está associado a uma $(M-1)$ -upla (l_2, \dots, l_M) válida para aquele valor de a . A dimensão de um vetor de ponteiros é igual a N , calculado em relação à fila j . Logo, valores de a de 0 até $N-1$ são considerados. Um algoritmo de indexação, discutido em [12], foi empregado para, a partir de uma $(M-1)$ -upla (l_2, \dots, l_M) para um certo valor de a , obter a sua posição correspondente no vetor de probabilidades. O algoritmo usado também faz a operação inversa, ou seja, fornece uma $(M-1)$ -upla a partir de sua posição no vetor. Para tal, o algoritmo constrói uma matriz de trabalho e depois a utiliza para as indexações. Também, para cada fila j e para cada valor do estado inicial da fila j , com exceção do 0 (fila vazia), temos uma estrutura idêntica para armazenar as probabilidades $\Delta[l_2, \dots, l_M, a; n]$ e $\Xi[l_2, \dots, l_M, a; n]$. Portanto, para cada fila j , $2B$ estruturas idênticas à de Φ são mantidas para guardar as probabilidades Δ e Ξ . Na figura 3.4, vemos um exemplo dessa segunda subestrutura, para um sistema com $M=3$, $B=2$ e $N=6$.

No que tange especificamente à implementação realizada neste trabalho, o algoritmo de indexação usado para as $(M-1)$ -uplas válidas (l_2, \dots, l_M) exige que, às vezes, algumas $(M-1)$ -uplas não válidas também sejam armazenadas juntamente com as válidas. Neste caso, alguns vetores de probabilidades terão seus

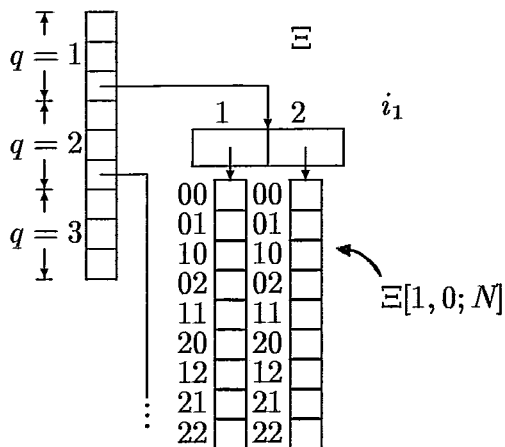


Figura 3.5: Sub-estrutura Após a Reorganização

tamanhos aumentados. Porém, este custo extra de armazenamento é muito pequeno e pode ser desprezado. Por exemplo, a subestrutura da figura 3.4, ao ser implementada pela ferramenta, terá uma pequena diferença: os vetores de probabilidades que abrigam as probabilidades Φ , Δ e Ξ , para $3 \leq a \leq 5$, têm um elemento extra correspondendo à dupla inválida $(l_2, l_3) = (1, 1)$. Isto tem a ver com o fato de que, para $a > B$, as $(M - 1)$ -uplas (l_2, \dots, l_M) válidas são tais que $B \leq \sum_{i=2}^M l_i \leq a$, embora o contrário não se verifique.

Inicialmente, as probabilidades $\Xi[l_2, \dots, l_M, a; n]$ são calculadas usando as recursões envolvendo Δ e Φ , já discutidas. Depois, para efeito de eficiência, calculamos as probabilidades $\Xi[l_2, \dots, l_M; N]$, ou seja, obtemos Ξ independente de a , fazendo

$$\Xi[l_2, \dots, l_M; N] = \sum_{a=0}^{N-1} \Xi[l_2, \dots, l_M, a; N].$$

Neste momento há uma reorganização na subestrutura, visando liberar memória para outras estruturas serem armazenadas posteriormente. Os vetores para Φ e Δ são destruídos pois não serão mais utilizados. Com a eliminação da dependência em a , as probabilidades Ξ são agora armazenadas na seguinte subestrutura: para cada fila j e para cada valor do estado da fila j , com exceção do 0 (fila vazia), temos um vetor de probabilidades $\Xi[l_2, \dots, l_M; N]$ com dimensão igual ao número de $(M - 1)$ -uplas (l_2, \dots, l_M) válidas para todos os valores de a considerados. Veja na figura 3.5 como a subestrutura da figura 3.4 fica após a reorganização. Finalmente, ao precisarmos de uma probabilidade $d_{s,s'}^{(j)}$, para multiplicar por um vetor β , basta somarmos as probabilidades $\Xi[l_2, \dots, l_M; N]$ adequadas.

Uma vez obtido o vetor β , em estado estacionário, isto acontecendo ao se alcançar a convergência tomada em relação a um ponto embutido fixo, os demais vetores β são então obtidos nos outros pontos embutidos do ciclo de *polling*. Juntamente com os vetores β , probabilidades marginais de estado do sistema em estado estacionário são obtidas e armazenadas para uso posterior no cálculo das probabilidades de estado de uma fila em particular, em estado estacionário. As

probabilidades marginais são de três tipos: probabilidades de haver k clientes em uma certa fila de interesse nos pontos embutidos associados aos inícios dos vários intervalos de *switch-over*, probabilidades de haver k clientes em uma fila no ponto embutido correspondente ao início do mini-ciclo de serviço desta mesma fila, e probabilidades de haver k clientes em uma certa fila de interesse e l clientes em uma outra fila, no ponto embutido correspondente ao início do mini-ciclo de serviço desta última. A estrutura de dados usada para armazenar as probabilidades marginais é composta de três vetores. Para o primeiro tipo, é usado um vetor tridimensional de dimensão $(B + 1)(M)^2$; para o segundo tipo, um vetor bidimensional de dimensão $(B + 1)M$; e para o terceiro tipo é usado um vetor quadridimensional de dimensão $(B + 1)^2(M)^2$.

Com as probabilidades marginais obtidas e armazenadas, as probabilidades $P_{\mathcal{R}}$ são então calculadas exatamente como foi apresentado na sub-seção 2.3.2. Lembre que aqui, \mathcal{R} é um sub-conjunto de estados, onde uma dada fila tem um certo número n de clientes armazenados, e $P_{\mathcal{R}}$ é portanto a probabilidade de haver n clientes nesta fila, em estado estacionário. A ferramenta desenvolvida neste trabalho calcula, para todas as filas do sistema, a probabilidade da fila conter n clientes, para $0 \leq n \leq B$. Como as quantidades do denominador da equação 2.17 são independentes de \mathcal{R} , elas são calculadas e armazenadas inicialmente representando o valor esperado do ciclo de *polling*.

Finalmente, as probabilidades $P_{\mathcal{R}}$ são usadas para determinar medidas de desempenho de interesse, tais como, retardo médio de clientes (e. g. pacotes) nas filas, e probabilidades de bloqueio em modelos com *buffer* limitado.

Vamos agora estimar o custo de armazenamento das duas subestruturas. Primeiramente, estimaremos o custo para a segunda subestrutura. Dividiremos a análise em dois casos. No primeiro caso, temos $N - 1 \leq B$. Para exemplificar, sem perda de generalidade, tomemos um modelo com tamanho de *buffer* limitado, com $N - 1 \leq B$, onde todas as filas têm parâmetros idênticos, ou seja, têm as mesmas quotas de serviço e as mesmas taxas de chegada e de serviço. Seja a fila 1, a fila servida. A estimativa é a mesma para modelos com *buffer* ilimitado. Neste modelo, para cada valor de a , as $(M - 1)$ -uplas (l_2, \dots, l_M) válidas correspondentes são tais que $\sum_{i=2}^M l_i = a$. Portanto, para cada valor de a , existe um vetor de probabilidades com uma quantidade de elementos dada por

$$\binom{a + M - 2}{a}$$

onde M é o número de filas do sistema. Seja T' o número total de elementos da segunda subestrutura. Aqui, estamos considerando tanto os vetores de probabilidades, quanto os vetores de ponteiros. Contudo, na realidade, o conjunto de vetores de ponteiros tem pouca influência na estimativa. Como $0 \leq a \leq N$, e são necessários $M(1 + 2B)$ conjuntos de vetores para armazenar as probabilidades Ω , Γ e Θ , temos então

$$T' = M \cdot \sum_{a=0}^{N-1} \binom{a + M - 2}{a}$$

$$\begin{aligned}
& + 2MB \cdot \sum_{a=0}^{N-1} \binom{a+M-2}{a} \\
& + M[N + 2B(N+1)] + 3M.
\end{aligned}$$

O primeiro termo se refere ao conjunto de vetores que é usado no cálculo das probabilidades Ω . O segundo termo, por sua vez, se refere aos conjuntos de vetores que armazenam as probabilidades Γ e Θ , onde B indica que existe 1 conjunto para cada valor de i_q , para $1 \leq q \leq M$. Já o terceiro termo corresponde ao total de elementos dos conjuntos de ponteiros na subestrutura. Resolvendo os somatórios de combinações, e rearrumando a expressão acima, ficamos com

$$T' = M \left[\binom{N-2+M}{N-1} + 2B \binom{N-2+M}{N-1} \right] + M[N + 2B(N+1) + 3]$$

ou ainda

$$T' = C_1 \binom{N-2+M}{N-1} + C_2$$

onde $C_1 = M(1 + 2B)$ e $C_2 = M[N + 2B(N+1) + 3]$. Em termos de complexidade, temos, para N grande

$$T' \approx O(N^{M-1}).$$

Observa-se que em casos práticos de sistemas reais, N não é de alta magnitude, pois é proporcional ao número de mensagens (pacotes) que podem ser servidas em um miniciclo. Nos casos estudados, N é da ordem de 100.

No segundo caso, onde $N - 1 > B$, encontraremos o valor limitante superior do espaço total gasto para armazenar a segunda subestrutura, o que nos dá uma estimativa para o custo com esse armazenamento. Mais uma vez, sem perda de generalidade, tomemos como exemplo, um modelo com tamanho de *buffer* limitado, com $N - 1 > B$, onde todas as filas têm parâmetros idênticos. Mais uma vez, a fila 1 é a fila servida. Neste modelo, para cada valor de a , tal que $0 \leq a \leq B$, as $(M - 1)$ -uplas (l_2, \dots, l_M) válidas correspondentes são tais que $\sum_{i=2}^M l_i = a$. Logo, como no caso anterior, para cada valor de a , existe um vetor de probabilidades com uma quantidade de elementos dada por

$$\binom{a+M-2}{a}.$$

Por sua vez, para cada valor de a , tal que $B + 1 \leq a \leq N - 1$, as $(M - 1)$ -uplas (l_2, \dots, l_M) válidas correspondentes são tais que satisfaçam a condição $B \leq \sum_{i=2}^M l_i \leq \min(a, (M - 1)B)$. No entanto, a recíproca não é verdadeira. Por exemplo, em um modelo onde $M = 3$, $N = 6$ e $B = 2$, e com $a = 4$, as $(M - 1)$ -uplas (l_2, \dots, l_M) válidas são: $(0,2)$, $(2,0)$, $(1,2)$, $(2,1)$ e $(2,2)$. Note que as $(M - 1)$ -uplas, $(1,1)$, $(0,3)$, $(3,0)$, $(0,4)$, $(1,3)$, $(3,1)$ e $(4,0)$, não são válidas para $a = 4$, embora satisfaçam a condição $2 \leq \sum_{i=2}^M l_i \leq 4$. O valor para o número total T' de elementos da segunda subestrutura que calcularemos é limitante superior, pois contamos mais elementos do que a subestrutura realmente tem. No entanto, existe um algoritmo simples, implementado pela ferramenta, que permite calcular a quantidade correta de espaço

necessária para o armazenamento da subestrutura. O valor limitante superior para o número total T' de elementos da segunda subestrutura é dado por

$$\begin{aligned} SUP(T') &= M\left[\sum_{a=0}^B \binom{a-2+M}{a}\right] + \sum_{a=B+1}^{N-1} \sum_{l=1}^X \binom{X}{l} B^{X-l} \\ &+ 2MB\left[\sum_{a=0}^B \binom{a-2+M}{a}\right] + \sum_{a=B+1}^{N-1} \sum_{l=1}^X \binom{X}{l} B^{X-l} \\ &+ M[N + 2B(N + 1)] + 3M \end{aligned}$$

onde $X = \min((M - 1), \lfloor \frac{a}{B} \rfloor)$. Reescrevendo a expressão acima, temos então para $a \geq B(M - 1)$

$$\begin{aligned} SUP(T') &= M\left[\binom{B-1+M}{B}\right] + (N - 1 - (B + 1))[(B + 1)^{M-1} - B^{M-1}] \\ &\times [1 + 2B] \\ &+ M[N + 2B(N + 1) + 3]. \end{aligned}$$

Em termos de complexidade, para B grande, temos agora

$$SUP(T') \approx O(B^M).$$

Agora analisaremos a primeira subestrutura. Conforme vimos, essa subestrutura, usada para o cálculo das probabilidades de transição $d_{s,s'}^{(j)}$, do caso 1, foi implementada neste trabalho com $T'' = B(M)^2 + M(B + 1)B$ elementos. Portanto

$$T'' \approx O(B(M)^2 + M(B)^2).$$

Juntando as duas subestruturas, temos que $T = T' + T''$ é o número de elementos da estrutura que substitui as matrizes $D^{(j)}$. Em termos de custo, finalmente obtemos

$$T \approx O(B^M) + O(B(M)^2 + M(B)^2).$$

Capítulo 4

Resultados Numéricos

Neste capítulo, apresentaremos resultados numéricos relativos à sensibilidade das medidas de desempenho, retardo médio por pacote e probabilidade de bloqueio, frente aos vários parâmetros de configuração de sistema, tais como: número de estações (filas), tamanho (dimensão) dos *buffers* das filas, carga de pacotes que chega ao sistema, quotas de tempo de serviço para as filas, e duração de intervalos de *switch-over*. Também é feita uma análise comparativa entre o esquema estudado neste trabalho e o esquema de multiplexação TDM, a fila M/M/1/K e o esquema de prioridades do sistema HOL (*Head-Of-the-Line*).

Todos os resultados a seguir, apresentados através de curvas, foram obtidos de experimentos com diversas configurações de sistemas com 2 e 3 filas. Os sistemas com 2 filas (indicadas por fila 1 e 2) representam sistemas (T_1, T_2) , e os sistemas com 3 filas (indicadas por fila 1, 2 e 3) são usados no estudo comparativo de retardo com o esquema de prioridades do sistema HOL. Lembremos que todas as filas têm o mesmo tamanho de *buffer* B . Nos experimentos, o intervalo de *switch-over* relativo a uma fila é tomado como sendo 1% do valor da quota de tempo de serviço correspondente. Todas as unidades de tempo, para as quotas de tempo de serviço e os intervalos de *switch-over*, são dadas em milissegundos (ms) e as taxas de chegada (λ) e de serviço (μ) de pacotes são dadas em pacotes por milissegundo (pac/ms). Estas unidades, e também alguns valores de parâmetros, foram escolhidas para permitir comparações com resultados apresentados em [2], a partir de simulações. Inclusive, é importante atestar que os resultados analíticos obtidos aqui são bastante coerentes com resultados de simulações. Inicialmente, os casos apresentados são para sistemas com 2 filas.

4.1 Influência do Tamanho do *Buffer*

Para analisarmos o comportamento ou sensibilidade do sistema em relação ao tamanho do *buffer* B das filas, adotamos a seguinte configuração: seja um sistema de

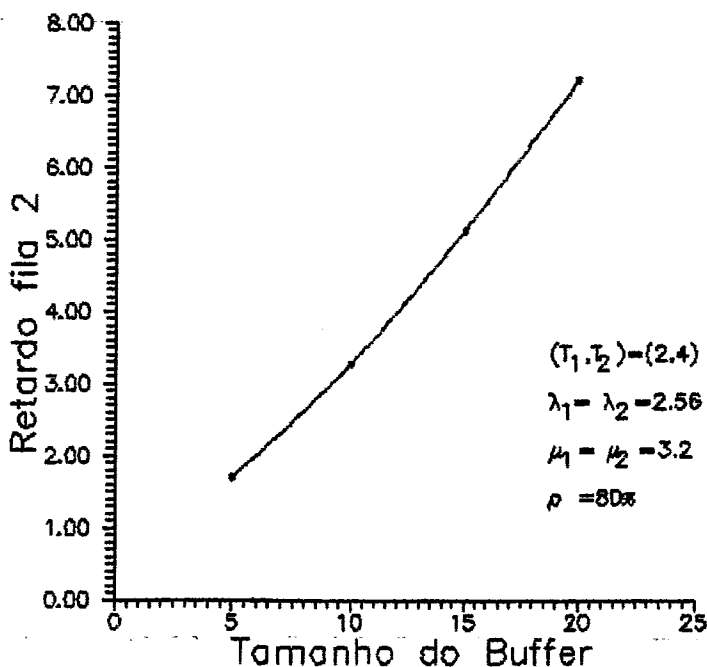


Figura 4.1: Retardo X B

duas filas, usando o esquema (T_1, T_2) , onde $T_1 = 2$ e $T_2 = 4$, com taxas de chegada idênticas $\lambda_1 = \lambda_2 = 2.56$ e taxas de serviço idênticas $\mu_1 = \mu_2 = 3.2$. A utilização total do sistema ρ é de 80%. Podemos observar na figura 4.1 o comportamento do retardo por pacote na fila 2, frente à variação do tamanho do *buffer* das duas filas. Aqui, o retardo cresce com o aumento do *buffer*, pois haverá mais pacotes no *buffer* expandido da fila para serem servidos. O retardo médio calculado considera portanto, uma maior quantidade de retardos individuais de pacotes. Nesta situação, a probabilidade de bloqueio na fila 2 é alta, ou seja, é alta a probabilidade do seu *buffer* estar cheio de pacotes aguardando serviço. Isto se verifica, pois se a probabilidade de bloqueio fosse baixa, ou seja, o *buffer* estivesse subutilizado, o retardo seria insensível ao aumento do tamanho do *buffer*. Não adiantaria aumentar a capacidade do *buffer*, se este espaço extra provavelmente não seria utilizado por um pacote. Aqui, como a probabilidade de bloqueio é alta, quanto mais aumentarmos o tamanho do *buffer*, mais pacotes estarão armazenados aguardando serviço, e portanto maior será o retardo por pacote na fila 2.

Nas figuras 4.2 e 4.3 a seguir temos a sensibilidade das medidas probabilidade de bloqueio e retardo, frente a variações do tamanho do *buffer* B das filas em um sistema totalmente simétrico com 2 filas. Na figura 4.2, temos a evolução da probabilidade de bloqueio em uma fila, frente a variação do tamanho do *buffer* das duas filas. Como podemos ver, a probabilidade de bloqueio é inversamente proporcional ao tamanho do *buffer* das filas. Esta relação se dá, pois quanto maior for o tamanho dos *buffers*, para armazenar os pacotes nas filas, menor é a probabilidade de que um pacote que chegue a uma fila, encontre o seu *buffer* lotado. Já na figura 4.3, temos indicada a evolução do retardo em uma fila, frente a variação do tamanho do *buffer* B das filas. Conforme vimos, quanto maior for o tamanho do *buffer*, maior será o retardo, pois mais pacotes estarão armazenados nas filas aguardando serviço,

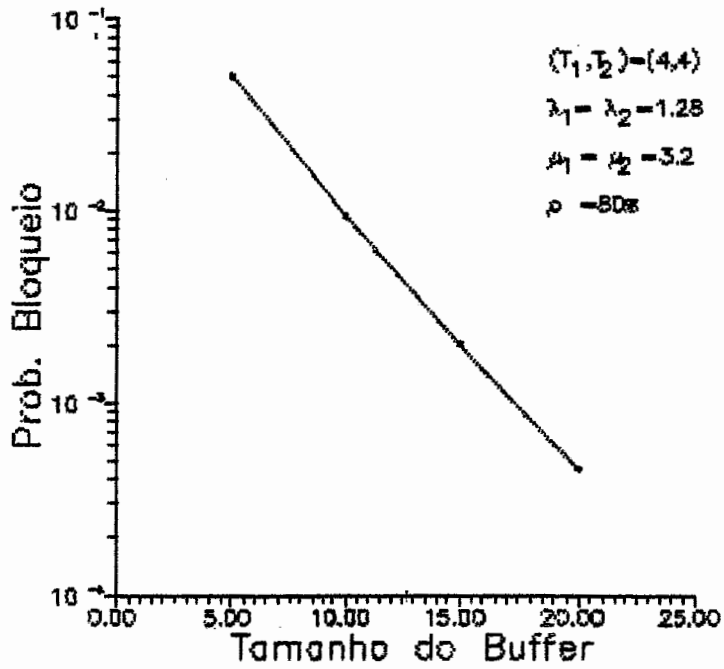


Figura 4.2: Prob. Bloqueio X B (Sistema Simétrico)

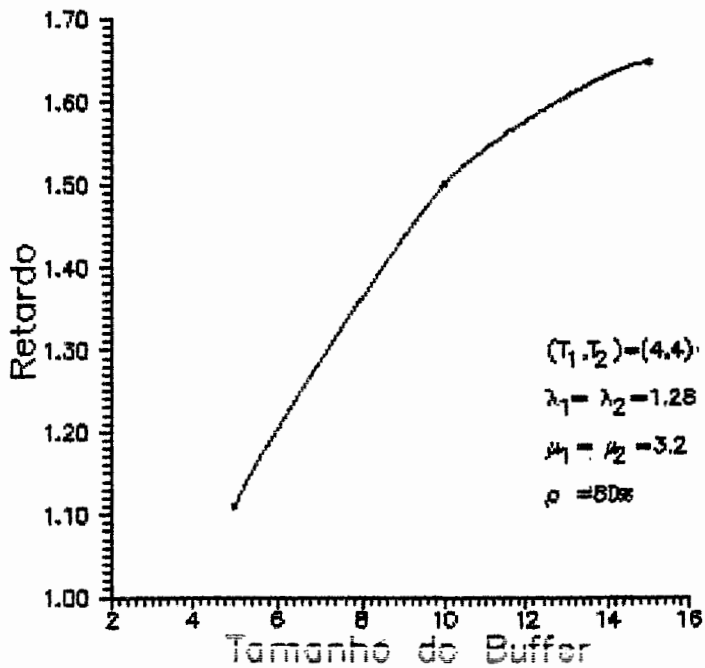


Figura 4.3: Retardo X B (Sistema Simétrico)

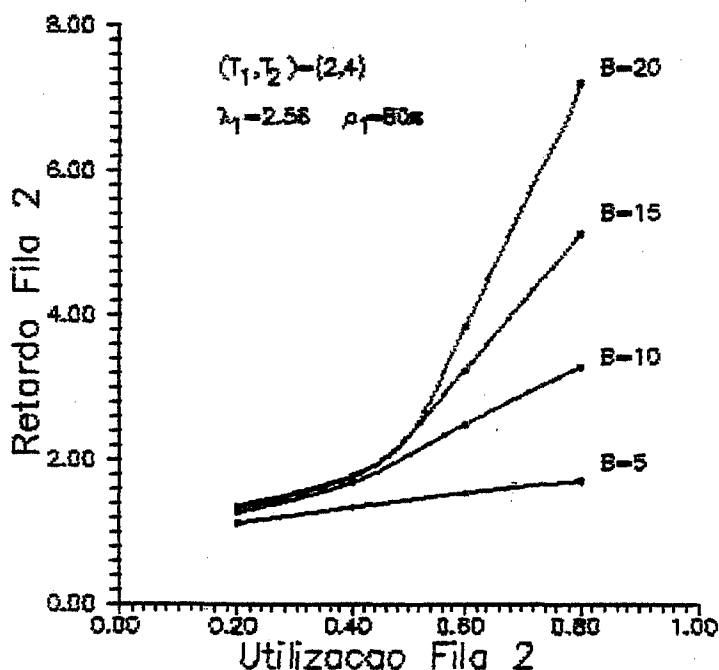


Figura 4.4: Retardo X Utilização da Mesma Fila

e portanto contribuindo com os seus retardos individuais para aumentar o retardo calculado. Como a probabilidade de bloqueio é baixa, conforme a figura 4.2, para B entre 10 e 15, não adianta aumentar o tamanho do *buffer*, pois este espaço extra, com grande probabilidade, não será ocupado. Daí porque, a variação do retardo é relativamente pequena nesta faixa.

Na figura 4.4 temos o comportamento do retardo na fila 2 em relação à utilização ρ_2 , da própria fila 2, para vários tamanhos de *buffer*. Nesta configuração, as taxas de serviço μ_1 e μ_2 são iguais a 3.2. É fácil ver que, quanto mais pacotes chegam à fila 2, maior será o retardo por pacote nesta fila, pois leva mais tempo para servir um número maior de pacotes. Para a curva com $B = 5$, a partir de $\rho_2 = 80\%$, a probabilidade de bloqueio na fila 1 é relativamente alta (em torno de 0.3), e o seu *buffer* está quase lotado. Neste instante, o efeito limitante do *buffer* finito surge. Não adianta aumentar a taxa de chegada de pacotes (utilização), que não aumentará mais o retardo, pois com o *buffer* lotado, os pacotes que chegam serão descartados. Obviamente, a tendência das demais curvas é também tender a um retardo máximo limitante, com o aumento de ρ_2 .

4.2 Influência dos Intervalos de *Switch-over*

Agora analisaremos a sensibilidade do retardo, por pacote no sistema, ao se variar os intervalos de *switch-over* entre as filas. No gráfico da figura 4.5, o intervalo de *switch-over*, entre uma fila e outra, é indicado por um percentual em relação à duração da quota de tempo de serviço T_i , ($i = 1, 2$), da fila seguinte na ordem de visitas do servidor. Por exemplo, o intervalo de *switch-over* SO_2 tem duração igual

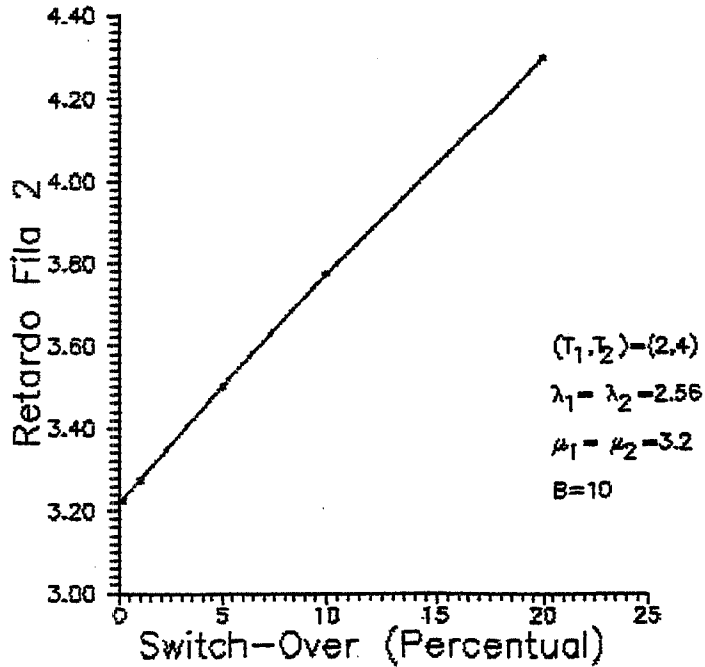


Figura 4.5: Influência dos Intervalos de *Switch-over*

a 5% da duração de T_2 , ou seja, $SO_2 = 0.5T_2$. Portanto, no eixo horizontal da figura 4.5 estão indicados os percentuais, e não valores nominais dos intervalos de *switch-over*. Com a utilização das filas em 80%, o retardo medido é respectivo à fila 2. Para um pacote nesta fila aguardando serviço, quanto maior forem os intervalos de *switch-over*, maior será o seu retardo, pois levará mais tempo para o servidor se deslocar entre as filas após expiradas as quotas T_1 e T_2 , desperdiçando assim tempo para a efetiva transmissão. Os pacotes terão que esperar mais tempo, para que o servidor chegue à fila 2, após concluir o serviço da outra fila.

4.3 Influência das Quotas de Tempo de Serviço (T_1, T_2)

A seguir, analisaremos a sensibilidade das medidas de desempenho de interesse, retardo e probabilidade de bloqueio, face a variações nos valores das quotas (T_1, T_2) , em diferentes configurações de sistema. Note que, mantendo-se os demais parâmetros constantes, a variação (aumento ou redução) das quotas T_1 e T_2 não implica em variação da capacidade do sistema. Note também que, embora não muito natural, ao variarmos as quotas de tempo de serviço, variamos também os intervalos de *switch-over*, uma vez que estes são uma percentagem das quotas. Inicialmente, estudaremos os casos onde $T_1 = T_2$. Para o primeiro caso na figura 4.6, temos um sistema totalmente simétrico. Tomemos $(T_1, T_2) = (2, 2)$ como o ponto base para analisar as variações. Por exemplo, se formos de $(T_1, T_2) = (2, 2)$ para $(T_1, T_2) = (1, 1)$, significa que dividimos, igualmente pela metade, os valores das quotas, ou também, se formos de $(T_1, T_2) = (2, 2)$ para $(T_1, T_2) = (6, 6)$, significa que triplicamos as quotas. Seja k , com $k \in \{0.5, 1, 2, 3\}$, o fator multiplicador do par $(T_1, T_2) = (2, 2)$ base,

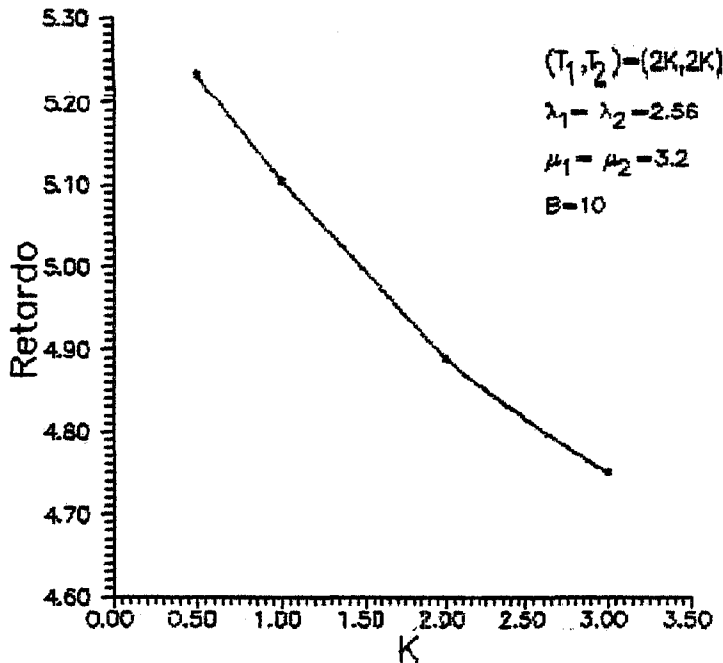


Figura 4.6: Retardo X (T_1, T_2) (Sistema Simétrico)

isto é, $(4, 4) = k(2, 2)$, onde $k = 2$. No eixo horizontal da figura 4.6, por exemplo, temos indicado o valor de k . Portanto, a partir dele, temos o valor do par (T_1, T_2) correspondente, tendo como base o par $(T_1, T_2) = (2, 2)$. Na figura 4.6, estudamos a curva do retardo em uma fila contra a variação dos parâmetros T_1 e T_2 . À medida que diminuimos os valores de T_1 e T_2 igualmente, maior é o número de ciclos para transmissão dos pacotes, o que faz com que os períodos de *switch-over* influenciem no retardo, embora a influência desses períodos seja de segunda ordem. Neste caso, o retardo cresce. Além disso, quanto menor for (T_1, T_2) , maior será o número de interrupções que um pacote sendo transmitido sofrerá, levando a um aumento na taxa de retransmissões. Claramente, isto também leva a um aumento no retardo.

Já nas figuras 4.7 e 4.8, a mesma análise de retardo é estendida para um sistema não simétrico. Aqui, a fila 2 recebe em média o dobro de pacotes por unidade de tempo que a fila 1. Na figura 4.7, observamos que o retardo da fila 1 cresce com o aumento de (T_1, T_2) . Isto se dá principalmente porque os pacotes na fila 2, que é a mais utilizada ($\rho_2 = 50\%$), utilizam efetivamente o acréscimo na quota de tempo T_2 para serviço, ocupando mais o servidor. Além disto, na fila 1, que é pouco utilizada, há frequentes absorções (a fila esvazia antes da quota T_1 expirar), e portanto não adianta aumentar o valor de T_1 , que este tempo extra não será utilizado efetivamente pela fila para transmissão. Outro fator importante, é que com o aumento das quotas (T_1, T_2) , os intervalos de *switch-over*, que são um percentual dessas quotas, também aumentam, levando então a um aumento no retardo. Já na figura 4.8, é apresentado o retardo na fila 2. Partindo de $(T_1, T_2) = (1, 1)$, vemos que inicialmente o retardo cai com o aumento de (T_1, T_2) . Isto é explicado pelo fato de que, com uma quota T_2 maior, os pacotes são servidos em menos ciclos de *token*, portanto mais rapidamente. Porém, a partir de $(T_1, T_2) = (6, 6)$, este retardo começa a crescer, pois os intervalos de *switch-over* ficam maiores, aumentando assim o ciclo

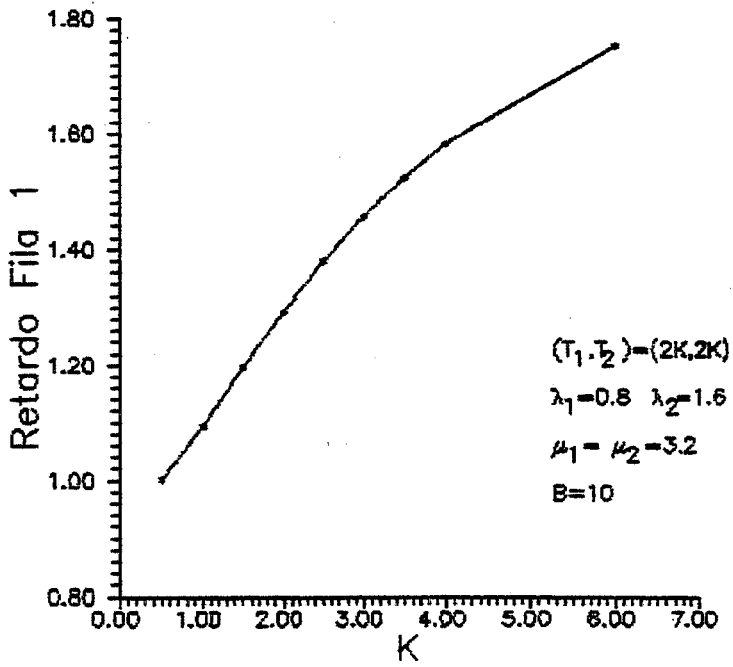


Figura 4.7: Retardo Fila 1 X (T_1, T_2) (Sistema não Simétrico)

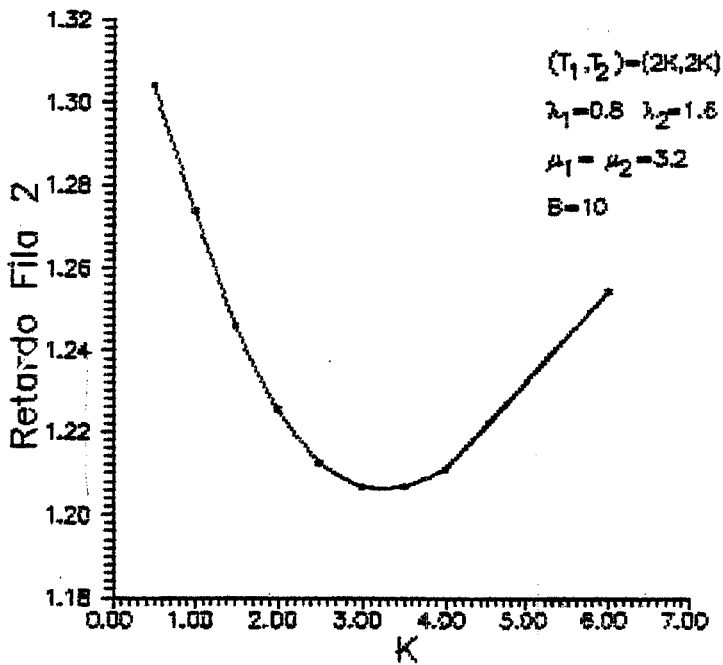


Figura 4.8: Retardo Fila 2 X (T_1, T_2) (Sistema não Simétrico)

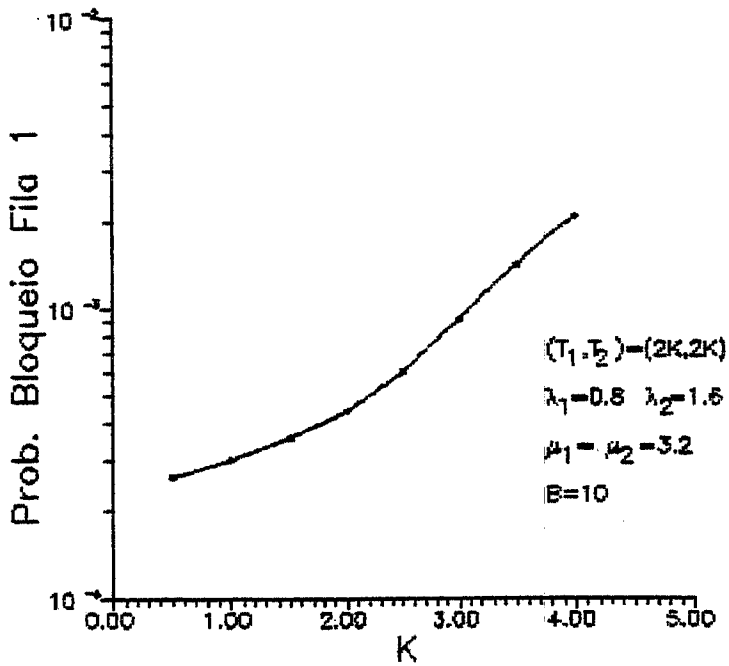


Figura 4.9: Prob. Bloqueio Fila 1 X (T_1, T_2) (Sistema não Simétrico)

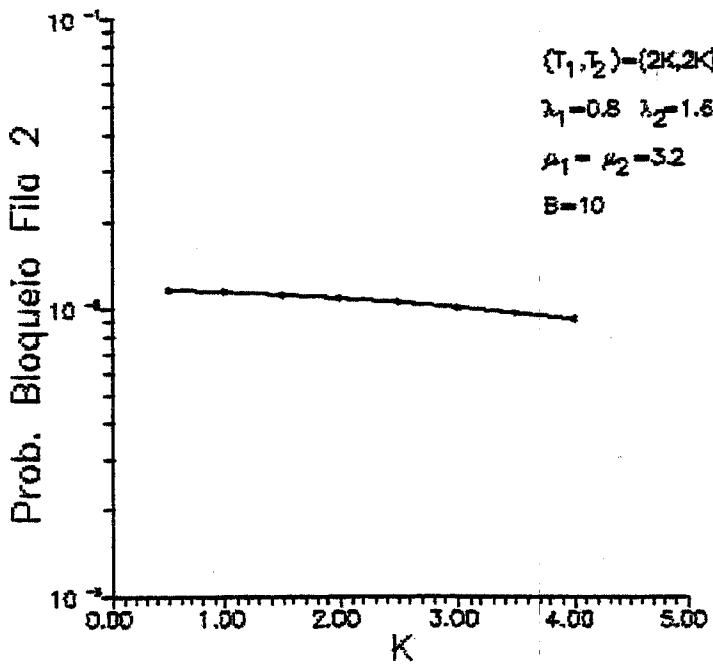


Figura 4.10: Prob. Bloqueio Fila 2 X (T_1, T_2) (Sistema não Simétrico)

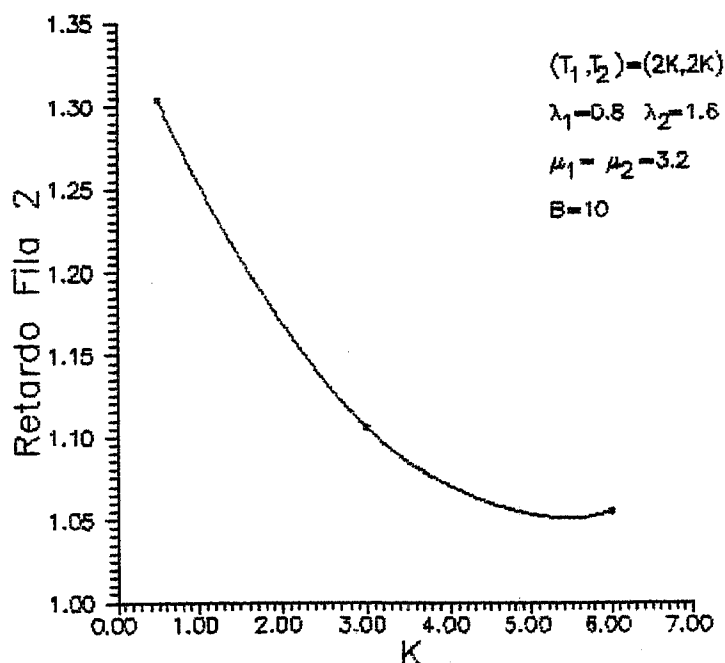


Figura 4.11: Retardo Fila 2 X (T_1, T_2) (*Switch-Over* Constante)

de *token*. Embora com uma contribuição menor para o crescimento do retardo, o serviço de pacotes da fila 1 começa a ter uma influência predominante sobre o retardo da fila 2. Note que a probabilidade de bloqueio da fila 1 começa a crescer (figura 4.9), o que indica que a fila 2 está influenciando cada vez mais a fila 1. O aumento do retardo na fila 1 indica um maior número de pacotes sendo servido na própria fila 1, e a influência deste retardo começa a predominar sobre o efeito benéfico do aumento de T_2 para a fila 2. Além disto, como vemos na figura 4.10, a probabilidade de bloqueio na fila 2 vai diminuindo, sendo então inútil o aumento de T_2 . Supondo agora que os intervalos de *switch-over* são constantes e iguais a 10^{-2} durante a variação das quotas (T_1, T_2) , vemos na figura 4.11 a curva do retardo da fila 2 para o mesmo sistema. Agora, com os intervalos de *switch-over* constantes, o retardo cai mais bruscamente com o aumento de (T_1, T_2) . A partir de (T_1, T_2) aproximadamente igual a $(12, 12)$, o retardo começa a crescer por influência da fila 1. Em ambas filas, verificamos que no limite $(T_1, T_2) = (\infty, \infty)$, o retardo da fila tende a ser igual ao retardo em um sistema tradicional de *polling* exaustivo. Como a probabilidade de bloqueio é baixa, os resultados para $B = 10$ são semelhantes aos para *buffer* ilimitado. As figuras 4.8 e 4.11 indicam claramente que podemos calibrar o sistema variando as quotas T_1 e T_2 . Daí, na figura 4.8, se desejamos que o retardo da fila 2 seja o menor possível, devemos fazer com que (T_1, T_2) seja aproximadamente em torno de $(6.5, 6.5)$. Este efeito é muito interessante e mostra a importância deste trabalho no projeto de um sistema real (T_1, T_2) .

Na figura 4.12, os retardos da fila 1 e 2 são confrontados. Se no projeto de um sistema (T_1, T_2) , objetiva-se obter um retardo conjunto das duas filas o menor possível, podemos definir uma medida de desempenho *RC* (Retardo Conjunto) e determinar o seu valor mínimo. Seja

$$RC = R_1 \times R_2$$

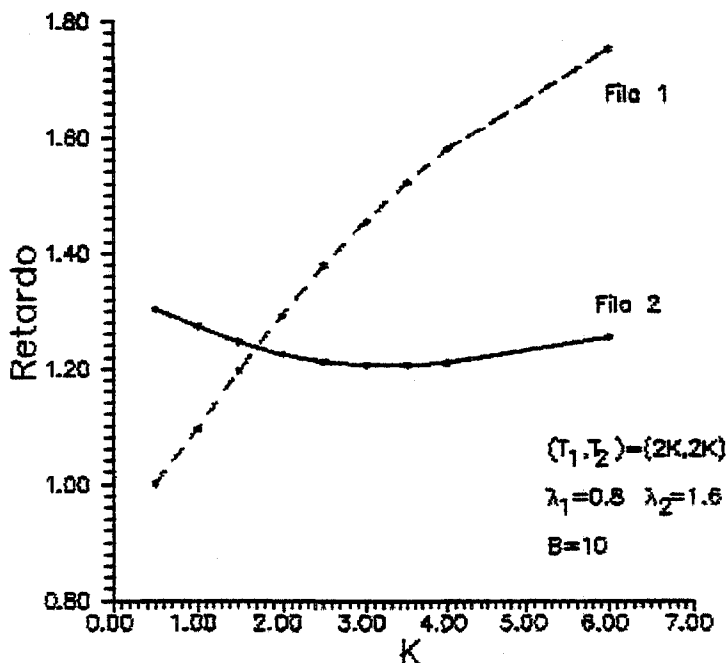
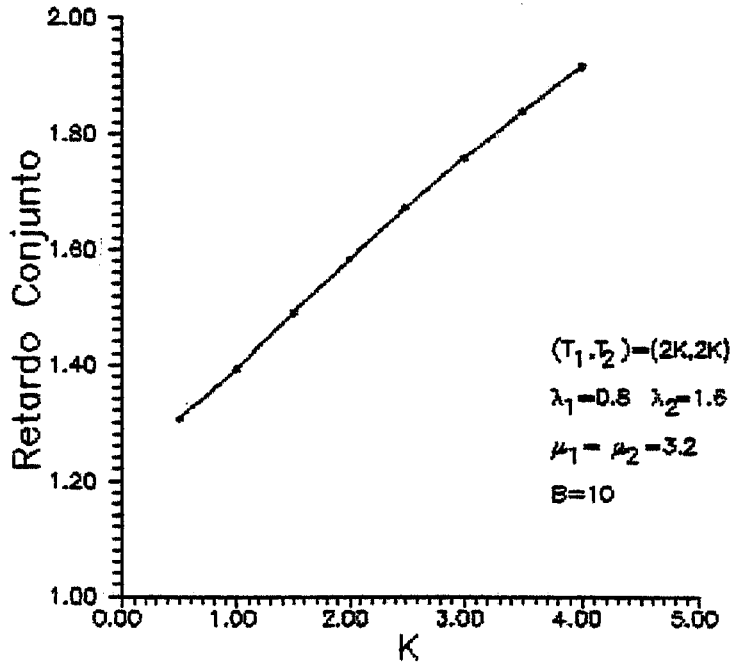
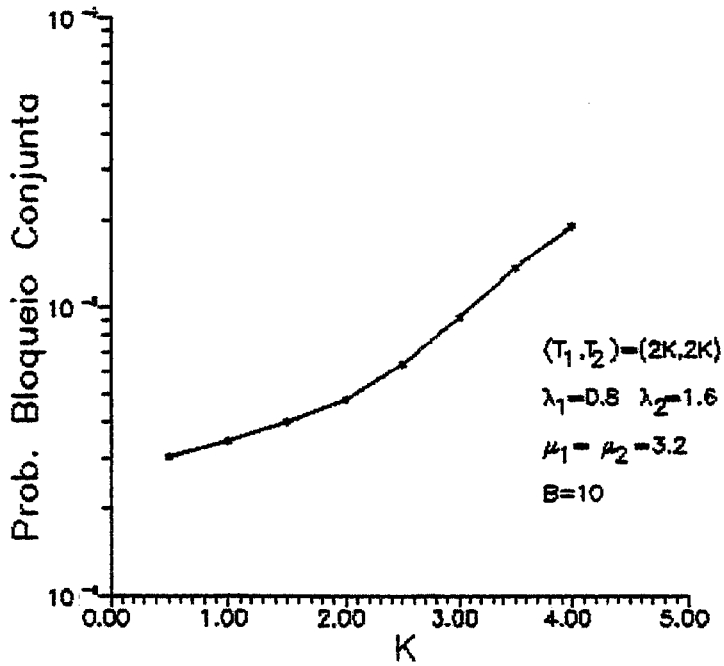


Figura 4.12: Retardos X (T_1, T_2)

onde R_1 e R_2 são os retardos das filas 1 e 2 respectivamente. Na figura 4.13 temos a curva de RC em relação à variação de (T_1, T_2) . Note que o retardo conjunto é mínimo quando (T_1, T_2) é aproximadamente $(1, 1)$, ou seja, nesta região, o retardo da fila 1 e o retardo da fila 2 são, conjuntamente, mínimos. Entretanto para (T_1, T_2) em torno de $(6.5, 6.5)$, obtém-se um retardo ótimo para a fila 2.

Para este mesmo sistema assimétrico, analisaremos agora o comportamento da probabilidade de bloqueio frente a variações das quotas de serviço T_1 e T_2 . Na figura 4.9, temos a curva da probabilidade de bloqueio da fila 1 contra (T_1, T_2) . À medida que aumentamos os valores das quotas, a probabilidade cresce. Isto acontece pois a fila 2, mais utilizada, passa a gastar cada vez mais tempo servindo seus pacotes, e fazendo assim com que os pacotes da fila 1 fiquem mais tempo aguardando serem servidos, ocupando assim o *buffer*. Embora não indicado no gráfico, observamos que, para valores de (T_1, T_2) muito pequenos (menores que o tempo médio para transmissão de um pacote $1/\mu = 0.3125$) a probabilidade de bloqueio é muito alta. Nestas configurações, em média, não é possível servir um único pacote sequer por visita, e daí, o *buffer* está quase sempre lotado. Se formos aumentando (T_1, T_2) , a probabilidade vai diminuindo, pois é possível servir mais pacotes, até voltar a crescer, em torno de $(T_1, T_2) = (1, 1)$, devido à influência da fila 2. A curva da probabilidade de bloqueio da fila 2 é apresentada na figura 4.10. Diferente da fila 1, a fila 2, mais utilizada, se privilegia com o aumento de (T_1, T_2) . Com isto, obviamente a probabilidade de bloqueio tende a diminuir. Tanto na fila 1 quanto na fila 2, as probabilidades tendem a um certo limite que é pequeno, para grandes valores de T_1 e T_2 .

Se no projeto de um sistema (T_1, T_2) , devemos estabelecer uma configuração de parâmetros, tal que as probabilidades de bloqueio nas filas sejam conjun-

Figura 4.13: $RC X (T_1, T_2)$ Figura 4.14: $PBC X (T_1, T_2)$

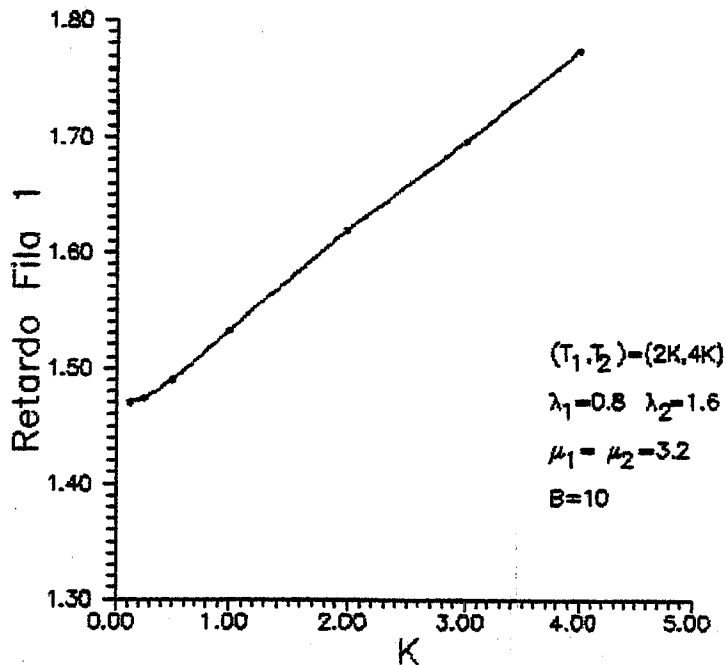


Figura 4.15: Retardo Fila 1 X (T_1, T_2) ($T_2 = 2T_1$)

tamente as menores possíveis, podemos definir uma medida de desempenho *PBC* (Probabilidade de Bloqueio Conjunta) e determinar o seu valor mínimo. Seja

$$PBC = PB_1 \times PB_2$$

onde PB_i é a probabilidade de bloqueio na fila i . Na figura 4.14, a curva da probabilidade de bloqueio conjunta *PBC*, das filas 1 e 2, para vários pares (T_1, T_2) , é apresentada.

Agora, vejamos a evolução do retardo por pacote e da probabilidade de bloqueio para casos onde $T_2 = 2T_1$. Para efeito das variações, o par (T_1, T_2) base é igual a $(2, 4)$. Os demais parâmetros são os mesmos do sistema recém analisado, ou seja, $\lambda_1 = 0.8$, $\lambda_2 = 1.6$, $\mu_1 = \mu_2 = 3.2$ e $B = 10$. Os valores do par (T_1, T_2) , no eixo horizontal, também são aqui representados pelo fator multiplicador k , ou seja, para um valor k no eixo, entendamos $(T_1, T_2) = (2k, 4k)$. Nas figuras 4.15 e 4.16, são apresentados os retardos nas filas 1 e 2, respectivamente, frente (T_1, T_2) . Em ambos os casos, quando (T_1, T_2) tende para $(0, 0)$, o retardo tende para infinito. Para valores de T_1 e T_2 muito pequenos, em média não dá para servir ninguém em uma visita do servidor. Daí, o retardo de um pacote na fila é imenso. À medida que aumentarmos (T_1, T_2) , este retardo vai diminuindo, pois os pacotes começam a ser servidos. No entanto, o tempo gasto com o serviço destes pacotes faz depois o retardo voltar a crescer. Na figura 4.15, é mostrada a curva do retardo da fila 1. O retardo, para valores não muito pequenos de (T_1, T_2) , cresce com o aumento de (T_1, T_2) , pois são precisos mais ciclos de *token* para a fila transmitir seus pacotes, devido à fila 2 utilizar efetivamente o acréscimo de T_2 , uma vez que ela tem uma maior utilização. Além disso, como já vimos, os intervalos de *switch-over* estão aumentando. A tendência é a subida da curva ir sofrendo uma desaceleração advinda do aumento da probabilidade de bloqueio na fila 1 (veja o gráfico na figura 4.18),

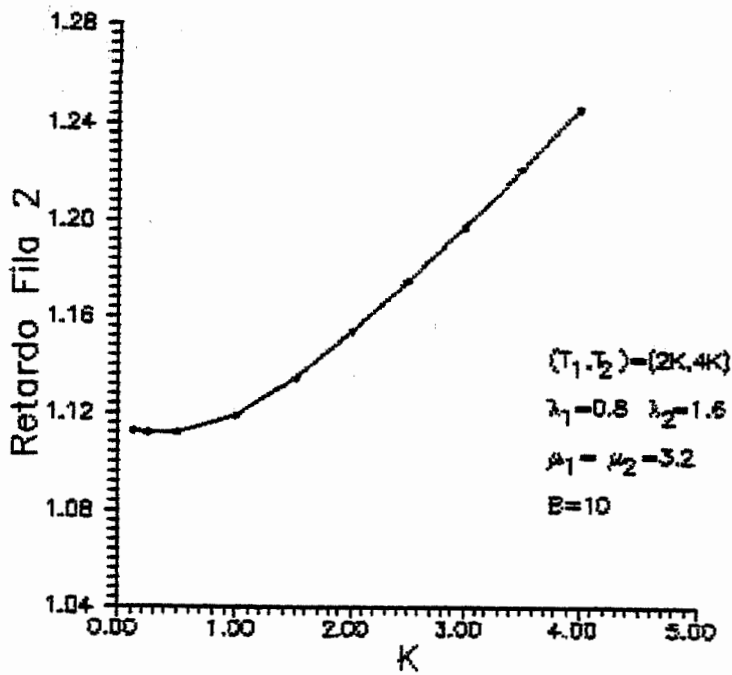


Figura 4.16: Retardo Fila 2 X (T_1, T_2) ($T_2 = 2T_1$)

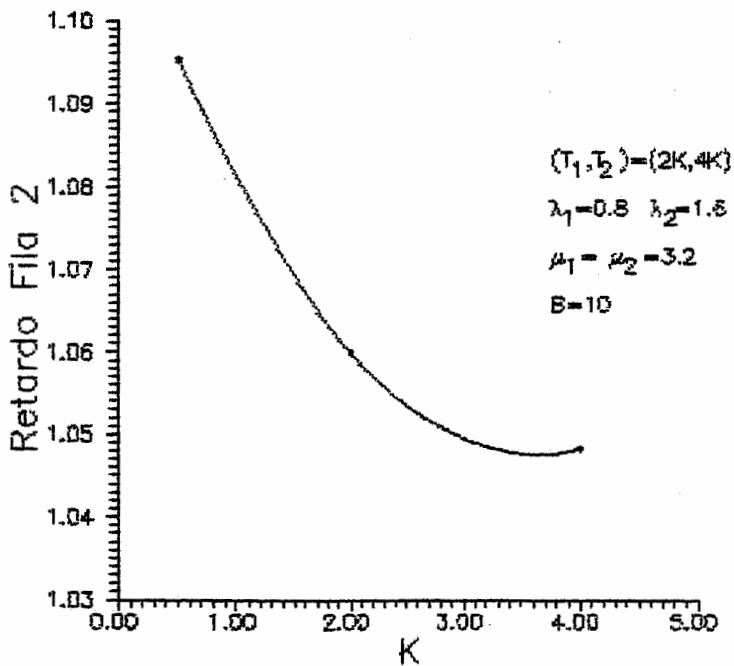


Figura 4.17: Retardo Fila 2 X (T_1, T_2) (Switch-Over Constante)

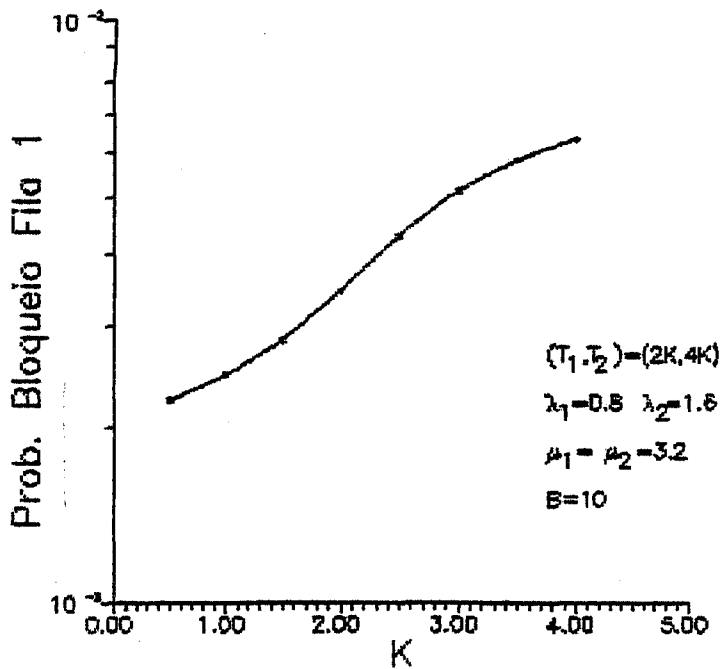


Figura 4.18: Prob. Bloqueio Fila 1 X (T_1, T_2) ($T_2 = 2T_1$)

o que indica que esta fila passa a usar mais a sua quota T_1 . Na figura 4.16, é mostrada a curva do retardo da fila 2. A partir de $(T_1, T_2) = (1, 2)$, ele cresce pois os intervalos de *switch-over* vão crescendo e assim aumentam o ciclo de *token*. O crescimento quase linear da curva comprova essa influência. Também verifica-se que, embora menos influente, aumenta o número de pacotes na fila. Como a fila 2 é mais utilizada, ela aproveita bastante o aumento de T_2 para transmitir seus pacotes, e fica portanto com mais pacotes recém chegados armazenados e aguardando serviço. Note que a probabilidade de bloqueio na fila 2 também está crescendo (veja figura 4.19). Supondo agora novamente os intervalos de *switch-over* constantes e iguais a 10^{-2} durante a variação de (T_1, T_2) , temos na figura 4.17 o comportamento do retardo da fila 2 para esse mesmo sistema. Veja que, com os intervalos de *switch-over* constantes, o retardo cai com o aumento de (T_1, T_2) . Isto acontece pois, como a fila 2 é mais utilizada, ela passa a usar mais o canal, reduzindo assim o número de ciclos de *token* para transmissão. Depois, o retardo começa a sofrer um aumento devido à influência da fila 1. Mais uma vez, quando (T_1, T_2) tender para (∞, ∞) , o sistema tende para um sistema tradicional de *polling* exaustivo. É válido também, observar que o retardo da fila 1 é maior, em valor absoluto, que o da fila 2, o que era esperado. Tanto na figura 4.15 quanto na figura 4.16, podemos encontrar o ponto ótimo onde o retardo é mínimo.

Na figura 4.18, analisamos a forma da curva da probabilidade de bloqueio da fila 1 contra (T_1, T_2) . A probabilidade cresce, pois quando se aumenta T_2 , além de se aumentar os intervalos de *switch-over*, está-se dando mais tempo para serviço à fila 2, que efetivamente a usa, por ser a mais utilizada. Dar mais tempo para serviço à fila 2 significa obviamente, tirar mais tempo para serviço da fila 1, o que faz com que os pacotes desta fila demorem mais a serem servidos. A curva da probabilidade de bloqueio da fila 2 é mostrada na figura 4.19. O seu comportamento

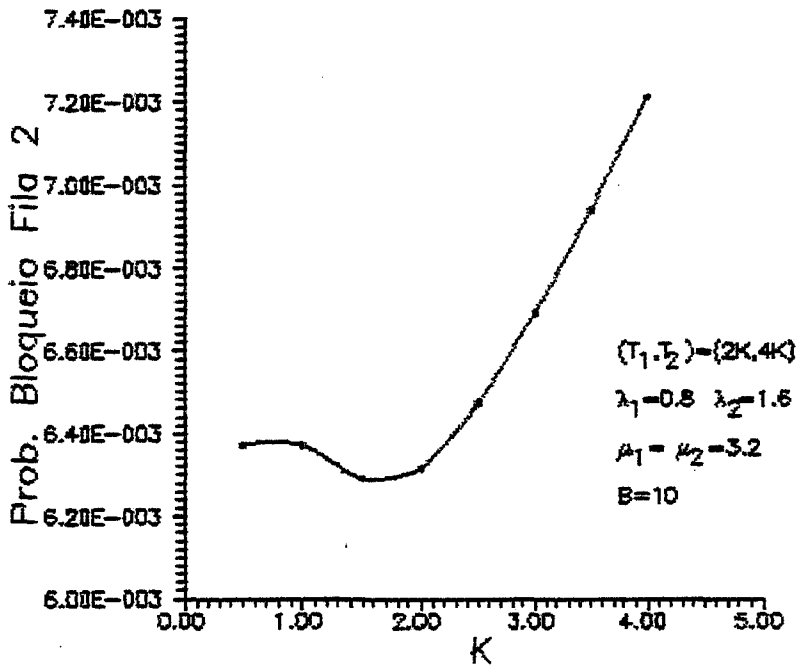


Figura 4.19: Prob. Bloqueio Fila 2 X (T_1, T_2) ($T_2 = 2T_1$)

pode ser dividido em duas etapas: primeiro, a probabilidade cai e depois sobe com o aumento de (T_1, T_2) . A queda se dá pois, os pacotes da fila, que antes não podiam ser transmitidos devido a quota T_2 expirar e tinham que retornar à fila, vão sendo transmitidos e liberando espaço no *buffer*. A subida da probabilidade ocorre pois, além da influência dos intervalos de *switch-over*, a fila 1 começa a usar mais a sua quota T_1 . Para visualizar melhor, podemos supor, sem perda de generalidade, que o *buffer* da fila 2 está lotado. Note que a fila 2, com $T_2 = 4$, já está relativamente folgada para transmitir, uma vez que ela leva em média aproximadamente 3 ms para esvaziar seu *buffer* lotado com 10 pacotes. Portanto o acréscimo em T_2 é inútil para a fila 2. O que realmente tem influência aqui, é a utilização mais efetiva da capacidade pela fila 1.

Temos agora, nas figuras 4.20 e 4.21, um sistema onde as filas 1 e 2 têm as mesmas taxas de chegada e de serviço, mas ainda $T_2 = 2T_1$. Vamos supor, sem perda de generalidade, que a fila 2 está folgada para transmitir e a fila 1 não. Isto é razoável pois, como já vimos, as filas levam em média aproximadamente 3 ms para esvaziar um *buffer* lotado com 10 pacotes, e o par (T_1, T_2) base é igual a $(2, 4)$. O retardo da fila 1, na figura 4.20, cai com o aumento de (T_1, T_2) , porque diminui o número de ciclos de *token* para transmitir os pacotes da fila. O aumento de T_2 nesta região não interfere muito pois a fila 2 já estava folgada inicialmente. No entanto, se continuarmos a crescer (T_1, T_2) , o retardo sofrerá um aumento advindo do fato da outra fila passar a usar mais a sua quota T_2 . Por esta mesma razão, o retardo da fila 2, na figura 4.21, cresce com o aumento de T_1 . Podemos achar também aqui, um ponto ótimo onde os retardos sejam conjuntamente mínimos.

Para um sistema (T_1, T_2) , com $T_1 = T_2$, e $\lambda_1 = 0.8$, $\lambda_2 = 1.6$, temos nas figuras 4.22, 4.23 e 4.24, a seguir, a influência do tamanho do *buffer* B das

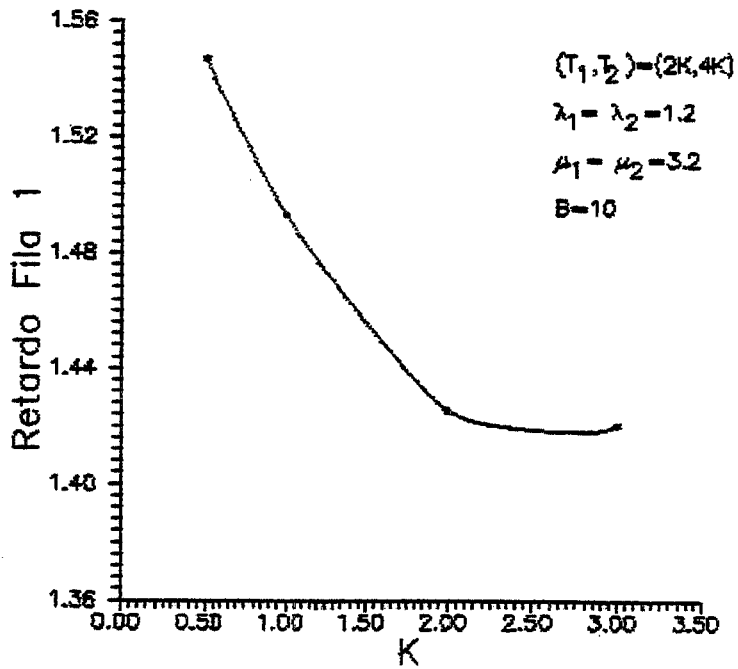


Figura 4.20: Retardo Fila 1 X (T_1, T_2) ($\lambda_1 = \lambda_2$ e $\mu_1 = \mu_2$)

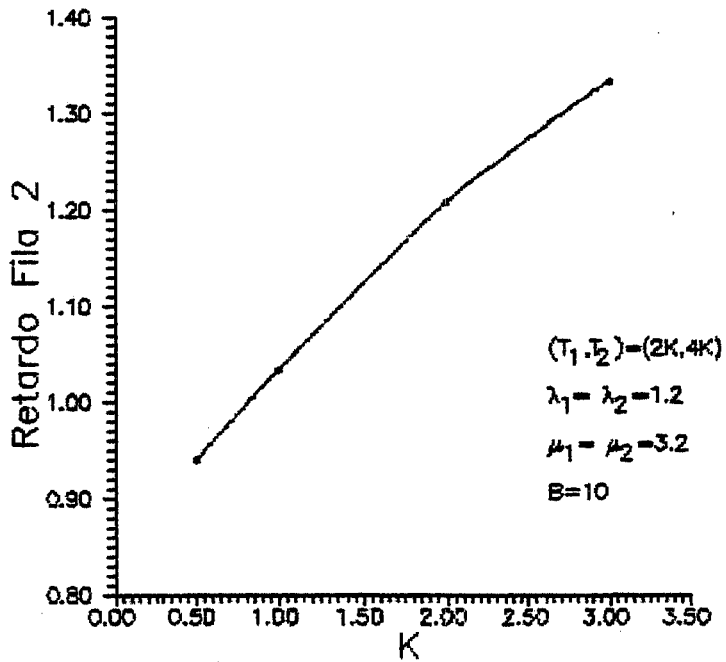


Figura 4.21: Retardo Fila 2 X (T_1, T_2) ($\lambda_1 = \lambda_2$ e $\mu_1 = \mu_2$)

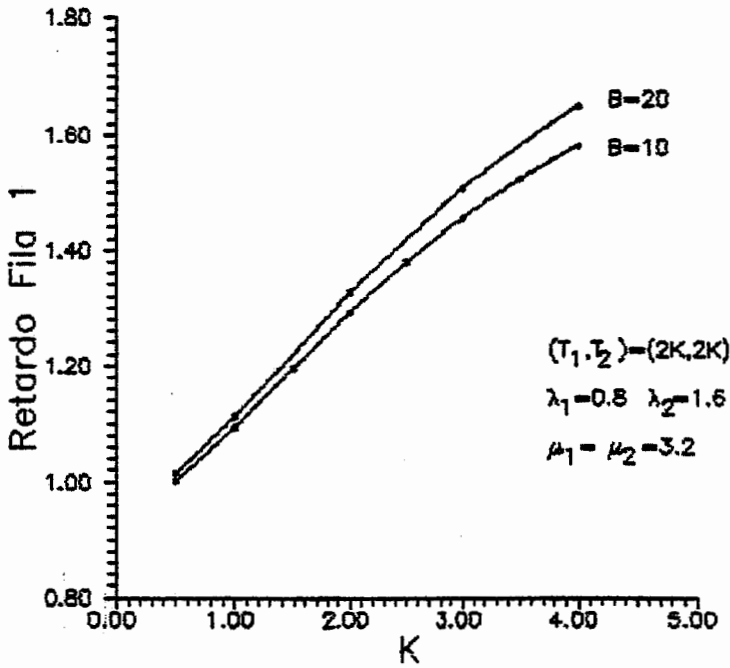


Figura 4.22: Retardo Fila 1 X (T_1, T_2) ($T_1 = T_2$ e $\lambda_2 = 2\lambda_1$)

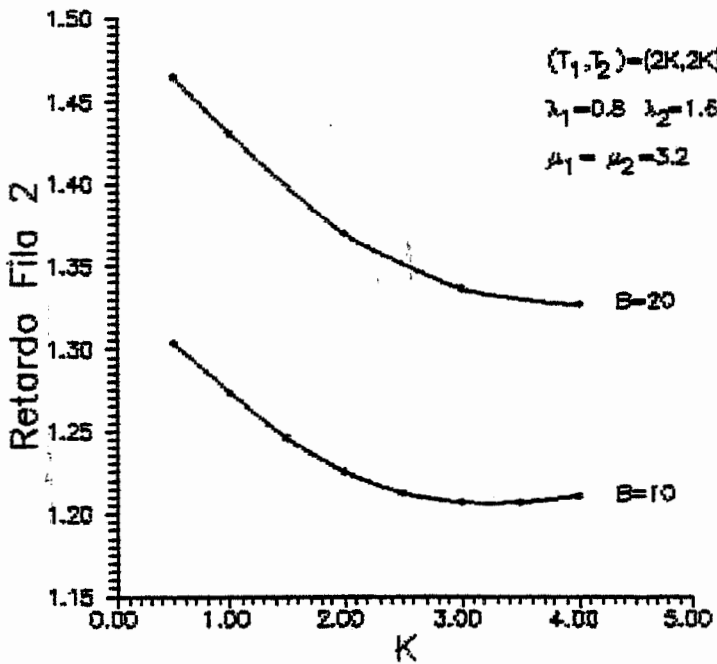


Figura 4.23: Retardo Fila 2 X (T_1, T_2) ($T_1 = T_2$ e $\lambda_2 = 2\lambda_1$)

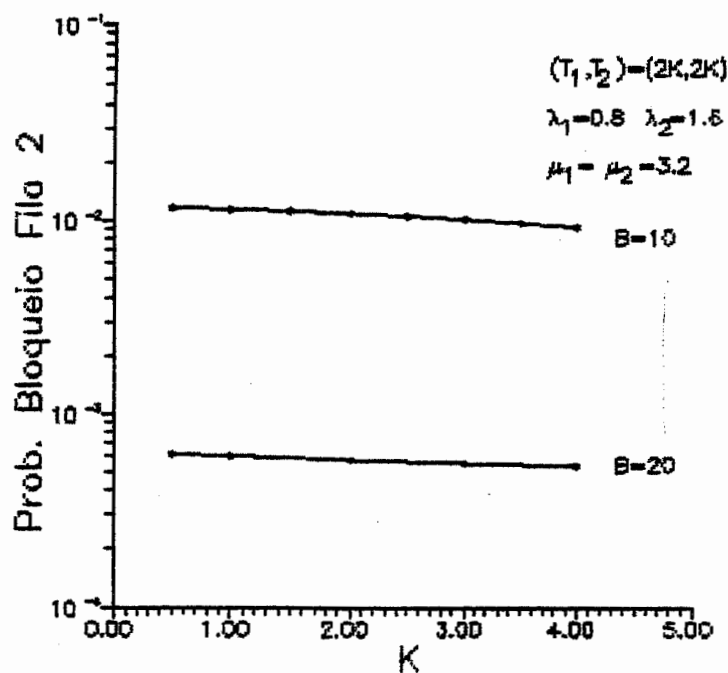


Figura 4.24: Prob. Bloqueio Fila 2 X (T_1, T_2) ($T_1 = T_2$ e $\lambda_2 = 2\lambda_1$)

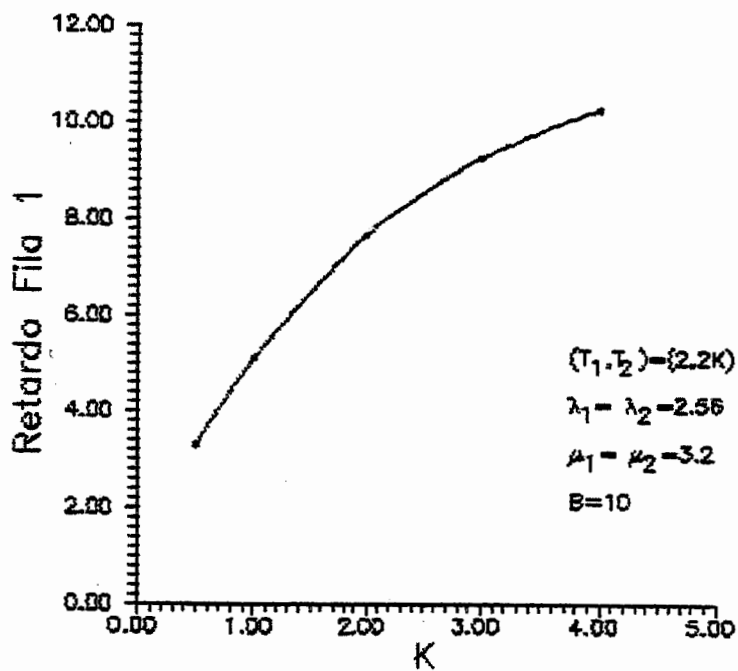


Figura 4.25: Retardo Fila 1 X T_2

filas no retardo e na probabilidade de bloqueio, quando variamos (T_1, T_2) . Vemos na figura 4.22 que, como a fila 1 tem baixa utilização, o aumento no tamanho do *buffer* pouco influi no seu retardo. Note que, estamos aumentando o *buffer* para as duas filas igualmente. Daí, a pequena variação que acontece quando dobramos o valor de B é devido, mais à influência da fila 2, do que propriamente da fila 1. Na figura 4.23, vemos que o retardo na fila 2 é mais sensível ao tamanho do *buffer* B , uma vez que esta fila é a mais utilizada. Note na figura 4.24, que a probabilidade de bloqueio na fila 2 é bastante sensível ao tamanho do *buffer* e é drasticamente reduzida ao dobrarmos B . Agora, consideremos um sistema com taxas de chegada e de serviço idênticas para as duas filas, onde analisaremos o retardo de uma fila, ao aumentarmos somente a quota de serviço da outra fila. A quota da fila em estudo é mantida fixa. Na figura 4.25, a curva do retardo da fila 1 é apresentada contra a variação de T_2 . Seja $T_2 = 2$ como base para a variação de T_2 . Por sua vez, T_1 é fixo e igual a 2. Como os demais gráficos desta seção, no eixo horizontal está indicado o fator multiplicador k , de forma que, o ponto K no eixo representa o par $(T_1, T_2) = (2, 2k)$. À medida que T_2 cresce, mais capacidade é dada para fila 2. Daí então, a fila 1 passa a experimentar um retardo médio maior. O retardo da fila 1 tende a um limite superior, devido ao fato de que não adianta aumentar T_2 indefinidamente, porque a fila 2 sofrerá absorção antes da quota T_2 expirar. Conforme vimos, supondo que a fila 2 esteja com seu *buffer* lotado, ela já está relativamente folgada para transmitir com $T_2 = 6$. Portanto, aumentos em T_2 serão inúteis, pois frequentemente haverá absorções.

A seguir, veremos a influência da variação de apenas uma das quotas sobre o retardo e a probabilidade de bloqueio de uma fila, quando aumentamos a utilização ρ (taxa de chegada de pacotes) da outra fila. Na figura 4.26, temos a curva do retardo na fila 2 frente a utilização ρ_1 da fila 1, para duas configurações de (T_1, T_2) . O retardo é diretamente proporcional a ρ_1 , pois com o aumento de ρ_1 , há mais pacotes na fila 1 para servir, e daí demora mais para o *token* retornar à fila 2. Ao aumentarmos T_1 , o retardo também aumenta, pois damos mais capacidade para a fila 1, em detrimento da fila 2. Para baixas taxas de chegada de pacotes, a variação de T_1 é pouco influente, porque não há muitos pacotes a transmitir em um ciclo, sendo portanto inútil aumentarmos T_1 . Note que, na curva para $(T_1, T_2) = (4, 4)$, o retardo alcançará um valor limitante superior, não devido ao *buffer* estar lotado, mas às custas da propriedade do esquema de alocação (T_1, T_2) que protege um tráfego quando há sobrecarga do outro tráfego. A probabilidade de bloqueio na fila 1, frente à variação da utilização ρ_2 da fila 2, é analisada na figura 4.27, também para os pares $(T_1, T_2) = (4, 4)$ e $(T_1, T_2) = (8, 4)$. A probabilidade cresce com o aumento de ρ_2 , pelo mesmo motivo do retardo na figura 4.26. Quando damos mais capacidade à fila 1, a probabilidade cai, pois os pacotes são mais rapidamente servidos liberando assim espaço no *buffer*. Aqui, para baixas taxas de chegada de pacotes à fila 2, a variação de T_1 pouco influi na probabilidade de bloqueio na fila 1, pois nestas situações, a fila 1 já usa quase toda capacidade do canal, devido a frequentes esvaziamentos do canal em um ciclo na fila 2. Então, não adianta muito aumentar a quota T_1 da fila 1.

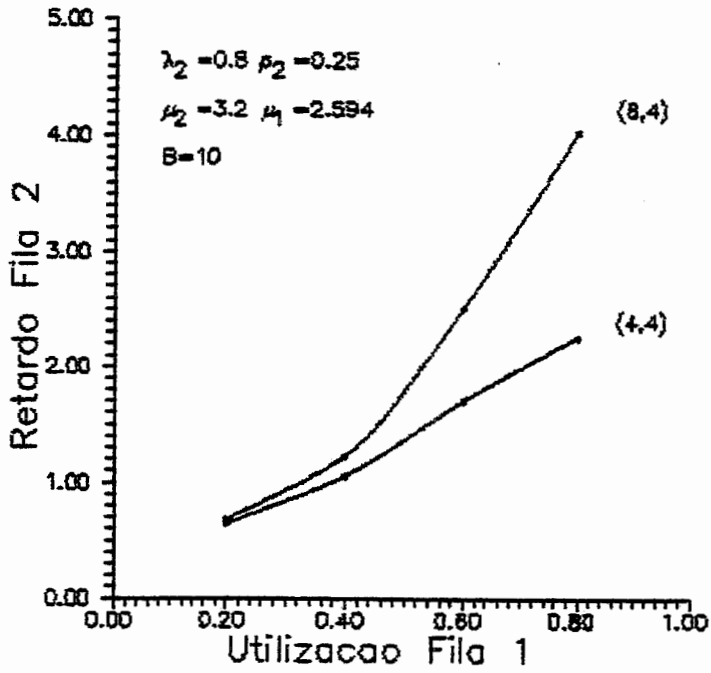


Figura 4.26: Retardo Fila 2 X ρ_1

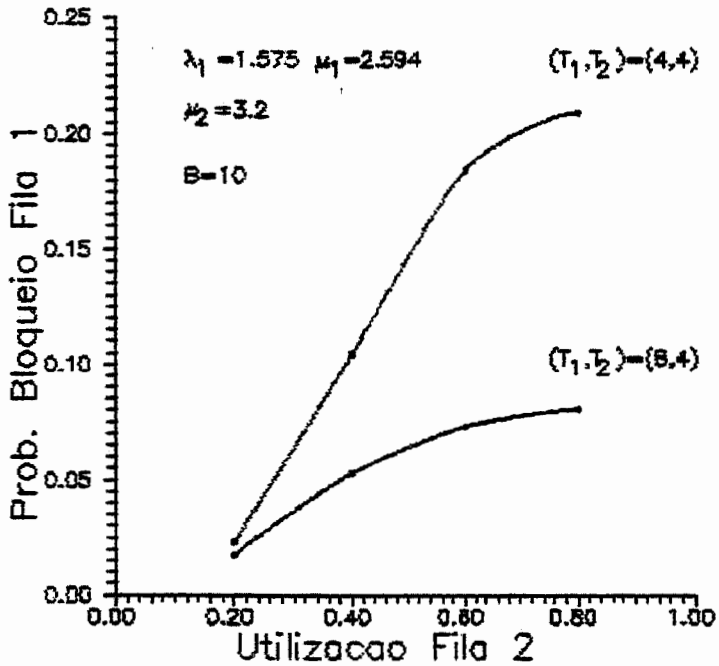


Figura 4.27: Prob. Bloqueio Fila 1 X ρ_2

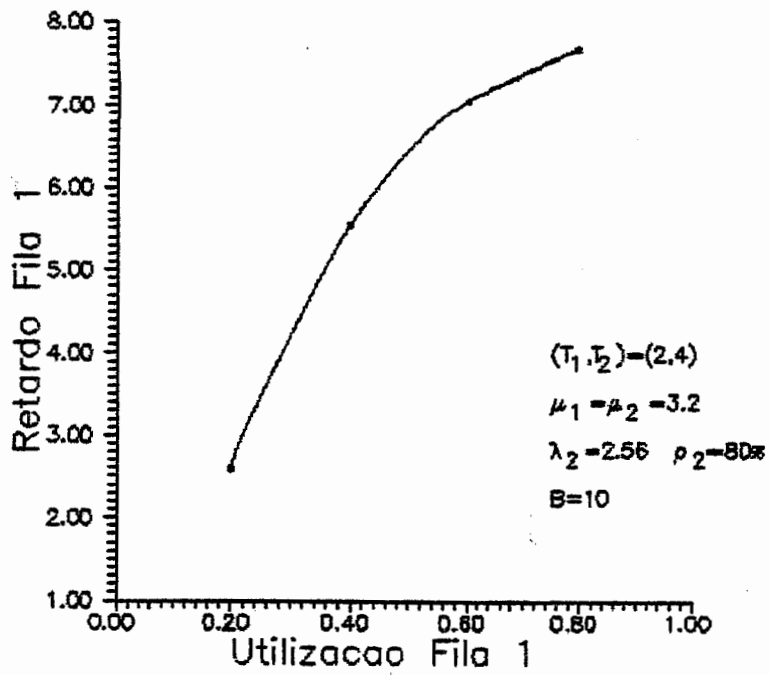


Figura 4.28: Retardo Fila 1 X ρ_1

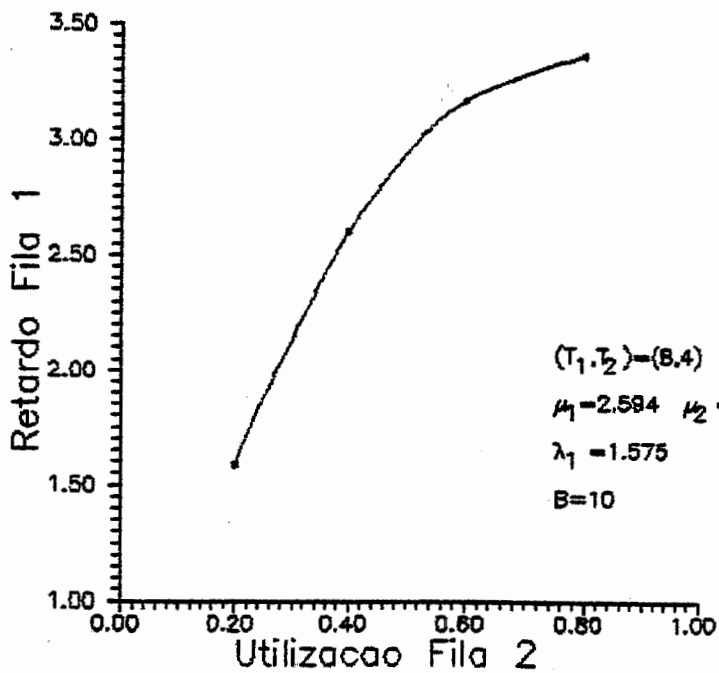


Figura 4.29: Retardo Fila 1 X ρ_2

4.4 Influência da Utilização de uma Fila

Nesta seção, estudaremos alguns sistemas (T_1, T_2) , objetivando analisar a sensibilidade das medidas retardo e probabilidade de bloqueio de uma fila, a variações na utilização de uma das filas. Na análise a seguir, manteremos a taxa de serviço constante. Portanto, para variarmos a utilização de uma fila, basta variarmos a taxa de chegada de pacotes naquela fila, ou seja, a sua carga ofertada de pacotes. Para a fila analisada, a sua utilização ρ , é mantida constante. Inicialmente, veremos a influência da utilização de uma fila, sobre o retardo e a probabilidade de bloqueio desta mesma fila. Na figura 4.28, vemos que o retardo da fila 1 cresce com o aumento da utilização da fila 1. Este efeito acontece, pois o número de pacotes na fila 1 para serem servidos aumenta. Embora cresça, este retardo tem um valor limitante superior, que é imposto pelo tamanho finito do *buffer* $B = 10$. Quando o *buffer* se torna lotado, não adianta mais injetar pacotes, pois estes serão descartados e não interferirão no retardo dos pacotes já armazenados no *buffer*. Note que, quando a utilização da fila 1 é de 80% ($\lambda_1 = 2.56$), a fila 1 está, em média, com aproximadamente 9 pacotes, ou seja, está quase lotada. No sistema da figura 4.4, temos o mesmo fenômeno. Para $B = 5$, o efeito do tamanho do *buffer* já é sentido com $\lambda_2 = 2.56$ ($\rho_2 = 80\%$), pois o *buffer* é pequeno. Para as outras curvas, com B igual a 10, 15 e 20, este efeito é sentido mais tarde, com maiores taxas de chegada de pacotes. Considerando o mesmo sistema da figura 4.28, é apresentado, na figura 4.30 o comportamento da probabilidade de bloqueio na fila 1, frente a variações na sua utilização. Como era esperado, a probabilidade é diretamente proporcional à utilização da fila 1 pois, com mais pacotes chegando à fila, é mais provável que o seu *buffer* esteja lotado com B pacotes.

Agora analisaremos o comportamento das medidas retardo e probabilidade de bloqueio de uma fila, frente a variações na utilização (carga ofertada de pacotes) da outra fila. Na figura 4.29, temos o comportamento do retardo da fila 1, quando incrementamos a utilização da fila 2, aumentando λ_2 . O retardo cresce, pois há mais pacotes para servir na fila 2, ocupando mais o canal e fazendo com que o *token* demore mais a retornar. Embora crescente, o retardo da fila 1 tem um limitante superior. Diferente do gráfico da figura 4.28, este limite não é imposto pelo tamanho do *buffer* da fila. Na realidade, ele é fruto do esquema de alocação do protocolo (T_1, T_2) , que protege um tráfego quando há sobrecarga do outro. Note que no ponto correspondente a $\rho_2 = 0.8$, onde a fila 2 já está razoavelmente sobrecarregada, o número médio de pacotes na fila 1 é aproximadamente 5, portanto bem abaixo do limite do *buffer*, que é igual a 10. A mesma curva é mostrada na figura 4.31, juntamente com as curvas para tamanho de *buffer* igual a 15 e 20. Na figura 4.27, temos a curva da probabilidade de bloqueio para este sistema.

Para o mesmo sistema, porém com $B = 15$, apresentamos, na figura 4.32, as curvas do retardo nas filas 1 e 2, contra a utilização da fila 2. O objetivo deste gráfico é analisar conjuntamente o efeito sofrido pelo retardo, em uma fila, quando se aumenta a utilização da própria fila, ou da outra fila. Neste sistema (T_1, T_2) , ao aumentarmos a utilização da fila 2, o retardo de ambas as filas cresce.

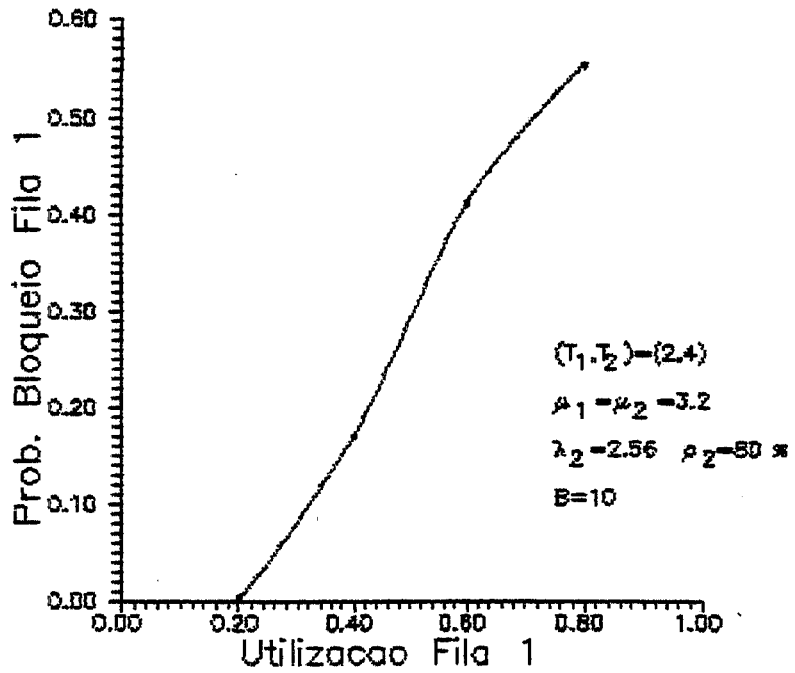


Figura 4.30: Prob. Bloqueio Fila 1 X ρ_1

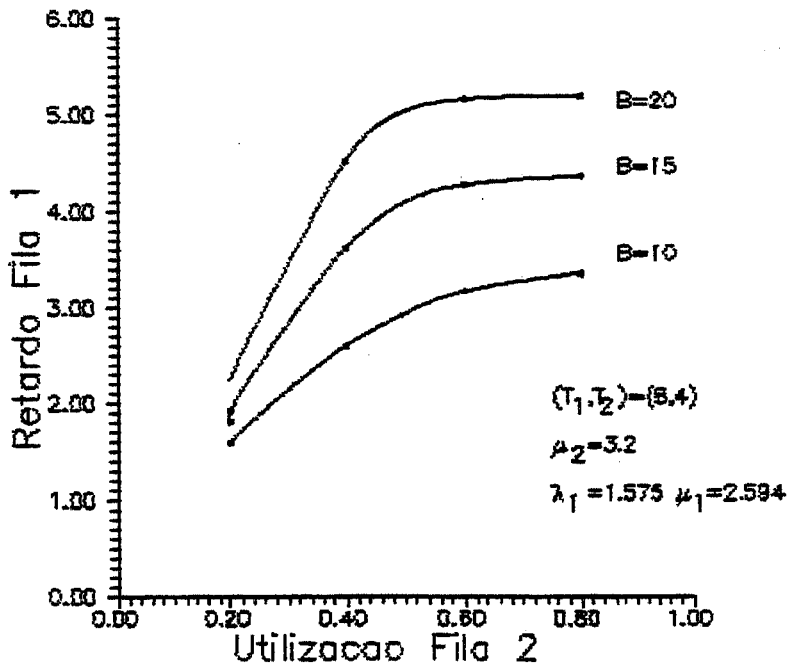


Figura 4.31: Retardo Fila 1 X ρ_2 (Influência de B)

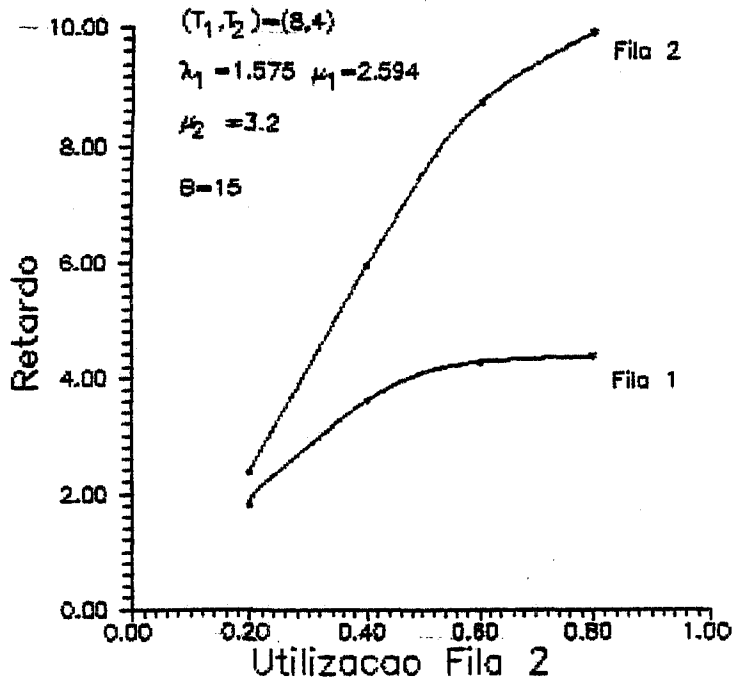


Figura 4.32: Retardo X ρ_2

Cresce na fila 2, pois há mais pacotes para servir, e cresce na fila 1, pois o servidor (*token*) demora mais a voltar para ela, devido a fila 2 passar a utilizar mais a sua quota T_2 . Conforme já vimos, o retardo para a fila 2 é limitado superiormente pelo tamanho finito do *buffer*, pois a fila tende a se encher de pacotes e, uma vez lotada, os próximos pacotes a chegar são simplesmente rejeitados. Já para fila 1, o retardo é limitado superiormente, pela ação do protocolo (T_1, T_2) , que emprega um esquema de tempos de serviço limitados, protegendo o tráfego da fila 1, na presença de sobrecarga da fila 2. Portanto, o esquema garante uma largura de banda mínima para o tráfego da fila 1, limitando assim o retardo dos seus pacotes. É fácil ver que, quando $\rho_2 = 80\%$, o número médio de pacotes armazenados na fila 1 é aproximadamente 6, portanto bem abaixo do tamanho do *buffer* $B = 15$. Já na fila 2, este número médio é aproximadamente 15, ou seja, a fila se encontra com seu *buffer* praticamente lotado. Na figura 4.26, temos um outro sistema (T_1, T_2) onde temos a evolução do retardo de uma fila frente ao aumento da utilização da outra fila. Lá também, o retardo é limitado pela ação do protocolo (T_1, T_2) , que permite um uso mais justo da capacidade para transmissão. Para $(T_1, T_2) = (4, 4)$, a fila 2 não está lotada quando ρ_1 alcança 80%, mas sim com apenas 2 pacotes aproximadamente.

A seguir, vamos analisar a probabilidade de bloqueio de uma fila quando variamos a utilização da outra fila. Na figura 4.33, temos duas curvas da probabilidade de bloqueio da fila 2 contra a utilização ρ_1 da fila 1, para $(T_1, T_2) = (8, 2)$ e $(T_1, T_2) = (4, 4)$. Como podemos ver, para as duas configurações de (T_1, T_2) , a curva é ascendente para valores crescentes de ρ_1 . Como o número de pacotes a serem servidos na fila 1 aumenta, esta passa a utilizar mais o canal para transmissão, fazendo com que o *token* demore mais a retornar à fila 2. Daí, a probabilidade de bloqueio na fila 2 aumenta, pois lá existem mais pacotes armazenados no *buffer*

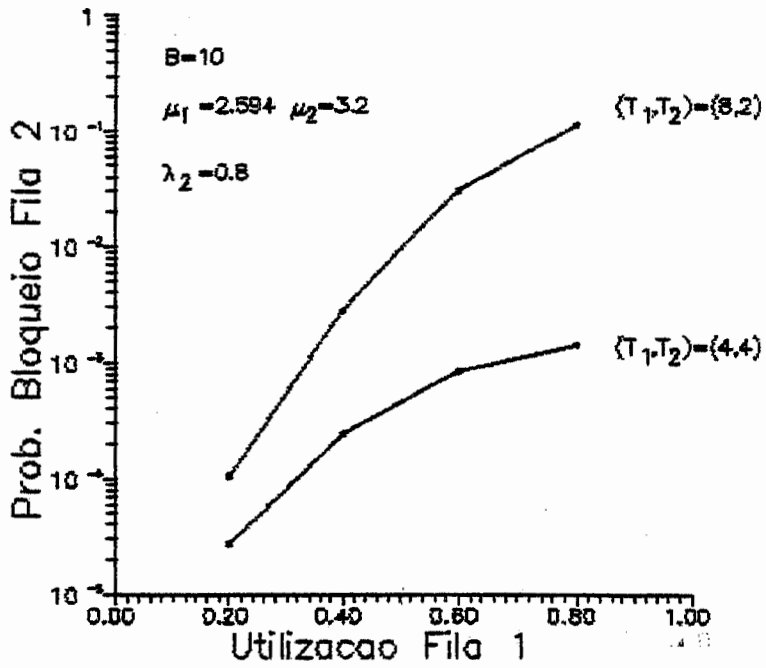


Figura 4.33: Prob. Bloqueio Fila 2 X ρ_1

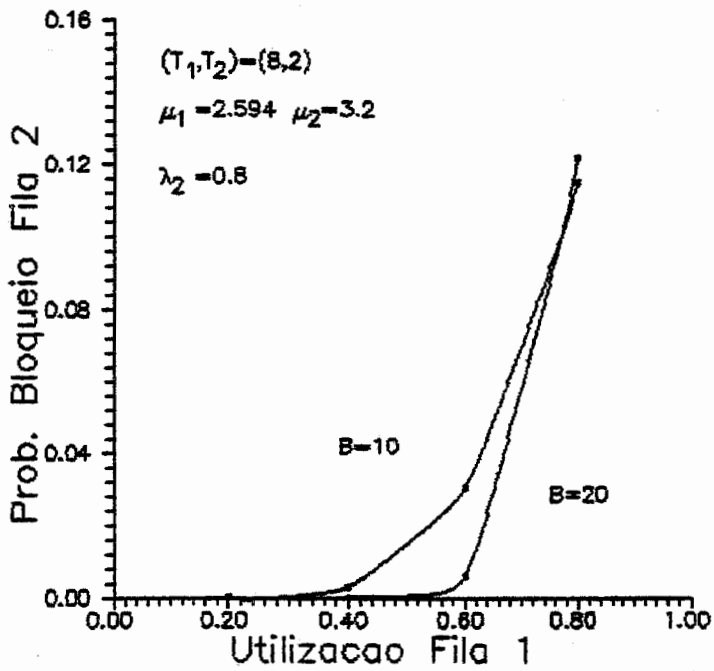


Figura 4.34: Prob. Bloqueio Fila 2 X ρ_1 ($(T_1, T_2) = (8, 2)$)

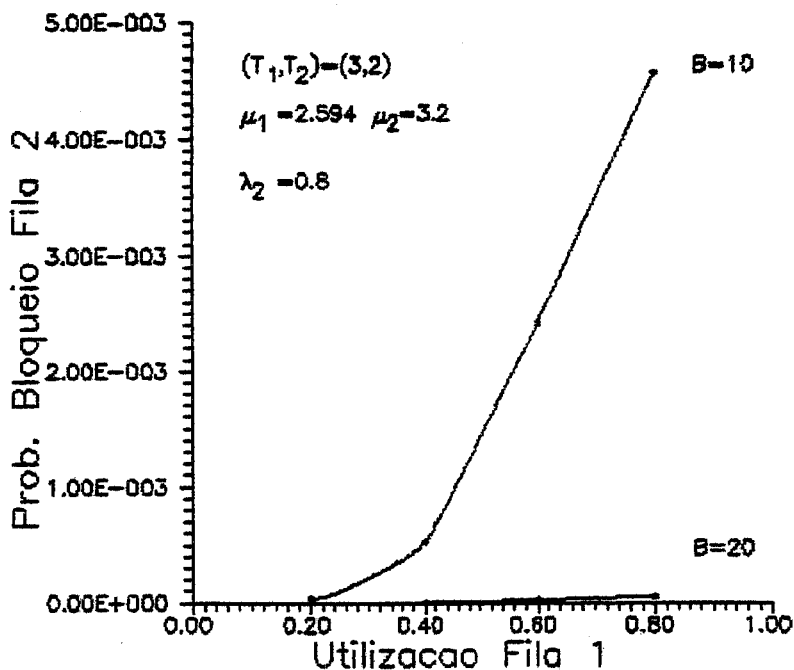


Figura 4.35: Prob. Bloqueio Fila 2 X ρ_1 ($(T_1, T_2) = (3, 2)$)

aguardando serem servidos. Como na figura 4.26, para $(T_1, T_2) = (4, 4)$, quando $\rho_1 = 80\%$, a fila 2 não está lotada. A probabilidade cai, ao passarmos de $(T_1, T_2) = (8, 2)$ para $(4, 4)$, pois tiramos capacidade da fila 1 e além disso, demos mais capacidade à fila 2. Portanto, a fila 2 servirá mais rapidamente os seus pacotes, liberando espaço no seu *buffer* para outros pacotes que chegarem. Na figura 4.34, confrontamos a curva para $(T_1, T_2) = (8, 2)$, com a curva da probabilidade de bloqueio de um sistema com os mesmos parâmetros, com exceção do tamanho do *buffer* B das filas, que é dobrado para 20. Neste gráfico, nos defrontamos com um fenômeno interessante, correspondente ao ponto de cruzamento da figura 4.34, para $\rho_1 \approx 80\%$. Com a utilização da fila 1 baixa ($\rho_1 < 80\%$), o número médio de pacotes na fila 1 é pequeno. Ao dobrarmos B , então estamos dando mais espaço para armazenar pacotes na fila 2, diminuindo assim a sua probabilidade de bloqueio. O aumento de B é pouco influente para a fila 1, devido a baixa utilização desta fila. Com a utilização ρ_1 baixa, os pacotes na fila 1 são servidos rapidamente, e a fila se esvazia logo sem dar tempo de mais pacotes chegarem. A fila 1 absorve rapidamente e não usa toda a sua quota $T_1 = 8$. Quando dobramos B , isto se mantém, embora o mini-ciclo médio de serviço para a fila 1 aumente um pouco, pois o mini-ciclo é maior para tamanhos de *buffer* maiores, já que o número médio de pacotes na fila é maior. Com a sua utilização alta, a fila 1 está com um número médio de pacotes alto, ou até mesmo lotada no pior caso. Vamos supor aqui, sem perda de generalidade, que a fila 1 está com seu *buffer* lotado. Lembremos que todas as filas têm o mesmo tamanho de *buffer* B . Portanto, embora dobramos B para a fila 2, o que nos levaria a crer que a sua probabilidade de bloqueio diminuiria, também dobramos B para a fila 1, simultaneamente. Com $B = 10$, a fila 1 leva aproximadamente 4 ms para servir todos os 10 pacotes armazenados em seu *buffer* lotado. Se B é dobrado para 20, a fila 1 usa quase toda a sua quota $T_1 = 8$ ms para servir os 20 pacotes que se encontram no seu *buffer*, também assumido lotado. Então, o aumento do tamanho

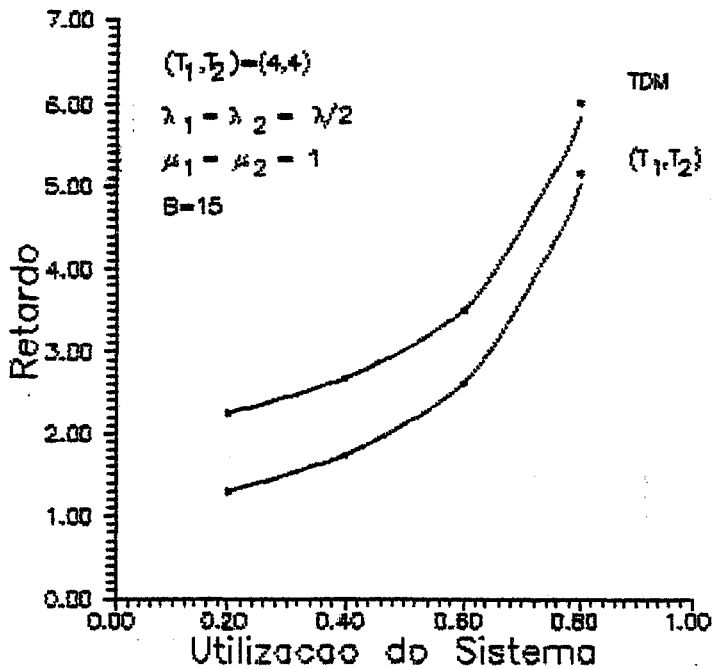


Figura 4.36: (T_1, T_2) X TDM (Retardo)

do *buffer* para a fila 1, levou a um aumento no seu mini-ciclo médio de serviço, o que fez a probabilidade de bloqueio da fila 2 aumentar, pois o *token* leva mais tempo para voltar. O aumento de B para a fila 2 é pouco relevante, pois a sua utilização ρ_2 é relativamente baixa ($\rho_2 = 0.25$).

Já na figura 4.35, a mesma situação é analisada para o par de quotas $(T_1, T_2) = (3, 2)$. Agora, com $T_1 = 3$, aquele fenômeno de inversão não mais acontece. Supondo de novo a fila 1 lotada, para uma alta utilização da fila 1, a quota de serviço T_1 , na média, sempre expira (*time-out*), antes que todos os pacotes tenham sido transmitidos. Isto acontece pois, como já vimos, ela leva aproximadamente 4 ms para servir todos os 10 pacotes em seu *buffer* lotado. Para $B = 20$, a quota de serviço T_1 também expira antes. Logo, o aumento do tamanho do *buffer* B não aumenta o mini-ciclo de serviço da fila 1. O que explica o gráfico da figura 4.35, é o fato de que, quando dobramos B , embora seja para as duas filas, estamos aumentando principalmente para a fila 2, e a provendo de mais espaço para armazenar os pacotes. Isto leva, obviamente, a uma queda na sua probabilidade de bloqueio. Note também, que a probabilidade na fila 2 para $(T_1, T_2) = (3, 2)$ é bem menor que para $(T_1, T_2) = (8, 2)$, pois o mini-ciclo médio de serviço da fila 1 neste último, é bem maior, fazendo com que a fila 2 se enche mais e fique portanto com menos espaço livre para novos pacotes. Na figura 4.27, como já vimos, temos o gráfico da probabilidade de bloqueio da fila 1 contra a utilização ρ_2 da fila 2, em um sistema com $B = 10$, para as configurações $(T_1, T_2) = (4, 4)$ e $(8, 4)$.

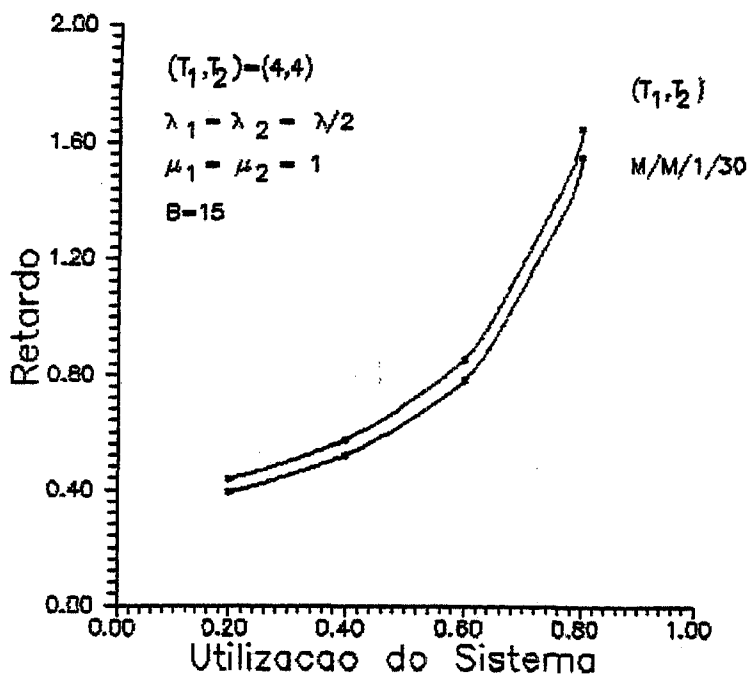


Figura 4.37: (T_1, T_2) X M/M/1/K (Retardo)

4.5 Comparando com o Esquema TDM

Nesta seção, fazemos um estudo comparativo entre o esquema de multiplexação (T_1, T_2) e o esquema de multiplexação por divisão do tempo TDM. A modelagem para o esquema TDM utilizada foi obtida em [13] para duas estações (filas). Neste modelo, o eixo do tempo é dividido em quadros (*frames*) de dois *slots*, com um *slot* dedicado a cada estação. Cada *slot* tem duração de uma unidade de tempo (aqui, 1 milissegundo) e nele, somente um pacote pode ser transmitido. O tempo de serviço para um pacote é portanto igual a 1 ms. Na figura 4.36, as curvas de retardo para os esquemas (T_1, T_2) e TDM são confrontadas em um gráfico de retardo versus a utilização ρ total. O modelo para TDM assume que as filas têm tamanho de *buffers* infinitos, embora, para o sistema (T_1, T_2) considerado, o tamanho do *buffer* assumido é igual a 15. A comparação entre os esquemas é razoável pois a probabilidade de bloqueio no (T_1, T_2) é praticamente zero ($< 0.2\%$). Podemos ver que o esquema (T_1, T_2) é melhor que o TDM, pois neste último, há perda no uso da capacidade do canal, quando uma fila está vazia e o servidor mesmo assim fica dedicado a ela por uma quota de tempo pré-fixada. Note portanto que, quanto menor for a utilização ρ , maior é a vantagem do esquema (T_1, T_2) sobre o TDM. Por outro lado, quanto maior for ρ , menor é a diferença, quanto ao retardo, entre os esquemas. No entanto, embora não mostrado no gráfico, para uma utilização total bastante próxima a 1, o esquema TDM apresenta um retardo menor, ou seja, ele é melhor que o esquema (T_1, T_2) . Com ρ próximo a 1, as filas estão quase lotadas, e o servidor serve cada fila por toda a sua respectiva quota de tempo, tanto no TDM quanto no (T_1, T_2) . Daí, o retardo para TDM é menor, pois no modelo desse esquema, não são considerados os intervalos de *switch-over*, que existem no esquema (T_1, T_2) .

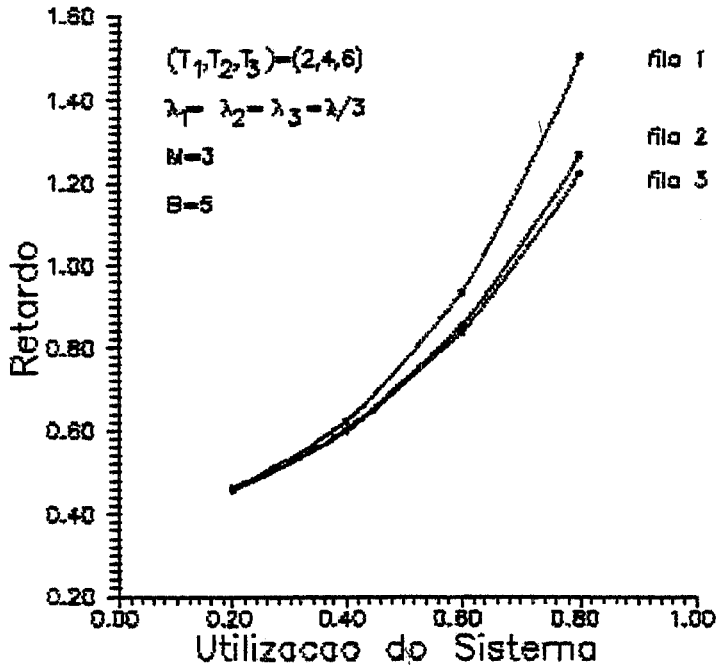


Figura 4.38: Retardo $\times \rho$ ($(T_1, T_2, T_3) = (2, 4, 6)$)

4.6 Comparando com uma Fila M/M/1/K

Nesta seção, vamos comparar o esquema de alocação (T_1, T_2) com o modelo de uma fila M/M/1/K, quanto ao comportamento das medidas de desempenho, retardo e probabilidade de bloqueio. Seja o esquema $(T_1, T_2) = (4, 4)$, com tamanho de *buffer* $B = 15$. A comparação é feita com uma fila M/M/1/K, pois os *buffers* no esquema (T_1, T_2) são, como já vimos, limitados. Aqui, para fazermos uma comparação coerente, façamos $K = 2B$, de forma que o espaço total de armazenamento dos dois sistemas seja o mesmo. Na figura 4.37, temos o retardo no esquema (T_1, T_2) , confrontado com o retardo de uma fila M/M/1/K, frente à utilização do sistema. A fila M/M/1/K tem um desempenho melhor, em termo de retardo, pois nela não há intervalos de *switch-over*, e os pacotes na fila são servidos imediatamente um após o outro. De fato, um pacote só tem que esperar pelo serviço dos pacotes que chegaram à fila antes dele. Entretanto, a diferença é muito pequena, o que mostra a eficiência do esquema (T_1, T_2) . A desvantagem do esquema de uma fila M/M/1/K surge quando aumenta a carga ofertada de pacotes de um dos tipos de tráfego. Neste caso, o outro tipo de tráfego é muito penalizado, quanto ao retardo, pois não há a garantia de uma capacidade mínima para transmissão, como no esquema (T_1, T_2) .

4.7 Comparando com o Esquema HOL (*Head-Of-the-Line*)

Nesta seção, o esquema (T_1, T_2, T_3) , com três filas, é comparado com o esquema de prioridades do sistema HOL. A modelagem do sistema HOL, obtida de [15],

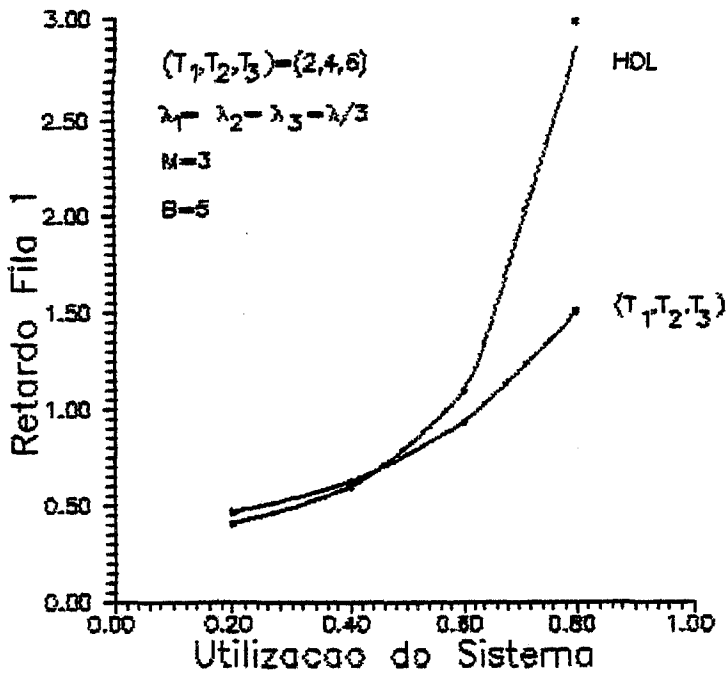


Figura 4.39: (T_1, T_2, T_3) X HOL (Retardo Tráfego Prior. Baixa)

estabelece um esquema de enfileiramento em uma fila única com três grupos de prioridade ($p=1, 2$ e 3), onde quanto maior o valor de p , maior é a prioridade. Os pacotes que chegam à fila pertencem a cada um destes grupos. O tamanho do *buffer* da fila é assumido infinito. No modelo, a disciplina de serviço é não preemptiva, e a taxa de serviço de pacotes μ é igual a 3.2. No esquema (T_1, T_2, T_3) estudado, as três filas têm tamanho de *buffer* igual a 5. Apesar de estarmos comparando este esquema de *buffer* finito, com outro de *buffer* infinito, a comparação entre os dois é razoável, pois a probabilidade de bloqueio no esquema (T_1, T_2, T_3) é bastante pequena e pode ser desprezada. A taxa de serviço é igual a 3.2, e é a mesma para as três filas ($\mu_1 = \mu_2 = \mu_3 = \mu$). A configuração $(T_1, T_2, T_3) = (2, 4, 6)$ foi escolhida para refletir, neste esquema, prioridades estáticas entre os pacotes das diferentes filas, como no caso do HOL. Portanto, quanto maior for a quota de serviço da fila, maior será a sua prioridade no uso da capacidade do canal. Na figura 4.38, temos as curvas de retardo para as três filas, contra a utilização total ρ do sistema. Quanto maior for a quota de tempo de serviço de uma fila, mais tempo de acesso ao canal ela terá, portanto, menor será o retardo de seus pacotes. Na figura 4.39, temos um gráfico comparativo entre os retardos dos tráfegos menos prioritários, no esquema (T_1, T_2, T_3) e no sistema HOL. Como já vimos, o tráfego de prioridade inferior no esquema (T_1, T_2, T_3) é o da fila 1, com $T_1 = 2$ e, no sistema HOL, é o tráfego do grupo de prioridade $p = 1$. Para uma utilização total ρ baixa, o sistema HOL é melhor. Isto acontece pois no HOL não existem intervalos de *switch-over* como no esquema (T_1, T_2, T_3) . Para $\rho = 0.2$, a diferença entre os dois retardos é aproximadamente igual à metade do somatório dos três intervalos de *switch-over*, que é o tempo médio que um pacote da fila de menor prioridade tem que esperar, para ser servido a uma baixa utilização do sistema. À medida que a utilização cresce, o esquema (T_1, T_2, T_3) se torna melhor, pois garante uma largura de banda mínima para transmissão do tráfego menos prioritário. Lembre que, no sistema

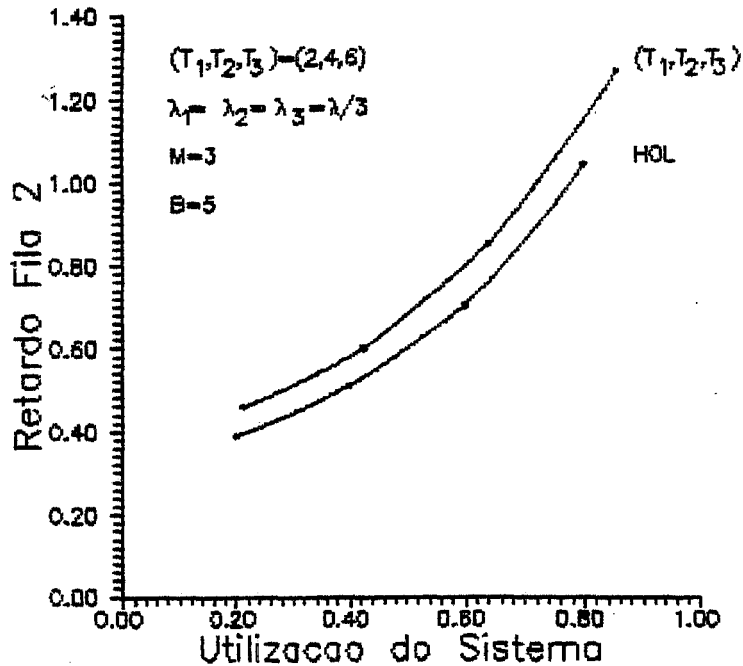


Figura 4.40: (T_1, T_2, T_3) X HOL (Retardo Tráfego Prior. Média)

HOL, um pacote de prioridade inferior ($p=1$) sempre tem que esperar que todos os outros pacotes mais prioritários ($p=2$ e 3) tenham sido servidos, para poder ser servido. Portanto, no caso de sobrecarga de pacotes mais prioritários, o tráfego menos prioritário experimentará um alto retardo médio.

Já nas figuras 4.40 e 4.41, os retardos para os tráfegos de prioridade intermediária e superior, respectivamente, são comparados aos tráfegos de prioridades correspondentes no sistema HOL. Note nestas figuras, que o sistema HOL é sempre melhor que o esquema (T_1, T_2, T_3) . Aqui, este comportamento é explicado pelo fato de que, no sistema HOL, os pacotes do tráfego de prioridade intermediária ($p=2$), ou superior ($p=3$), não têm que esperar pelos pacotes do outro tráfego menos prioritário ($p=1$). No entanto, no esquema (T_1, T_2, T_3) , um pacote da fila 2 ou 3 pode, eventualmente, ter que esperar pelo término da quota de serviço T_1 . Na figura 4.40, o sistema HOL é melhor, pois o tráfego de prioridade intermediária só sofre influência, além dele próprio, do tráfego de prioridade superior. No esquema (T_1, T_2, T_3) , a fila 1, além da fila 3, também contribui para o retardo da fila 2. Na figura 4.41, a diferença entre os retardos é maior ainda, pois o tráfego mais prioritário é tratado no sistema HOL, como se não existisse nenhum outro tipo de tráfego concorrendo pelo uso do canal. Finalmente, na figura 4.42, vemos o gráfico comparativo do retardo no esquema (T_1, T_2, T_3) e no sistema HOL, apresentado conjuntamente para os três tipos de tráfego.

Fazendo $(T_1, T_2, T_3) = (5, 5.5, 6)$, e mantendo os demais parâmetros iguais, podemos observar na figura 4.43, que os retardos das três filas são praticamente iguais. Ao fazermos isto, estamos dando praticamente a mesma quota de tempo de serviço (ou aqui, prioridade de uso do canal) para as três filas, o que faz com que elas tenham, em média, o mesmo comportamento. Se agora fizermos

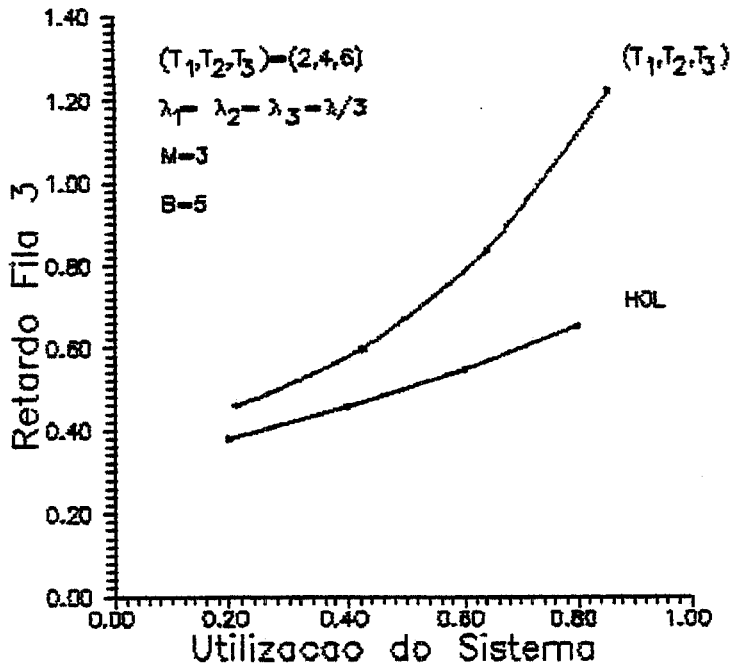


Figura 4.41: (T_1, T_2, T_3) X HOL (Retardo Tráfego Prior. Alta)

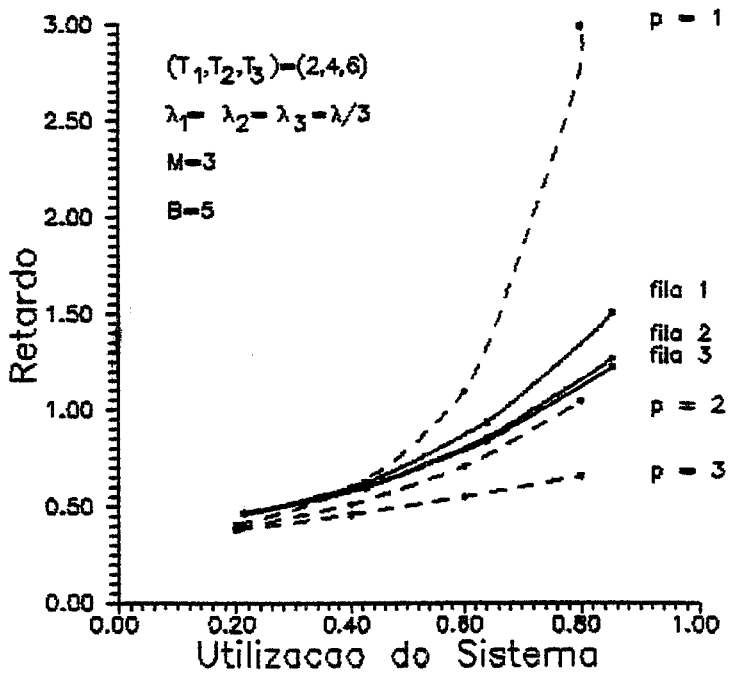


Figura 4.42: (T_1, T_2, T_3) X HOL (Retardo)

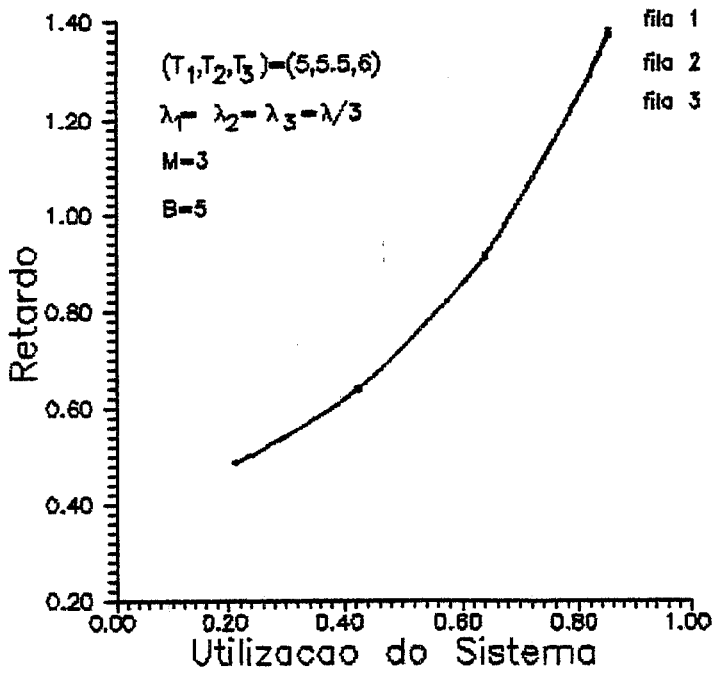


Figura 4.43: Retardo X ρ $((T_1, T_2, T_3) = (5, 5.5, 6))$

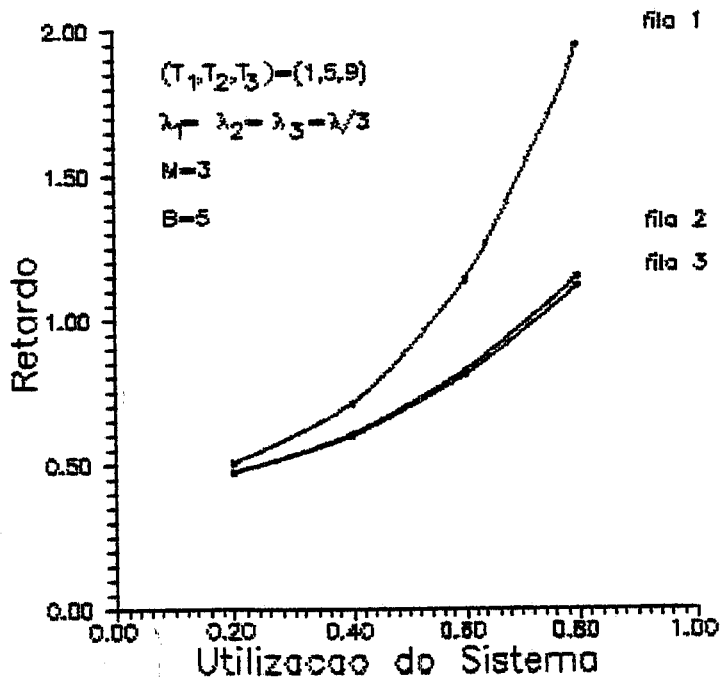


Figura 4.44: Retardo X ρ $((T_1, T_2, T_3) = (1, 5, 9))$

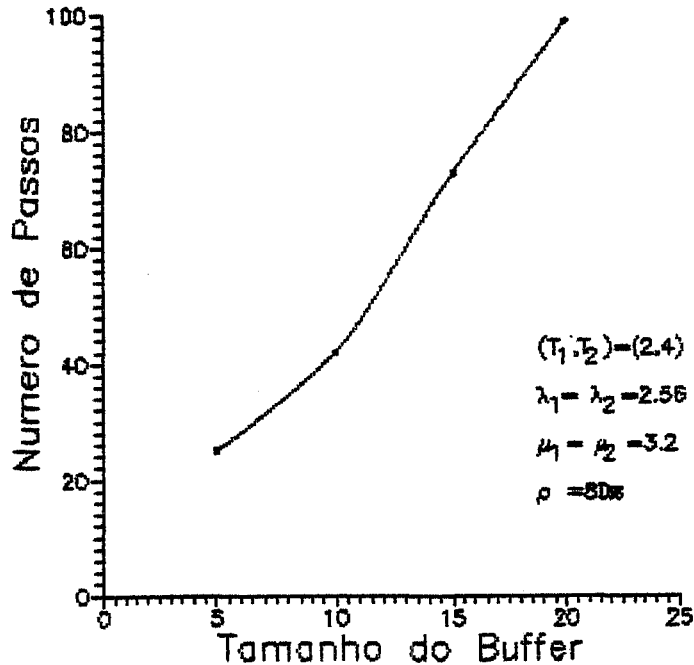


Figura 4.45: Velocidade de Convergência X B

$(T_1, T_2, T_3) = (1, 5, 9)$, vemos na figura 4.44, que os retardos ficam mais diferenciados. Esta diferenciação é mais sentida para a fila 1. Portanto, podemos ver nas figuras 4.43, 4.38 e 4.44 que, quanto mais espaçados forem T_1, T_2 e T_3 , mais diferenciados serão os retardos das filas 1, 2 e 3. Daqui, podemos concluir que o esquema (T_1, T_2, T_3) é mais flexível e variável que o sistema HOL, pois pode atender conjuntamente, diversos requisitos de retardo, exigidos por diferentes tipos de tráfego. Para tal, basta que escolhamos convenientemente os valores das quotas (T_1, T_2, T_3) . O esquema do sistema HOL não permite esta flexibilidade, ao atribuímos classes de prioridades estáticas aos vários tipos de tráfego.

4.8 Analisando a Convergência

Por último, analisaremos a velocidade do método para determinar o vetor β , em estado estacionário. Quanto maior o tamanho do *buffer* B das filas, mais estados tem o modelo, e daí a convergência é mais lenta. Veja na figura 4.45 que o número de passos até a convergência cresce com B , para um sistema com 2 filas, com $(T_1, T_2) = (2, 4)$. Vemos na figura 4.46, a curva do número de passos até a convergência contra o valor das quotas T_1 e T_2 . Neste sistema, $B = 10$ e o par (T_1, T_2) base é igual a $(2, 4)$. Neste caso, ao incrementarmos o par (T_1, T_2) , a convergência é mais rápida, ou seja, é preciso menos passos até se alcançar a convergência.

No geral, verificamos que o método tem uma convergência rápida, e atendeu razoavelmente nossas expectativas em custo de processamento. No entanto, algumas mudanças podem ser feitas no processo de obtenção, e no teste de convergência do vetor β , em estado estacionário, visando aumentar a velocidade de

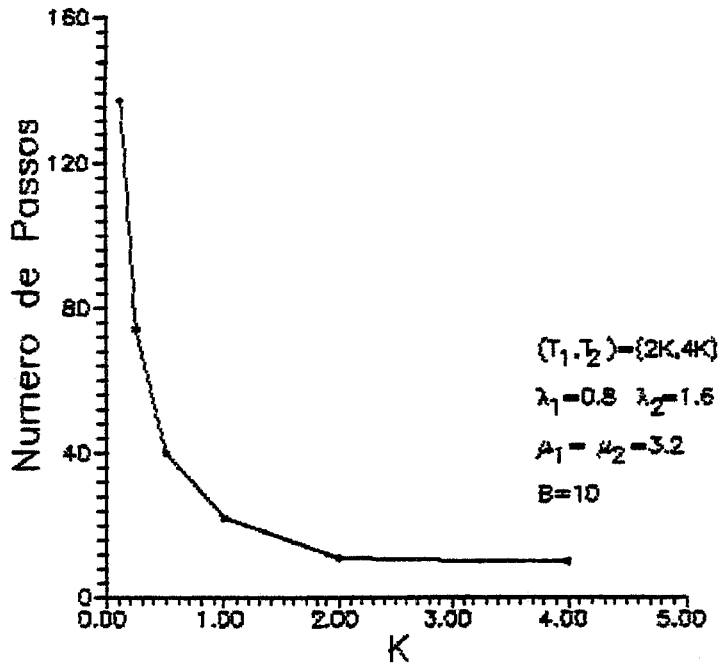


Figura 4.46: Velocidade de Convergência $X(T_1, T_2)$

convergência do método.

Capítulo 5

Conclusão

Neste trabalho, estudamos o método de modelagem e análise proposto em [1] e desenvolvemos uma ferramenta para implementar as suas equações visando avaliar sistemas de *polling* com tempos de serviço limitados. O método pode ser usado para modelar e analisar vários sistemas reais, dentre eles, o multiplexador (T_1, T_2) , o mecanismo para transmissão de tráfegos síncronos (tempo-real) e assíncronos restritivos do protocolo FDDI e as subestações virtuais mais prioritárias (classe 6) especificadas no padrão IEEE 802.4 *Token Bus*.

Inicialmente, introduzimos dois conceitos fundamentais para o desenvolvimento da teoria: randomização e cadeia de Markov com recompensas. Em seguida, estudamos o método de modelagem propriamente dito, onde foi identificada uma cadeia de Markov embutida e depois, utilizando conceitos de processos regenerativos, mostramos as equações para o cálculo de algumas medidas de desempenho de interesse. Também indicamos os aspectos computacionais da implementação do método, avaliando custos de processamento e armazenamento e identificando problemas de *underflow/overflow*. Finalmente, estudamos o desempenho de diversos sistemas (T_1, T_2) e (T_1, T_2, T_3) .

Na análise de desempenho, sistemas foram estudados e alguns também comparados ao esquema de multiplexação TDM, a uma fila M/M/1/K, e ao esquema de prioridades do sistema HOL. Nesta análise, estudamos a sensibilidade das medidas de desempenho, retardo médio por pacote e probabilidade de bloqueio nas filas, quando o sistema é sujeito a variações de carga ofertada de pacotes, tamanho do *buffer* das filas, duração dos intervalos de *switch-over* e das quotas de tempo de serviço.

Para os sistemas de *polling* com 2 filas, podemos concluir que o esquema de alocação (T_1, T_2) é justo na alocação da largura de banda e, garantindo uma capacidade mínima de transmissão para os tráfegos, protege um tráfego na eventualidade de sobrecarga do outro. Foi concluído também que o tamanho do *buffer* de uma fila pode influenciar bastante o retardo e a probabilidade de bloqueio dos seus pacotes. No projeto de um sistema, para se alcançar baixas probabilidades

de bloqueio, filas com *buffers* grandes devem ser usadas. Portanto, podemos encontrar um tamanho de *buffer* que satisfaça os requisitos de probabilidade de bloqueio para um sistema. A duração dos intervalos de *switch-over* e das quotas de serviço (T_1, T_2) também têm uma grande influência nas medidas de desempenho. Podemos portanto, a partir de uma análise de desempenho, encontrar valores para os intervalos de *switch-over* e as quotas de serviço (T_1, T_2) , que atendam os requisitos de retardo e probabilidade de bloqueio exigidos pelo sistema sendo projetado. Neste trabalho, nós mostramos que existem pontos ótimos de desempenho que podem ser identificados com a ferramenta aqui desenvolvida.

Quando comparado ao TDM, o sistema (T_1, T_2) mostrou um melhor desempenho quanto ao retardo, pois não desperdiça capacidade de canal, que poderia estar sendo utilizada. A comparação do sistema (T_1, T_2) com uma fila M/M/1/K nos mostrou que, em termos de retardo, a fila M/M/1/K é melhor que o esquema (T_1, T_2) , pois na fila não há períodos de *switch-over*, e os pacotes são servidos imediatamente um após o outro, na ordem de chegada. A desvantagem da fila M/M/1/K surge, quando a carga de um dos tráfegos aumenta. Nessas situações, o outro tráfego é bastante penalizado, em termos de retardo, pois não lhe é garantida uma capacidade mínima para transmissão, como no esquema (T_1, T_2) .

Além dos sistemas de *polling* com duas filas, também alguns sistemas (T_1, T_2, T_3) , com três filas, foram analisados e comparados com o esquema de prioridades do sistema HOL. O sistema (T_1, T_2, T_3) se mostrou mais flexível que o HOL, pois pode atender diversos requisitos de retardo exigidos conjuntamente pelos três tipos de tráfego, a partir da escolha apropriada das quotas T_1 , T_2 e T_3 .

Foi observado nos experimentos, que o método de modelagem e análise estudado aqui tem uma convergência rápida, e o custo de memória com as estruturas de dados usadas na sua implementação é relativamente pequeno.

Achamos interessante aprofundar, em trabalhos futuros, a pesquisa na modelagem e análise dos *overruns*, em sistemas com *time-out* não preemptivo. Também é importante estudar a extensão do modelo para outras distribuições dos intervalos de *switch-over*, bem como implementar as equações para as outras disciplinas de serviço modeladas em [1]. Além disso, é interessante aprofundar também a pesquisa na modelagem e análise do esquema de serviço, dependente do ciclo de *token*, para tráfegos assíncronos no protocolo FDDI e nas subestações virtuais menos prioritárias (classes 4, 2 e 0) do padrão IEEE 802.4 *Token Bus*. Este estudo irá nos permitir compreender melhor a dinâmica destes sistemas, ao mesmo tempo que desenvolverá a área de modelagem e análise de sistemas de *polling*.

O estudo e aplicação deste método, na modelagem de sistemas de *polling* com tempos de serviço limitados, dá importantes subsídios para o projeto de um sistema real, de forma que atenda aos requisitos de vazão, retardo e probabilidade de bloqueio, exigidos conjuntamente por diferentes tipos de tráfego.

Apêndice A

A seguir apresentamos o algoritmo usado para se determinar o valor de N_{min} e de $\mathcal{N}(N_{min})$ que aparecem em diversas equações e expressões deste trabalho. Este algoritmo, desenvolvido em [11], recebe como entrada o produto Rt , onde R é uma taxa e t é um intervalo de tempo, e como saída calcula o par $(N_{min}, \mathcal{N}(N_{min}))$.

```

$$N_{min} \leftarrow \max(0, Rt - 10\sqrt{Rt})$$

Se  $(N_{min} = 0)$  então
{

$$\mathcal{N}(N_{min}) = e^{-Rt}$$

retorne  $(0, \mathcal{N}(N_{min}))$ 
}
senão
{

$$E \leftarrow Rte^{-Rt/(N_{min}+1)}$$


$$I \leftarrow \lceil E \rceil$$


$$esq \leftarrow dir \leftarrow I$$


$$prod \leftarrow E/I$$

enquanto  $(esq > 1 \text{ e } dir < N_{min})$ 
{
enquanto  $(prod \leq 1 \text{ e } esq > 1)$ 
{

$$esq \leftarrow esq - 1$$


$$prod \leftarrow prod \times (E/esq)$$

}
enquanto  $(prod > 1 \text{ e } dir < N_{min})$ 
{

$$dir \leftarrow dir + 1$$


$$prod \leftarrow prod \times (E/dir)$$

}
}
enquanto  $(esq > 1)$ 
{

$$esq \leftarrow esq - 1$$


$$prod \leftarrow prod \times (E/esq)$$

}
```

```
    }  
enquanto (prod > 1 e dir <  $N_{min}$ )  
    {  
        dir ← dir + 1  
        prod ← prod × ( $E / dir$ )  
    }  
 $\mathcal{N}(N_{min}) = e^{-Rt/(N_{min}+1)} \times prod$   
retorne ( $N_{min}, \mathcal{N}(N_{min})$ )  
}
```

Referências Bibliográficas

- [1] DE SOUZA E SILVA, E., GAIL, H. R. e MUNTZ, R. R., *Polling Systems with Server Timeouts*, A ser publicado, 1992.
- [2] SRIRAM, K., *Dynamic Bandwidth Allocation and Congestion Control Schemes for Voice and Data Multiplexing in Wideband Packet Technology*, ICC-90, pp. 1003-1009, Abril, 1990.
- [3] SEVCIK, K. C. e JOHNSON, M. J., *Cycles Time Properties of the FDDI Token Ring Protocol*, IEEE Transactions on Software Engineering, v. SE-13, n. 3, pp. 376-385, Março, 1987.
- [4] PANG, J. W. M. e TOBAGI, F. A., *Throughput Analysis of a Timer Controlled Token Passing Protocol Under Heavy Load*, IEEE Transactions on Communications, v. 37, n. 7, pp. 694-702, Julho, 1989.
- [5] LEUNG, K. K. e EISENBERG, M., *A Single-Server Queue with Vacations and Gated Time-Limited Service*, IEEE Transactions on Communications, v. 38, n. 9, pp. 1454-1462, Setembro, 1990.
- [6] TAKAGI, H., *Effects of the Target Token Rotation Time on the Performance of a Timed Token Protocol*, IBM Research Report RT 0022, Novembro, 1989.
- [7] TSAI, Z. e RUBIN, I., *Performance of Token Schemes Supporting Delay-Constrained Priority Traffic Streams*, IEEE Transactions on Communications, v. 38, n. 11, pp. 1994-2003, Novembro, 1990.
- [8] YUE, O. e BROOKS, C. A., *Performance of the Timed Token Scheme in MAP*, IEEE Transactions on Communications, v. 38, n. 7, pp. 1006-1012, Julho, 1990.
- [9] GROSS, D. e MILLER, D. R., *The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes*, Operations Research, v. 32, n. 2, pp. 343-361, Março-Abril, 1984.
- [10] HEYMAN, D. P. e SOBEL, M. J., *Stochastic Models in Operations Research*, v. I, New York, McGraw-Hill, 1982.
- [11] DE SOUZA E SILVA, E. e GAIL H. R., *Performability Analysis of Computer Systems: from Model Specification to Solution*, Performance Evaluation, v. 14, pp. 157-196, 1992.

- [12] DE SOUZA E SILVA, E., Notas pessoais não publicadas, 1985.
- [13] BERTSEKAS, D. e GALLAGER, R., *Data Networks*, New Jersey, Prentice-Hall, 1987.
- [14] HAYES, J. F., *Modeling and Analysis of Computer Communications Networks*, New York, Plenum Press, 1984.
- [15] KLEINROCK, L., *Queueing Systems V. II : Theory*, New York, John Wiley & Sons, 1975.
- [16] ROSS, F. E., *An overview of FDDI: The Fiber Distributed Data Interface*, IEEE Journal on Selected Areas in Communications, v. 7, n. 7, pp. 1043-1051, Setembro, 1989.
- [17] SCHWARTZ, M., *Telecommunication Networks. Protocols, Modeling and Analysis*, Reading, Addison-Wesley, 1987.
- [18] DE SOUZA E SILVA, E. e GAIL, H. R., *Analyzing Scheduled Maintenance Policies for Repairable Computer Systems*, IEEE Transactions on Computers, v. 39, n. 11, pp. 1309-1324, Novembro, 1990.
- [19] SOARES, L. F. G., *Redes Locais*, Rio de Janeiro, Editora Campus, 1986.
- [20] TAKAGI, H., *Queueing Analysis of Polling Models: an Update*, ACM Computing Surveys, v. 20, n. 1, pp. 5-28, Março, 1988.
- [21] TAKAGI, H., *Analysis of Polling Systems*, The MIT Press, 1986.