Política de escalonamento de tempo real baseada em exigência para provisão de QoS absoluto em serviços Web

Lucas dos Santos Casagrande

Política de escalonamento de tempo real baseada em exigência para provisão de QoS absoluto em serviços Web

Lucas dos Santos Casagrande

Orientador: ***Prof. Dr. Francisco José Monaco***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC, USP, como parte dos requisitos para a obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	14/06/2007
Visto do Orientador :	

USP - São Carlos
Julho de 2007

A minha família, em especial aos meus pais, Antonio Pedro e Maria Helena.

Agradecimentos

Agradeço, primeiramente, a Deus, pelas oportunidades na minha vida.

A Universidade de São Paulo, USP - Campus São Carlos, pela oportunidade de realizar o curso de Pós-Graduação.

Aos meus pais, Antônio Pedro e Maria Helena, pela educação, por estarem presentes na maioria dos momentos difíceis e felizes da minha vida, pelo apoio incondicional e pelo carinho.

A meu orientador, Francisco José Monaco, pela orientação e confiança durante o desenvolvimento do trabalho, além da amizade e dos ensinamentos transmitidos.

Ao professor Rodrigo Fernandes de Mello, pelas conversas e contribuições sugeridas para o trabalho, além da amizade.

Aos meus colegas e amigos da USP e do LaSDPC, em especial ao Bert, pela ajuda e atenção demandada com intuito de melhorar a qualidade do trabalho realizado, bem como pela amizade. Ao Augusto pelas discussões e sugestões acerca do trabalho, mostrando-se sempre pronto a ajudar. Ao Tott, Maycon, Michelle, Sarita e Júlio pelas correções e sugestões feitas para melhor apresentação do trabalho. Também ao Geraldo, Fábio, Marcelo, Luis, Alessandra, Toni, Matheus, Hima Carla, Caio, Gustavo, Bruno, Henderson, Alexandre, Valter, Juliano, Japa, Lourenço, Thiago, Bruno, Felipe pela ajuda e por proporcionarem um ambiente de trabalho alegre.

Aos funcionários do ICMC-USP, Arli, Marcos, Roberto, Jô, pelo convívio amigo e descontraído, e pela disposição em sempre bem atender.

A CAPES, pelo apoio financeiro dado a este trabalho.

"*Do what you can, with what you have, where you are.*"

Theodore Roosevelt

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

ASPOL	A Simulation Process - Oriented Language
ATC	Apparent Tardiness Cost
CPU	Central Processing Unit
CSIM	C-based process-oriented simulation language
DiffServ	Differentiated Services
DM	Deadline-Monotonic
DS field	Differentiated Service field
EBS	Exigency Based Scheduling
EDD	Earliest Due Date
EDF	Earliest Deadline First
FCFS	First-Come First-Served
FIFO	First In First Out
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IntServ	Integrated Services
IP	Internet Protocol
ISP	Internet Service Provider
NS-2	Network Simulator - 2
PDF	Probability Density Function
PRIAdap	Prioridades Adaptativo
PS	Polling Server
QoS	Quality of Service
RM	Rate Monotonic
Rspec	Reservation Specification
RSVAdap	Reserva Adaptativa de Recursos

RSVP	Resource Reservation Protocol
RT	Real-Time
SBSA	Session Based Scheduling Algorithm
SFD	Short Flow Differentiating
SJF	Shortest Job First
SLA	Service Level Agreement
SMPL	SiMulation Program Language
SRPT	Shortest Remaining Processing Time
TOS	Type Of Service
Tspec	Traffic Specification
VoIP	Voice Over Internet Protocol
Web	World Wide Web
WFQ	Weighted Fair Queueing
WSPT	Weighted Shortest Processing Time

Resumo

Este trabalho apresenta um estudo, implementação e validação em ambiente simulado de uma política de escalonamento de tempo real para provisão de *QoS* absoluto em serviço *Web*. Sintetizando características de escalonamento de tempo real, com baixa latência e de modelo re-alimentado, a política proposta permite um ajuste ponderado pela quantificação da exigência à qual o sistema está submetido por meio de suas classes. A meta é oferecer ações imediatas às requisições mais urgentes, sem, entretanto, degradar a qualidade do sistema como um todo. Verificou-se que a estratégia de escalonamento baseada em exigência (*EBS - Exigency-Based Scheduling*) é benéfica para o controle da qualidade de serviço oferecida. Escalonar de forma a evitar demasiado peso imposto ao sistema permite que o servidor tenha mais condições de cumprir os requisitos contratuais. Também foi alvo do estudo a criação de uma métrica de avaliação da satisfação de atendimento por parte dos usuários dos serviços. Os resultados alcançados com o emprego da política *EBS* sinalizam uma melhoria em termos de qualidade de serviço e melhor satisfação dos clientes de forma balanceada.

Abstract

The present work presents a study, implementation and validation in a simulated environment of a real time scheduling policy to provide absolute QoS for web services. Synthesizing characteristic from real time scheduling, low latency and feedback scheduling, the proposed policy allows an adjustment weighed by the quantification of the exigency which the system is exposed through its classes. The goal is to offer immediate actions to most urgent requests, without decreasing the system quality as a whole. It was verified that the scheduling strategy based on exigency (EBS - Exigency-Based Scheduling) helps to control the quality of service offered. Scheduling in order to avoid imposing a heavy load to the system gives more condition to the server to fulfill the requirements agreed. Another goal of this work is the creation of a metrics to evaluate the client satisfaction. The results achieved with the EBS policy indicate a higher quality of service and better client satisfaction.

Capítulo

1

Introdução

1.1 Contextualização

A *Web* tem desempenhado um papel de destaque na evolução e disseminação da *Internet*, integrando-a cada vez mais ao cotidiano das pessoas, seja para fins educativos, de entretenimento ou comerciais. Esse crescimento se traduz não apenas na multiplicação do número de usuários e aplicativos, mas também no surgimento, cada vez maior, de novos tipos de aplicações. No entanto, o modelo atual de serviços oferecido pela *Internet*, baseado no paradigma de "melhor esforço" (*best-effort*)[1], embora ainda funcione bem para vários tipos de aplicações, tem mostrado sinais de que está atingindo seu limite no sentido de ser capaz de atender os variados requisitos operacionais das novas aplicações.

A abordagem tradicionalmente utilizada nos sistemas computacionais que provêm tais serviços, trata todos os clientes sem distinção, atendendo suas requisições de acordo com a política *FCFS*[2], dessa forma, um servidor saturado deixará de responder adequadamente. As requisições dos clientes passam a acumular na fila de entrada do servidor *Web* e, em caso de sobrecarga da fila, elas são descartadas de forma indiscriminada, sem considerar sua importância. Esse tratamento imparcial pode não ser adequado para certas aplicações, como as de comércio eletrônico, nas quais descartes e atrasos nos tempos de resposta podem causar perdas significativas de rendimentos devido a insatisfação dos usuários com o serviço recebido. Para diversas outras aplicações, como *e-banking*, sistemas de reserva, sistemas multimídia, tele-medicina, ensino a distância, a ausência de garantias quanto ao

[1] A rede procura transportar os dados que lhe são confiados no menor tempo possível, de preferência sem erros ou inconsistências, mas sem oferecer garantias de que isso realmente ocorrerá.

[2] *Fist-Come First-Served* (Primeiro a chegar, primeiro a ser servido).

cumprimento dos requisitos operacionais também é um fator relevante, tornando assim Qualidade de Serviço (*QoS*[3]) na *Web* uma questão importante.

QoS se refere à capacidade dos elementos de uma rede em prover garantia(s) acerca de determinados parâmetros associados à percepeção da qualidade de um dado serviço oferecido. No caso de tráfego de rede, por exemplo, a *QoS* pode ser implementada, seja oferecendo recursos dedicados (como capacidade de banda), controlando latências e *jitter*[4] (requisitos fundamentais de aplicações de tempo real e tráfego interativo), seja diminuindo situações de perdas de pacotes (ou requisições). A preocupação se dá não apenas na priorização, mas também em garantir que a qualidade oferecida a um ou mais serviços não prejudique os demais. Isso é o que protocolos com suporte a qualidade de serviço são projetados para fazer. *QoS* não cria mais recursos, mas os gerenciam de modo mais eficiente, com o objetivo de atender os diversos requisitos das aplicações. A meta é prover certo grau de previsibilidade e controle que o serviço de melhor esforço não está preparado para oferecer.

Devido à complexidade da infra-estrutura da *Web*, para que se possa garantir a qualidade de seus serviços é preciso que ações sejam aplicadas em todos os seus componentes, desde a tecnologia de rede e protocolos, até arquiteturas de *hardware* e *software* dos servidores *Web* e *proxies*. A maioria dos esforços têm se concentrado em oferecer *QoS* no nível de rede, como os projetos para arquiteturas de Serviços Integrados (*IntServ*) (Braden *et al.*, 1994) e de Serviços Diferenciados (*DiffServ*) (Blake *et al.*, 1998). No entanto, para que, a qualidade de serviço seja realmente fim-a-fim é importante que os servidores *Web* também ofereçam suporte a *QoS* no nível de aplicação, para que em situações de sobrecarga, as requisições não sejam tratadas indiscriminadamente, o que anularia os esforços empreendidos ao longo da rede.

Um meio de assegurar níveis de qualidade de serviço nos servidores *Web* é com a predição de demanda de seus serviços e com o aumento de seus recursos. Entretanto, além de ser difícil prever com precisão a demanda, superestimar recursos para tratar curtos períodos de sobrecarga pode ser financeiramente inviável. A existência de tráfego em rajadas, inclusive, pode levar a demanda por recursos a níveis cada vez maiores. Uma medida mais viável e de melhor custo benefício é a inclusão de mecanismos e algoritmos capazes de gerenciar os recursos do servidor de acordo com as prioridades e necessidades estabelecidas para cada serviço oferecido.

A especificação da qualidade de serviço pode se dar de maneira relativa ou absoluta. Quando feita em termos relativos, a preocupação com a qualidade de serviço oferecida se dá de maneira qualitativa, comparando-se o tratamento oferecidos para as diversas classes de serviço. O que se pretende garantir nesses casos é que uma classe de maior prioridade tenha um tratamento melhor que o de qualquer classe inferior. Uma *QoS* absoluta, por sua

[3] *Quality of Service.*
[4] Variação dos atrasos.

vez, estabelece métricas a serem cumpridas, como garantir uma taxa mínima de serviço ou um atraso máximo de atendimento para as requisições, trabalhando com qualidade em termos quantitativos.

1.2 Motivação e Objetivo

Em vista à grande relevância atual da *Web*, como influente meio de comunicação e tecnologia fundamental para os sistemas de informação das mais avançadas empresas e organizações, o que motiva este trabalho é a importância de aprimoramento na infra-estrutura da *Web* para oferecer garantias de qualidade de serviço e atender assim as exigências das novas aplicações. Assim, em vista dessa motivação e do predomínio de trabalhos que abordam o tema no nível de rede, o objetivo deste trabalho é formular e avaliar uma abordagem para oferecimento de suporte a *QoS* também no nível de aplicação (i.e. servidor *Web*), contribuindo, assim, para a implementação da qualidade fim-a-fim.

Diferentemente da maioria dos trabalhos atuais sobre o tema, a especificação se dará em termos absolutos, com a qual se pode oferecer garantias mais estritas de qualidade, atendendo assim às necessidades do número crescente de aplicações com requisitos responsivos tais como: comércio eletrônico, tele-medicina, sistemas multimídia, telefonia, ensino a distância e demais aplicações *online*. O foco da abordagem está em políticas de escalonamento que ofereçam garantias quantitativas de tempos de resposta especificadas em termos estocásticos. Tal garantia objetiva a provisão de valores médios de latência de sistema[5] dentro de uma faixa de tolerância especificados por contratos de serviço firmados entre o usuário e o provedor *Web*.

1.3 Estrutura

O restante deste documento está organizado como segue:

- No capítulo **capítulo 2** apresenta-se a fundamentação teórica que embasa o trabalho, reunindo de forma sucinta os principais conceitos utilizados no seu desenvolvimento e que são importantes para sua análise, como Qualidade de Serviço (seção 2.2), Tempo Real (seção 2.3) e Modelagem e Simulação de Sistemas (seção 2.4).

- No **capítulo 3** relacionam-se alguns trabalhos mais relevantes da área estudada, mostrando o que tem sido feito, quais linhas de pesquisa têm recebido mais esforços, as que precisam ser mais desenvolvidas, e os fundamentos e contribuições que podem ser aproveitados na busca por melhorias para projetos na área.

[5]Tempo total de residência no sistema

- No **capítulo 4** descreve-se o projeto propriamente dito, apresentando a metodologia empregada (seção 4.2), algumas formulações e métricas criadas neste trabalho (seção 4.3) e as fases de desenvolvimento do algoritmo proposto (o *EBS*) (seção 4.4).
- No **capítulo 5** apresentam-se os resultados obtidos com a avaliação da política *EBS*, fazendo-se uma análise baseada nas métricas propostas pela modelagem analítica.
- No **capítulo 6** resumem-se os principais resultados obtidos, as contribuições do projeto e sugere-se melhorias como trabalhos futuros.
- Finalmente são apresentadas as **Referências** e o **Apêndice** com demais gráficos por clareza não incluídos na seção de análise dos resultados.

Capítulo

2

Fundamentação Teórica

2.1 Considerações Iniciais

Neste capítulo faz-se uma discussão sobre os novos desafios e demandas impostas à *Internet*, apresentando um embasamento teórico do modelo de serviços que tem sido proposto para lidar com eles: o de Qualidade de Serviço. Esforços em outras áreas, como Tempo Real, também têm sido feitos com o intuito de propor novas abordagens para o problema, sendo importante uma breve revisão de seus principais conceitos.

O capítulo começa com uma discussão de conceitos básicos relacionados à Qualidade de Serviço (seção 2.2), apresentando seus principais componentes, as formas como ela pode ser especificada (seção 2.2.1), as principais arquiteturas que oferecem suporte a esse modelo de serviços (seção 2.2.2), e destaca a importância de seu uso em todos os elementos da rede, inclusive no nível de aplicação (seção 2.2.3).

Em seguida, aborda-se de forma sucinta alguns fundamentos da área de Tempo Real (seção 2.3), pois ao relacionar parâmetros de serviço em termos temporais utiliza-se conceitos desse campo. Conceitos básicos da área são discutidos (seção 2.3.1), apresentando suas duas principais vertentes (seção 2.3.2), e as políticas de escalonamento mais empregadas (seção 2.3.3).

Por fim, apresenta-se um pouco da teoria de modelagem e simulação de sistemas (seção 2.4), ferramentas úteis para desenvolvimento e análise de novas políticas de escalonamento. Nessa seção, além de definições básicas de simulação e sistemas, apresenta-se também uma técnica (seção 2.4), e a teoria a partir da qual ela é fundamentada (seção 2.4.1), para modelagem de sistemas com ocorrência de filas, foco deste trabalho. Um pouco da teoria de probabilidade e estatística (seção 2.4.2) também é discutido devido a importância de suas ferramentas para a análise de modelos de sistemas.

2.2 Qualidade de Serviço

O modelo atual de serviços da *Internet*, de forma geral, trata todas as requisições de modo equivalente, seja quando processadas pelos servidores, ou quando transmitidas ao longo da rede. O grande crescimento e diversificação de suas aplicações têm exigido mudanças sobre esse paradigma de serviço, tornando a questão qualidade de serviço cada vez mais crítica e importante, uma necessidade para a evolução da *Internet*, principalmente no segmento da utilização comercial (Bhatti *et al.*, 2000; Strnadl, 2002).

Toda rede baseada no protocolo *IP* (*Internet Protocol*), como a *Internet*, essencialmente provê apenas um serviço de "melhor esforço" na entrega de dados. Isto possibilita que a complexidade do modelo resida apenas nos sistemas finais, o que além de tornar os elementos intermediários mais simples, garante à rede alta escalabilidade. Com esse tipo de serviço, quando um usuário envia seus dados, ele o faz compartilhando a capacidade de transmissão com todos os fluxos de dados dos demais usuários. Os fluxos realizam a melhor rota possível para chegar ao seu destino, conforme as rotas definidas e a capacidade de banda que estiverem disponíveis no momento. Quando há congestionamento, pacotes de dados[1] podem sofrer atrasos ou até serem descartados pelos elementos comutadores[2] sem distinção. Não há garantia de que o serviço será realizado com sucesso, nem mesmo de desempenho, justificando assim a denominação de serviço "melhor esforço" (Vasiliou, 2000).

Esse modelo de serviço, no entanto, tem contrastado com o atual estágio de expansão da *Internet* e com os novos requisitos de suas aplicações. A convergência de outras redes, como rádio, telefonia e TV, para *Internet* tem contribuído para torná-la como o principal meio de informações, condução de negócios e entretenimento do futuro. Isso tem diversificado cada vez mais o tipo de aplicações e serviços que surgem no ambiente *Web*, os quais para um correto funcionamento impõem à rede certos requisitos de tempo e tráfego que não estavam previsto no projeto original da *Internet* (Stardust, 1999).

À primeira vista, o aumento da capacidade de banda parece ser a forma mais direta de tratar o problema, já que em quantidade suficiente teoricamente não haveria congestionamento, e assim situações de atrasos e descartes não ocorreriam. Entretanto, a experiência mostra que, não importa a capacidade de banda seja disponibilizada, a tendência é, em pouco tempo, tal quantidade ser totalmente exaurida. A existência de tráfego em rajadas, inclusive, pode levar a demanda por capacidade de banda a níveis cada vez maiores. Além disso, dentre essas novas aplicações existem aquelas em que o ponto crítico não está na quantidade de banda disponível, mas sim na latência de transmissão, como ocorre com aplicações de telefonia sobre *IP* (Teixeira, 2004).

[1] Datagramas.
[2] Roteadores.

Como primeiro passo o acréscimo de capacidade de banda é uma medida necessária, mas não suficiente para resolver o problema. A solução é que certo nível de inteligência (i.e. complexidade) deve ser introduzido na *Internet* de tal forma que haja um gerenciamento ativo da capacidade de banda, buscando-se garantir os requisitos operacionais das aplicações. Dessa forma, tráfegos com requisitos estritos de tempo podem ser diferenciados daqueles que podem tolerar certos níveis de atraso ou perdas. Esse gerenciamento é feito por meio de mecanismos de Qualidade de Serviço (*QoS*), que buscam oferecer aos usuários da rede níveis de confiabilidade na satisfação de seus requisitos.

Nesta seção são apresentados os principais conceitos de Qualidade de Serviço (seção 2.2.1) em redes *IP*, descrevendo algumas de suas arquiteturas (seção 2.2.2), como a de serviços integrados e a de serviços diferenciados, discutindo também a necessidade de *QoS* em nível de aplicações (seção 2.2.3).

2.2.1 Conceitos Básicos de *QoS*

Qualidade de serviço pode ser definida como a capacidade de fornecer a um elemento de rede (como aplicação, *host*[3] ou roteador) algum nível de segurança de que seus requisitos de serviço serão satisfeitos. Está relacionada com a garantia de um atraso de entrega (*delay*) e uma perda de pacotes suficientemente baixos para certos tipos de aplicações ou tráfego.(Zhao *et al.*, 2000).

O emprego de *QoS* não é capaz de criar capacidade de banda, ou seja, a rede nunca poderá fornecer aquilo que ela não tem. O que ela faz é administrar a capacidade de banda existente segundo a demanda das aplicações e dentro de certos parâmetros de gerenciamento e desempenho da rede. Uma rede habilitada para fornecer *QoS* continuará dando suporte ao tráfego de melhor esforço (como transferência de hipertexto, *FTP*), contudo parte da capacidade de banda será otimizada para as aplicações com necessidades especiais (como voz sobre *IP*, vídeo-conferência). Também faz parte das atividades de gerenciamento de *QoS* garantir que essas aplicações menos exigentes não serão anuladas pelas aplicações mais nobres (Teixeira, 2004).

Dentre suas principais formulações estão: **garantia de desempenho** e **diferenciação de serviço**. No nível de rede, o primeiro está diretamente relacionado com capacidade de banda, atraso, *jitter*[4] e perda de pacotes; a diferenciação de serviço, por sua vez, refere-se a oferecer diferentes níveis de prioridade a diferentes aplicações, cada qual com requisitos distintos.

Existem aplicações aquelas mais sensíveis a perda de dados, como as de transferência de arquivos por *FTP*, em que cada *bit* de dado é importante. Aplicações de telefonia, por outro lado, toleram certas perdas ocasionais de pacotes, mas são sensíveis a atrasos (*delay*), exigindo que seus pacotes cheguem em tempos pré-determinados. Aplicações

[3]Máquina hospedeira de uma rede.
[4]Variação do atraso.

multimídia, além de baixos atrasos, têm sua utilidade relacionada a previsibilidade de *jitter*. O fator humano também influência nos requisitos de *QoS* exigidos. Apesar da transferência de arquivos não ser sensível a atrasos, a paciência humana demanda baixos tempos de transferência.

QoS pode ser especificada em termos **absolutos** ou **relativos**. Garantia de desempenho ou de confiabilidade tipicamente são descritas de forma absoluta (quantitativa), estabelecendo para isso limites de atrasos e perdas, como por exemplo: "nenhum pacote atrasará mais que 200 *ms*" ou "perda de pacotes não excederá 1%". Nos modelos absolutos o usuário só terá acesso à rede se seus níveis de desempenho e confiabilidade puderem ser garantidos, seguindo assim a lei do "tudo ou nada".

QoS relativa está estritamente relacionada com o conceito de diferenciação de serviço, garantindo apenas que uma classe mais alta receberá um serviço melhor (ou no mínimo não pior) que qualquer classe mais baixa. A diferenciação entre classes é realizada perante alguma política de escalonamento, que se baseia em alguns critérios para distinção dos serviços oferecidos, como por exemplo melhor serviço demanda maiores custos por parte do usuário. (Vasiliou, 2000)

Em termos de tráfego as especificações de *QoS* podem ser feitas para um grupo individual de pacotes (fluxo[5]) ou para um agregado de grupo de fluxos associados. A especificação feita para um fluxo permite uma granulosidade menor, oferecendo com isso maiores garantias de desempenho, a medida que leva em conta os requisitos dos fluxos individualmente. No entanto se mostra pouco escalável, pois com o crescimento do volume de dados a complexidade de roteamento torna-se bem maior, devido ao grande número de fluxos individuais. A agregação de grupos de fluxos associados soluciona o problema de escalabilidade e reduz a complexidade de roteamento, no entanto o aumento da granulosidade reduz as garantias de desempenho.

2.2.2 Arquiteturas para *QoS*

Dentre os vários modelos de serviço e mecanismos para suporte à *QoS* em nível de rede, atualmente em uso na *Internet*, destacam-se dois modelos propostos pela *IETF* (*Internet Engineering Task Force*)[6]: o de **Serviços Integrados** (*IntServ*) (Braden *et al.*, 1994) e o de **Serviços Diferenciados** (*DiffServ*) (Blake *et al.*, 1998). O modelo *IntServ* (*Integrated Services*) é caracterizado pela reserva de recurso. Ele utiliza um protocolo de sinalização que estabelece um caminho entre dois sistemas finais e realiza a reserva de recursos ao longo dele. Já o *DiffServ* (*Differentiated Services*) é uma proposta em que os pacotes de dados são marcados de acordo com classes de serviços pré-determinadas e comutados entre domínios de rede, segundo contratos regulamentadores de serviço estabelecidos entre o cliente e o provedor (Zhao *et al.*, 2000).

[5]Fluxo de dados individual e unidirecional estabelecido entre duas aplicações.

[6]*www.ietf.org*

Serviços Integrados

O modelo de serviços integrados é caracterizado pelo estabelecimento de chamada e pela reserva de recursos. Antes de iniciar uma comunicação, cada roteador da rede que compõe o trajeto entre a fonte e o destino deve estar habilitado a reservar recursos suficientes para garantir uma boa qualidade na transmissão dos dados. A reserva de recursos diz respeito à disponibilidade de *buffers*[7], capacidade da banda de enlace e ao tempo em que a conexão será mantida. Neste período, o emissor daquele serviço tem uma faixa da largura de banda disponível para transmitir seus dados. Este processo de estabelecimento de chamada está ilustrado pela Figura 2.1 (Kurose & Ross, 2006).

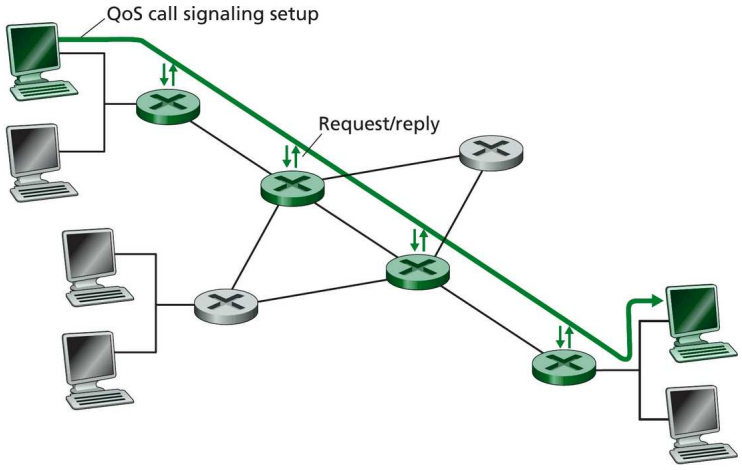


Figura 2.1: ***Processo de Estabelecimento de Chamada.*** ***(Kurose & Ross, 2006)***

As etapas desse processo estão descritas a seguir:

1. **Caracterização do tráfego e especificação da *QoS* desejada**: para que um roteador seja capaz de determinar se possui ou não recursos suficientes para garantir os requisitos de *QoS* de uma dada sessão, a mesma deve primeiramente informar suas exigências e caracterizar o tráfego que estará enviando à rede. Na arquitetura *IntServ*, a requisição que define a *QoS* exigida é denominada *Rspec* (*Reservation Specification*), enquanto que a requisição que caracteriza o tráfego a ser enviado pelo remetente é denominada *Tspec* (*Traffic Specification*).

2. **Sinalização para o estabelecimento da chamada**: as requisições *Rspec* e *Tspec* são transportadas aos roteadores por meio de mensagens de alocação de recursos. Um destes protocolos amplamente utilizados é o *RSVP* (*Resource Reservation Protocol*) (Braden *et al.*, 1997).

3. **Aceitação de chamada por elemento**: ao receber a especificação de tráfego e o tipo de serviço exigido, o roteador poderá analisar o quanto o seus recursos

[7] Área usada para armazenamento temporário de dados na memória do computador durante operações de entrada/saída.

existentes ficarão comprometidos, e assim determinar se é possível ou não aceitar a chamada.

A arquitetura de serviços integrados propõe mais duas classes de serviço, além da de melhor esforço:

- **Serviço garantido** (*guaranteed service*) para aplicações que exigem limite fixo para os atrasos de fila que um pacote sofrerá em um roteador;
- **Serviço de carga controlada** (*controlled-load service*) para aplicações que exigem serviço de melhor esforço mais confiável e aprimorado. Este tipo de serviço pode admitir que uma "porcentagem muito alta" de seus pacotes passará com sucesso pelo roteador sem ser descartada e sofrerá atraso próximo de zero na fila do roteador. Com isso se oferece garantias quantitativas de desempenho (não especifica o que constitui uma porcentagem muito alta de pacotes nem qual é a qualidade de serviço que fica muito próxima à de um elemento de rede que não tem carga).

Para oferecer tais serviços todos os roteadores ao longo do caminho devem ser capazes de realizar reserva de recursos e manter estado dos fluxos de dados transmitidos. Um dos protocolos amplamente utilizados para isso é o protocolo de sinalização *RSVP*, que atua sobre o *IP*, ocupando o lugar do protocolo de transporte, da mesma forma que o *ICMP* (*Internet Control Message Protocol*), *IGMP* (*Internet Group Management Protocol*) ou outros protocolos de roteamento.

Para oferecer qualidade de serviço a uma conexão, o *RSVP* estabelece um caminho entre os sistemas origem e destino, perguntando a todos os nós intermediários se eles suportam a *QoS* desejada, e reservando as necessidades daquela aplicação. Para tanto, todos os nós no meio do caminho devem suportar esse protocolo. O protocolo *RSVP* utiliza outros protocolos para efetuar roteamento e transmissão. Seu único objetivo é a reserva, manutenção e liberação de recursos quando solicitado. Assim, ele pode operar em *unicast*, *multicast*, *IPv4*, *IPv6* e outros (Estrella, 2005).

Um exemplo de funcionamento básico do *RSVP* está ilustrado pela Figura 2.2. Nele, um sistema emissor (*sender*) manda uma mensagem *PATH* ao receptor especificando as características do tráfego (1). Cada roteador intermediário (*router*) repassa esta mensagem para o próximo roteador (2), até a mensagem chegar ao receptor (*receiver*) final (3). Quando isto ocorre, o receptor responde com uma mensagem *RESV* (4), que percorre o caminho contrário no sentido do emissor (5), tentando solicitar a reserva de recursos para o fluxo de dados. Caso algum roteador intermediário rejeite a solicitação, uma mensagem de erro é enviada de volta ao receptor (2), sinalizando o fim do processo. Se a mensagem for aceita, certa capacidade de banda e espaço em *buffer* são reservados, com o roteador passando a manter informações de estado do fluxo de dados (2) (Zhao *et al.*, 2000).

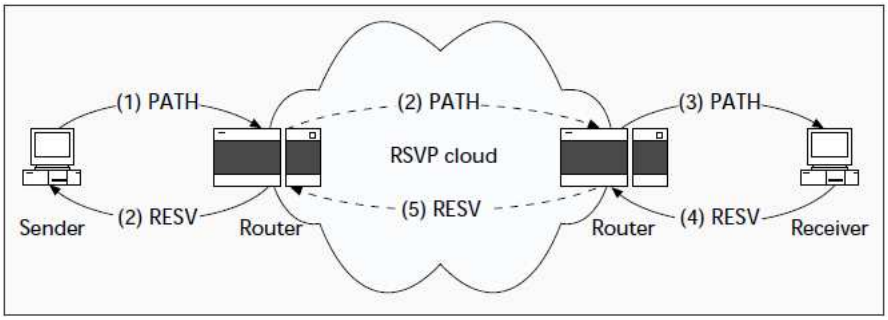


Figura 2.2: ***Sinalização do RSVP.*** *(Zhao et al., 2000)*

Alguns problemas dificultam a implementação e a implantação do modelo *IntServ*:

- Como a reserva de recursos é feita para cada fluxo individual, a quantidade de informações de estado aumenta consideravelmente com o aumento de fluxos, exigindo grande capacidade de armazenamento e elevado poder de processamento por parte dos roteadores, tornado o núcleo da rede mais complexo, e por conseqüência prejudicando a escalabilidade típica da Internet;

- Todos os roteadores ao longo da rota devem oferecer suporte à serviços integrados.

Serviços Diferenciados

Como alternativa à dificuldade de implementação e implantação de serviços integrados formulou-se um modelo que oferece *QoS* com base na definição de tipos de serviços, denominado Diferenciação de Serviços (*DiffServ*). No cabeçalho de um pacote *IP*, existe um campo chamado *TOS* (*Type of Service*) que pode representar o tipo de serviço. Basicamente, o que o modelo de serviços diferenciados faz é definir um *layout* para o campo *TOS*, que passa a ser chamado *DS field* (*Differentiated Service field*), e especificar um conjunto de procedimentos distintos de envio de pacotes baseado neste *layout*, oferecendo com isso diferentes classes de serviço.

Classes de serviço normalmente são especificadas de acordo com contratos de nível de serviço, denominados *SLA* (*Service Level Agreement*), firmados entre o usuário e o provedor de serviço de Internet (*ISP - Internet Service Provider*), ou entre domínios adjacentes. Um contrato *SLA* pode ser estáticos ou dinâmicos. Contratos estáticos são negociados de maneira regular (mensalmente ou anualmente, por exemplo). Já contratos dinâmicos possibilitam ao usuário solicitar serviços sob demanda. Este ultimo tipo de contrato se utiliza de algum protocolo de sinalização (como o *RSVP*) para indicar a necessidade do usuário em um dado momento (Zhao *et al.*, 2000).

As redes que implementam serviços diferenciados são chamados Domínios *DS*. Todos os pacotes que fluem de um domínio para outro são fiscalizados nos roteadores de borda para verificar sua conformidade com os contratos *SLA*. No centro da rede (*core*), os roteadores

simplesmente encaminham os pacotes para os seus destinos, oferecendo algumas garantias de *QoS* a determinados pacotes. Este tratamento distinto de encaminhamento é denominado comportamento de salto (*Per-Hop Behavior*).

A combinação do comportamento de salto no centro da rede com as regras de policiamento na borda, permitem a criação de vários serviços em uma rede de Serviços Diferenciados. Atualmente estão sendo padronizados dois comportamentos de salto pelo *IETF*: encaminhamento expresso (ou *premium*) e encaminhamento assegurado. O comportamento assegurado pode ser utilizado por serviços que necessitam de garantias não muito rígidas, para obter diferenciação (preferência) aos seus pacotes fluindo na rede. Eles são destinados aos clientes que exigem uma garantia de cumprimento de seus requisitos de *QoS* em momentos de congestionamento. Por outro lado, o comportamento de salto expresso define garantias mais rígidas de *QoS* para aplicações muito sensíveis a variações de características temporais da rede, que necessitam de baixo atraso e baixo *jitter* (Variação do atraso) (Santos, 1999).

A utilização do campo *DS*, apesar de possibilitar a representação de um número limitado de classes, torna o modelo de Serviços Diferenciados mais escalável, já que a manutenção de informações de estados é proporcional ao número de classes e não ao de fluxo. Além disso, ele exige menos dos roteadores, necessitando pouca atualização de *software* para prover bons métodos de classificação, policiamento, montagem e remarcação de pacotes.

2.2.3 *QoS* em Nível de Aplicação

Nos tópicos anteriores discutiu-se formas de garantir *QoS* no nível de rede, particularmente as abordagens de serviços diferenciados e integrados. Contudo, para que a provisão de *QoS* seja ampla é preciso a cooperação de todas as camadas da rede, de cima a baixo, assim como de todo e qualquer elemento da rede, de fim-a-fim. Um servidor *Web* não preparado para oferecer *QoS* anulará quaisquer esforços que tenham sido empreendidos pela rede nesse sentido, pois ela tratará todas as solicitações que receber por igual, ignorando a priorização relativa das mesmas. O aumento da utilização e diversificação de aplicações exigindo qualidade de serviço na *Internet* têm direcionado o foco de pesquisas em *QoS* para abranger também servidores *Web*.

A expansão da *Internet* de uma infra-estrutura de comunicação e navegação para também um centro de compras eletrônico tornou a provisão de qualidade de serviço adequada uma questão crítica, principalmente para o comércio. Com isso, as expectativas impostas sobre os servidores *Web* estão sendo cada vez exigentes, requisitando que eles ofereçam um serviço de qualidade, confiável e seguro, mesmo sob um ambiente tão dinâmico e imprevisível quanto a *Internet*. A não satisfação dessas expectativas tendem a implicar em impactos econômicos e sociais para usuários e provedores de serviço.

No entanto, a maior parte dos servidores atuais são desprovidos de tais capacidades e ainda tratam todas as solicitações uniformemente, sem nenhum tipo de diferenciação, segundo a disciplina *FCFS* (*First-Come First-Served*), o que muitas vezes os transformam em pontos de estrangulamento do sistema. Em situações de sobrecarga esse quadro se torna crítico, pois os servidores passam a oferecer tempos de resposta inaceitáveis ou então descartam requisições de forma indiscriminada. Isso é ruim por uma série de razões. Primeiro porque nem todas as transações são igualmente importantes, tanto para os clientes quanto para o provedor. O servidor *Web* pode, por exemplo, pode ter melhor política oferecer melhor serviço aos usuários assinantes, para os quais a percepção de *QoS* tem um impacto maior. Segundo porque o descarte indiscriminado pode significar desperdício de recurso, tendo em vista que operações de aceitação e fechamento de conexões são custosas, especialmente em situações de sobrecarga. Por fim, servidores sobrecarregados podem ter um impacto significativo sobre a percepção de tempo de resposta dos usuários, que exigem baixas latências e bom nível de desempenho do *site* visitado. Uma percepção negativa dessas expectativas repercutem negativamente na experiência do usuário quanto ao serviço prestado. Um gerenciamento ativo de recursos também no nível de aplicação é importante para evitar ou, pelo menos, lidar com situações como essas.

Assim, provisão de *QoS* apenas no nível de rede não é suficiente para garantir mecanismo completo de qualidade fim-a-fim, visto que somente o gerenciamento de banda e controle de congestionamento não podem resolver problemas de congestionamento e gargalos de atendimento no servidor. O escalonamento *FIFO* desempenhado por muitos servidores pode comprometer as garantias de *QoS* oferecidas ao longo da rede, uma vez que um servidor ocupado pode descartar indiscriminadamente pacotes de rede de alta prioridade. Assim, uma solução integrada para *QoS* fim-a-fim na *Web* deve ter como componente um servidor *Web* com mecanismos de *QoS* para proteção contra sobrecarga e capacidade de gerenciamento de recursos.

A leitura do capítulo 3 permite a análise de vários trabalhos que se preocupam com o fornecimento de *QoS* em nível de aplicação, particularmente no contexto de servidores *Web*.

2.3 Tempo Real

A garantia de requisitos de *QoS* dadas na forma de restrições temporárias insere o problema tratado no domínio das técnicas de análise e síntese de sistemas de tempo real.

Com o rápido crescimento das capacidades dos computadores modernos, sistemas de tempo-real (*RT*[8]) estão sendo empregados em diversas aplicações (controle de aviões e de tráfego aéreo, robótica, monitoramento de pacientes, realidade virtual) com crescente complexidade e importância econômica e social.

[8] *Real-Time*.

O tipo de computação exigido por tais sistemas pode variar desde complexos cálculos, como em sistemas de radares, até cálculos relativamente simples sobre extensas quantidades de dados, como processamento de imagens, apresentando assim diferentes escalas de tempo. Entretanto, o que mais caracteriza sistemas *RT* não é a escala de tempo operacional, mas a ênfase imposta na definição de seus comportamentos. Assim, tais sistemas não necessariamente implicam em desempenho computacional, mas sim em assegurar o tempo de resposta do sistema aos eventos externos (Nissanke, 1997).

Normalmente o termo *RT* é aplicado aos sistemas artificiais projetados de modo a atender determinados requisitos temporais. Entretanto, tal denominação não se refere diretamente a características inerentes aos mecanismos destes sistemas, mas sim à perspectiva empregada na concepção deles com respeito às suas características responsivas. Assim, rigorosamente, em vez de dizer que um dado sistema é ou não intrinsecamente *RT*, é mais próprio dizer que lhe cabe ou não um tratamento segundo a abordagem *RT*.

Esta seção apresenta os principais conceitos relacionados com a teoria que embasa sistemas *RT*, apresentando primeiramente alguns conceitos preliminares (seção 2.3.1), passando em seguida para a diferenciação que caracteriza duas principais vertentes destes sistemas (seção 2.3.2): *Hard* e *Soft-RT*. Por fim é apresentada uma breve descrição a respeito de escalonamento empregado para tais sistemas (seção 2.3.3).

2.3.1 Conceitos Básicos de Tempo-Real

Constitui uma definição útil aquela que caracteriza um sistema *RT* como aquele em que há requisitos de sincronismo de eventos internos ao sistema em relação a eventos externos ao sistema, isto é, as especificações temporais exigem que entradas e saídas devam ocorrer em determinados instantes, definidos com respeito ao tempo natural. Um sistema que não é *RT* pode ser designado de tempo-virtual.

Como ilustração da diferença entre as duas abordagens, caracterizam-se como de tempo-virtual sistemas de informações convencionais onde um banco de dados disponibiliza as entradas para a aplicação no momento que esta as requer, e está disponível para receber as saídas geradas pelo processamento no instante em que estas são geradas pela aplicação.

Como contra-exemplo, é *RT* um sistema computacional de monitoramento e controle industrial que deve obter a leitura de sensores mediante eventos que ocorrem assincronamente em relação aos estados internos do sistema (por exemplo um alarme), e deve gerar saídas em tempos referentes à dinâmica do processo, também independentes da temporização interna da aplicação. No primeiro caso, o projeto do sistema não leva em conta a preocupação com a sincronização entre as entradas e saídas com relação ao ambiente, as quais podem ocorrer segundo o tempo virtual interno ao sistema.

No segundo caso, as entradas não estão disponíveis quando a aplicação às requisitam, tampouco as saídas podem ser geradas a qualquer momento; o sistema deve sincronizar-se à eventos externos de modo a estar pronto para receber as entradas quando elas estiverem temporariamente disponíveis, e deve gerar as saídas em tempo hábil determinado pela dinâmica do ambiente (é o sistema que se sincroniza ao ambiente e não vice-versa).

A fim de descrever e caracterizar os diferentes tipos de sistemas *RT*, bem como os métodos para escalonamento e gerência de recursos, utilizam-se termos gerais para tratar da carga de trabalho de sistemas computacionais e de comunicação. Desta forma, a unidade de trabalho agendada e executada por um sistema será denominada por este trabalho como requisição (referindo-se a *job*), enquanto que o grupo de tarefas relacionadas, que conjuntamente desempenham alguma função no sistema, recebem o nome de tarefa (*task*). A transmissão de pacotes de dados, recuperação de um arquivo, cálculo para um sistema de controle são exemplos de uma tarefa (Liu, 2000).

Existem três modelos principais de tarefas tratadas em sistemas *RT* convencionais: **periódicas**, **esporádicas** e **aperiódicas**. Uma tarefa é dita periódica se ela apresenta várias instâncias (ou iterações) e um período fixo (intervalo regular) entre liberações consecutivas, com as requisições de processamento do mesmo tamanho. Um exemplo seria o de uma tarefa periódica processando sinais de radar a cada 2 segundos. Uma tarefa esporádica possui zero ou mais instâncias, apresentando um espaço mínimo entre duas liberações consecutivas. Por exemplo, uma tarefa esporádica pode executar uma manobra de emergência de uma aeronave quando o botão de emergência é pressionado, com um tempo mínimo de 20 segundos entre duas requisições de emergência. Já uma tarefa aperiódica nada mais é que uma tarefa esporádica com uma freqüência de ocorrência e durações indeterminadas (Cheng, 2002).

Restrições de tempo

Aplicações *RT* são caracterizadas por restrições de tempo, as quais devem ser respeitadas para que se tenha o comportamento temporal desejado ou necessário de uma requisição do sistema. Dentre todos os parâmetros que compõem estas restrições, o tipicamente presente em todas as requisições *RT* é o *deadline*. Ele corresponde ao instante máximo (prazo) em que uma requisição deve ser concluída. Pelo fato do tempo máximo de resposta permitido a uma requisição dado em geral na forma relativa, o parâmetro *deadline*, quando indicando um instante referente ao tempo natural do ambiente, é mais referenciado por *deadline* absoluto (Laplante, 2004; Liu, 2000).

Outras restrições temporais são importantes na definição deste comportamento temporal:

- **Tempo de início**: corresponde ao instante de início do processamento da requisição em uma ativação.

- **Tempo de término**: define o instante de conclusão da execução de uma requisição.
- **Tempo de resposta**: equivale ao tempo entre o recebimento de um estímulo externo pelo sistema (um conjunto de entradas) e a realização do comportamento esperado do sistema (resposta), incluindo a disponibilização de todas as saídas associadas.
- **Tempo de execução**: especifica a quantidade de tempo gasto para completar a execução de uma dada requisição, quando ela executa sozinha e de posse de todos os recursos necessários. Este é um parâmetro que depende diretamente da complexidade da requisição e da velocidade do processador utilizado para sua execução. Ele é útil para se determinar, na maioria das vezes, se uma requisição cumprirá seu *deadline*.
- **Tempo de chegada**: define o instante em que o escalonador toma conhecimento da ativação de uma requisição. Nas periódicas (seção 2.3.3), o tempo de chegada coincide sempre com o início do período de ativação. Requisições aperiódicas (seção 2.3.3), por sua vez, apresentam o tempo de chegada coincidindo com o tempo de requisição do processamento aperiódico.
- **Tempo de liberação** (*release time*): coincide com o instante em que uma requisição está disponível para processamento, o que pode ser modelado como a inclusão da requisição em uma fila de *jobs* prontos para serem executados. Em geral, é assumido que tão logo uma tarefa chegue a mesma será liberada para esta fila. A partir disto ela poderá ser escalonada e executada a qualquer momento, dependendo apenas da disponibilidade de seus dados e do cumprimento das dependências de controle existentes. Tempo de liberação e *deadlines* são dois parâmetros muito utilizados para a distinção de tarefas de sistemas *RT* e tempo-virtual.
- *Release Jitter*: representa a máxima liberação de uma requisição, dado o fato que nem sempre ocorre a coincidência do tempo de chegada com a liberação da requisição.

2.3.2 Sistemas *Hard* e *Soft-RT*

Para sistemas de tempo-virtual, o importante é a exatidão lógica do resultado. Já para sistemas *RT*, tanto a precisão do resultado, quanto a do seu tempo de resposta são importantes. Estes dois tipos de precisão possibilitam a classificação dos sistemas *RT* em *Hard* e em *Soft*.

Em sistemas *Hard-RT*, a exatidão do tempo de resposta é criticamente importante e não pode ser depreciada como alternativa para outros ganhos. Nestes sistemas, o não cumprimento das restrições temporais (como tempo de liberação e *deadline*) acarretaria

conseqüências desastrosas, tal como a possibilidade de colisão caso o comando de parar um trem sofresse um atraso. Em alguns casos a exatidão do resultado pode até sofrer um relaxamento em pró da garantia do tempo de resposta. Um exemplo seria um sistema de piloto automático de aeronave, em que uma coleta de informações não tão precisa da altura, mas dentro dos períodos fixos esperados pelo sistema é preferível à obtenção de dados extremamente precisos, mas obtidos ao custo de um atraso no tempo de coleta (Nissanke, 1997; Liu, 2000).

Para sistemas *Soft-RT* a precisão de tempo é importante, mas não é crítica. O atraso na conclusão de uma tarefa é algo indesejável, mas que se acontecer esporadicamente não trará sérios danos, apenas degradação de desempenho a cada não cumprimento de *deadline*. Para sistemas *Soft* as tarefas são desempenhadas tão rápidas quanto possíveis, podendo ter um relaxamento na precisão de alguns de seus tempos em situações de sobrecarga do sistema.

Uma definição mais formal de sistemas *Hard-RT* (Sha *et al.*, 2004) estabelece que, se $a(t)$ representa a fração de operações de I/O que sofrem um atraso maior que um dado tempo t, então tais sistemas atendem à restrição (2.1):

$$a(t) = 0 \tag{2.1}$$

Por outro lado, sistemas *Soft-RT* referem-se a uma grande classe de aplicações em que o requisito máximo de atraso determinístico não é realmente imprescindível. Em vez disso, é suficiente para o funcionamento adequado do sistema garantir limites estocásticos de tempo de resposta, ou seja, ao longo do tempo de operação, alguns valores estatísticos sejam preservados. Tais sistemas são especificados em termos probabilísticos, e são em geral formulados como em (2.2),

$$a(t) \leq \beta(t) \tag{2.2}$$

em que $a(t)$ representa a fração de requisições (*jobs*) cuja execução, ao longo do tempo, ocorre com um atraso t em relação ao *deadline* especificado, e $\beta(t)$ é uma função de t. Em lugar de garantir um valor máximo de atraso no tempo de resposta, um sistema *Soft-RT* pode determinar um limite superior para a média dos atrasos, de tal sorte que seja possível prever, em um dado intervalo, número de tarefas atendidas.

Outra forma de distinção entre sistemas *Soft* e *Hard-RT* se dá quanto a necessidade ou não de validação do sistema. Um sistema que exige uma validação do cumprimento de suas restrições de tempo é denominado *Hard-RT*. Esta validação é baseada em demonstrações comprovadamente corretas, procedimentos eficientes ou em testes e simulações excessivas. Por outro lado, se nenhuma validação é exigida, ou sendo suficiente apenas uma demonstração de que uma tarefa satisfaz algumas restrições estatísticas, este sistema é tido como *Soft*.

Muitas vezes a distinção entre *Soft* e *Hard* é medida quantitativamente em termos determinísticos ou probabilísticos. Se uma tarefa não permite a perda de *deadline* de suas requisições, então ela é classificada como *Hard*. Por outro lado, se for aceitável a perda ocasional de *deadline*, desde que signifique apenas degradação no desempenho do sistema, então ela é tida como *Soft*. Um exemplo seria o de um sistema multimídia que provesse ao usuário qualidade de serviço garantida. Nele, um filme exibido a uma taxa de trinta quadros por segundo, a diferença nas vezes em que cada quadro fosse exibido e o diálogo apresentado não deveria ser maior que 80 *msec*. Entretanto, como a rede pela qual um filme é transmitido pode incorrer em alguns erros de transmissão, pequenos trechos do filme podem ser retransmitidos, gerando breves falhas. Mas estas falhas podem ser toleradas se ocorrerem muito casualmente, quase raramente, já que o usuário, na maioria das vezes, não desejaria pagar o custo da completa eliminação desse problema. Por essa razão os requisitos de tempo de sistemas multimídia são garantidos por bases estatísticas, como uma média de atrasos/perdas de quadros por minuto menor que 2. Em sistemas *Soft-RT* a qualidade de serviço garantida é flexível, assim como a validação exigida e as restrições de tempo que especificam a qualidade (Liu, 2000).

2.3.3 Escalonamento de Sistemas *RT*

Escalonamento em sistemas *RT* refere-se à especificação da ordem temporal de alocação de requisições para um processador (ou um conjunto de processadores), respeitando as restrições de tempo de cada uma, bem como condições de precedência e a disponibilidade dos recursos. Em sistemas onde a noção de tempo e de concorrência são tratadas explicitamente, conceitos e técnicas de escalonamento formam o ponto central na previsibilidade do comportamento do sistema (Nissanke, 1997).

Em sistemas de tempo-virtual, a meta típica do escalonamento é maximizar a média do *throughput*[9] e/ou minimizar a média do tempo de espera das requisições. No caso de escalonamento sistemas *RT*, a meta corresponde a satisfazer o *deadline* de todas as requisições, garantindo para isso que cada uma possa completar sua execução dentro de seu *deadline* pré-estabelecido (Cheng, 2002).

Classificação do Escalonamento

Quanto ao Problema

Dependendo da natureza do problema tratado, o escalonamento pode ser classificado em diferentes tipos: estático, dinâmico e uma mistura de ambos.

Escalonamento estático assume o conhecimento prévio das características relevantes de todas as requisições, as quais podem ser levadas em conta na hora do escalonamento. Ele

[9]Número de requisições completadas por unidade de tempo.

é aplicável à problemas com requisitos de escalonamento bem conhecidos, como por exemplo freqüência de processamento de requisições e tempos de computação. As prioridades estabelecidas em tais escalonamentos são fixas enquanto durar a aplicação. Entretanto, qualquer mudança nas informações providas ao escalonador, especialmente quando se trata de novas requisições, requer o completo reescalonamento delas. Desse modo algoritmos de escalonamento estáticos não são indicados para situações imprevisíveis. Apesar dessa inflexibilidade, o escalonamento estático apresenta algumas vantagens como baixo custo (já que o escalonamento não é feito em tempo de execução), e grande capacidade de predição (característica relevante em aplicações críticas de segurança).

Por outro lado, algoritmos de escalonamento dinâmicos são projetados para trabalhar com requisições imprevisíveis quanto ao tempo de chegada, e possivelmente com tempo de execução desconhecido. Espera-se de tais escalonamentos uma resposta pronta às necessidades do ambiente, adaptando-se a elas. Apesar da flexibilidade, o escalonamento dinâmico demanda alto custo de processamento para garantir o reescalonamento durante o tempo de execução.

Um escalonamento misto é a mistura dos dois anteriores, escalonando estaticamente as requisições por meio de características previamente conhecidas e ajustando as outras à medida que chegam (Nissanke, 1997).

Quanto ao direito de Preempção

A natureza do escalonamento também pode variar de acordo com a possibilidade ou não de preempção de tarefas quando necessário. Um escalonamento com preempção (*preemptive scheduling*) é aquele em que uma requisição pode ser interrompida e continuada depois sem comprometer sua realização e garantindo qualquer restrição de tempo associada. Apesar de oferecer mais alternativas de escalonamento, o uso de preempção pode demandar um tempo de execução maior por causa da necessidade de troca de contexto. Algoritmos sem preempção não acarretam tal aumento de tempo. Além disso apresentam como vantagens maior simplicidade, maior previsibilidade, facilidade de testes e garantia de acesso exclusivo à recursos e dados compartilhados.

Quanto ao sincronismo

Em específico, escalonamento *RT* se divide em três principais abordagens. A primeira, denominada orientada ao tempo (*Clock-driven* ou *Time-driven*), toma as decisões de qual requisição executar de acordo com instantes previamente estabelecidos para cada uma antes da execução do sistema. Neste caso, geralmente se implementa o escalonador com o uso de *hardware* de tempo, para intervir nos instantes de execução pré-estabelecidos.

Outra abordagem, denominada *Weighted Round-Robim*, é mais comumente utilizada no escalonamento de aplicações de tempo compartilhado (*time-shared*), nas quais as requisições são organizadas de acordo com o modelo de fila *FIFO* quando se tornarem prontas

para execução. Desse modo, uma requisição é selecionada quando estiver no início da fila, podendo executar por no máximo um intervalo de tempo fixo (*slice time*). Caso não termine sua execução neste intervalo (*round*), a requisição é interrompida e realocada ao fim da fila, podendo continuar a execução na próxima vez que for escalonada.

A terceira abordagem é a orientada à prioridade, que toma decisão de escalonamento baseada em eventos (*Event-driven*), como por exemplo os de liberação e conclusão de requisições, além de levar em conta importância individual delas. Ele só faz sentido para requisições passíveis de preempção. Uma requisição $T1$ é dita ter uma prioridade maior sobre outra $T2$ quando esta última tem que ser interrompida para permitir a execução de uma nova requisição por $T1$. Se mais de uma requisição com alta prioridade for requerida simultaneamente, a com maior prioridade dentre todas é executada. Em caso de equivalência de prioridades, uma é escolhida arbitrariamente.

Quando as prioridades das requisições são atribuídas previamente e mantidas fixas, o escalonamento demanda baixo custo de execução e possibilita a implementação em *hardware*. Geralmente ele é encontrado em escalonamento estático. Já em algoritmos de escalonamento dinâmico as prioridades atribuídas às requisições podem variar ao longo do tempo de uma para outra, embora o princípio de atribuição se mantenha. Assim, dada duas requisições $T1$ e $T2$, $T1$ pode ter uma prioridade maior que $T2$ em um dado instante, enquanto $T2$ pode ter prioridade maior em outro instante. As prioridades das requisições são determinadas em tempo de execução (Nissanke, 1997).

O escalonamento a ser focado neste trabalho será o orientado à prioridade, dado sua importância na literatura e pelo fato de cobrir os mais variados aspectos dos comportamentos temporais de aplicações *RT*.

Escalonamento Orientado à Prioridade para requisições Periódicas

As atividades envolvidas nas aplicações de uma maneira geral, sejam de controle de processos ou mesmo aplicações de multimídia, se caracterizam basicamente pelo comportamento periódico de suas ações. Sua propriedade de previsibilidade permite que se obtenha ainda em tempo de projeto garantias de escalabilidade, ou seja, a ordem de execução escolhida para as requisições respeitará as restrições de tempo de cada uma.

As requisições de tarefas periódicas possuem fase (chegada), período, tempo de execução e *deadline*, e o comportamento temporal de uma requisição periódica T_i é descrito pela quádrupla (J_i,C_i,P_i,D_i), sendo:

J_i : *release jitter* da requisição (pior situação de liberação)

C_i : tempo de execução da requisição

P_i : período da requisição

D_i : *deadline* da requisição

Cada ativação de uma requisição periódica é definida a partir de tempos absolutos: tempo de chegada (a_i), tempo de liberação (r_i), tempo de início (st_i), tempo de término (ct_i) e o *deadline* absoluto (d_i). A figura 2.3 ilustra tal comportamento.

$$d_{ik} = (k\text{-}1)P_i + D_i$$
$$r_i = (k\text{-}1)P_i + J_i \quad \text{(pior situação de liberação).}$$

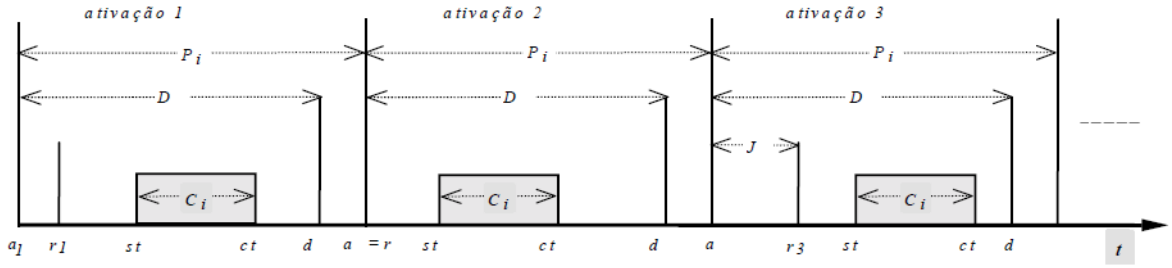


Figura 2.3: ***Ativações de requisições periódica.***

A seguir são apresentados alguns algoritmos clássicos para escalonamento orientado à prioridade de requisições periódicas e independentes.

Algoritmo *RM* (*Rate Monotonic*)

Este algoritmo (Liu, 2000) atribui prioridades às requisições baseado em seus períodos: quanto menor o período, maior a prioridade. A taxa (de liberação das requisições) é inversamente proporcional ao período. Assim, quanto maior a taxa, maior a prioridade. Sempre que os *deadlines* das requisições são proporcionais aos seus períodos, o RM é dito ótimo entre os escalonadores de prioridade fixa na sua classe de problema, ou seja, nenhum outro algoritmo da mesma classe pode escalonar um conjunto de requisições que não seja escalonável pelo *RM* (Nissanke, 1997).

A análise de escalabilidade no *RM* é baseada no cálculo de utilização. Para que n requisições tenham o atendimento de suas restrições temporais quando escalonadas pelo *RM*, a condição de suficiência (2.3) deve ser satisfeita.

$$U = \sum_{i}^{n} \frac{C_i}{P_i} \leq n\left(2^{1/n} - 1\right) \tag{2.3}$$

Algoritmo *DM* (*Deadline-Monotonic*)

Outro algoritmo de prioridade fixa é o *DM*(Liu, 2000), que atribui prioridades às requisições de acordo com seus respectivos *deadlines*: quanto menor o *deadline* relativo, maior a prioridade. Ele estende o modelo de requisições *RM*, sendo mais flexível ao assumir *deadlines* relativos menores ou iguais aos períodos das requisições ($D_i \leq P_i$).

Quando o *deadline* relativo de toda requisição for proporcional ao seu período, os algoritmos *RM* e *DM* serão idênticos. Porém, quando os *deadlines* relativos forem arbitrários,

o algoritmo *DM* apresentará melhor desempenho, já que ele pode apresentar escalonamento factível quando o *DM* falha, enquanto o algoritmo *RM* sempre falha quando o *DM* falha (Nissanke, 1997).

Algoritmo *EDF* (*Earliest Deadline First*)

O algoritmo *EDF*(Liu, 2000) realiza o escalonamento das requisições em tempo de execução (dinâmico), atribuindo prioridades à elas segundo seus *deadlines* absolutos. A mais prioritária é a requisição que tem o *deadline* mais próximo do tempo atual, ou seja, a priorização se dá baseada na urgência das requisições. A fila de requisições prontas é reordenada a cada nova requisição que chega, a fim de atualizar a nova distribuição de prioridades. A fim de um escalonamento pelo algoritmo *EDF* ser factível, a condição (2.4) deve ser satisfeita.

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1 \tag{2.4}$$

EDF é considerado um algoritmo ótimo quando aplicado a um escalonamento monoprocessado e preemptivo, no seguinte sentido: se um conjunto de requisições independentes, cada qual caracterizada por um tempo de chegada, tempo de execução, e um *deadline*, pode ser escalonado de tal forma que todas as requisições sejam atendidas dentro de seus *deadlines*, então o *EDF* escalonará esse conjunto de requisições de tal forma a cumprir todos os seus *deadlines* (Nissanke, 1997).

Escalonamento Orientado à Prioridade para requisições Aperiódicas e Esporádicas

Requisições aperiódicas são dirigidas por eventos e podem ter restrições de tempo *Hard*, *Soft* ou sem restrições de tempo. As requisições aperiódicas associadas com eventos críticos (possuindo restrições de tempo *Hard*) são caracterizadas por um intervalo mínimo entre suas ativações, limitando a carga aperiódica. Estas requisições são identificadas como esporádicas, possuem um comportamento temporal determinístico, facilitando assim a obtenção de garantias em tempo de projeto. Uma requisição esporádica é descrita pela tripla (C_i,D_i,min_i), sendo:

C_i : tempo de execução;

D_i : *deadline* relativo da requisição, medido a partir do instante de requisição do processamento aperiódico (chegada da requisição esporádica);

min_i : intervalo mínimo entre duas requisições consecutivas da requisição esporádica.

Estes parâmetros estão ilustrados na figura 2.4, a qual apresenta a requisição esporádica (C_i,D_i,min_i) com duas requisições.

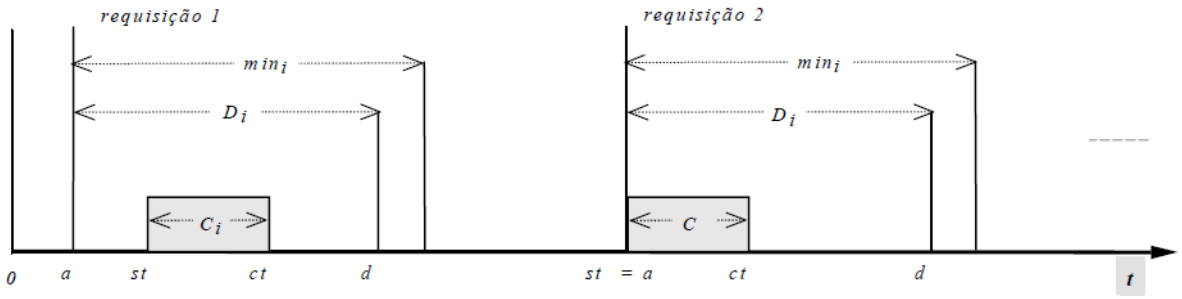


Figura 2.4: ***Ativações de uma requisição aperiódica.***

A descrição de uma requisição aperiódica pura se limita apenas as restrições C_i e D_i. Elas se caracterizam por não possibilitarem a determinação a *priori* da taxa máxima de chegada dos eventos. Devido a isso, é possível a obtenção de garantias dinâmicas ou, ainda, a utilização de técnicas de melhor esforço, executando-as conforme a disponibilidade do processador.

O escalonamento de requisições aperiódicas tenta completá-las tão logo quanto seja possível, mas sem ocasionar a perda de *deadline* de requisições periódicas ou esporádicas. Basicamente existem dois tipos de algoritmos para o escalonamento de requisições aperiódicas: *bandwidth-preserving* e *slack-stealing*(Liu, 2000). Já o escalonamento de requisições esporádicas é baseado no tempo de execução e *deadline* de cada nova tarefa esporádica, em que o escalonador decide se aceita ou rejeita a tarefa. Se a requisição for aceita, ela é escalonada de forma que complete no tempo desejado sem causar impacto negativo nas requisições periódicas e esporádicas previamente aceitas. Caso seja impossível para alguma requisição esporádica atender seu *deadline*, consideram-se duas alternativas: rejeitar a requisição esporádica que não pode completar no tempo determinado, ou aceitar todas as requisições esporádicas e permitir a algumas delas finalizar atrasada.

Alguns algoritmos de escalonamento para estes casos são para políticas baseadas em prioridades fixas (como *Rate Monotonic*) e conceito de servidores, dos quais se pode citar:

Servidor de *Background*

Este modelo é bastante simples, e tem como política escalonar requisições aperiódicas e executá-las somente quando não existem requisições periódicas ou esporádicas executando ou pendentes. (Liu, 2000) A determinação de prioridades é feita atribuindo, de acordo com o algoritmo *RM*, as prioridades mais altas para as requisições periódicas, e destinando as prioridades mais baixas para as aperiódicas. Com isso, o tempo de respostas para requisições aperiódicas é alto e, se a carga de periódicas for alta, o serviço de *Background* é baixo ou não freqüente, só sendo aplicável quando as requisições aperiódicas não são críticas e a carga periódica não é alta (Nissanke, 1997).

Servidora de *Polling*

Neste esquema define-se uma requisição periódica denominada Servidora de *Polling* (*PS* - *Polling Server*) para atender a carga aperiódica. Com um espaço periódico estabelecido, e com um tempo de computação pré-definido, a *PS* executa as requisições aperiódicas pendentes, de acordo com atribuições *RM*. Quando não houver Polling Server aperiódicas disponíveis, a *PS* se suspende até que uma nova chegue, no próximo período. Neste caso, a sua capacidade de tempo de execução é entregue para a execução de requisições periódicas pendentes. Se comparada com a abordagem de *Background*, a de *Polling Server* melhora o tempo de resposta médio de requisições aperiódicas, porém não fornece tempo de resposta imediato para o processamento delas. O tempo de resposta de requisições aperiódicas depende do período e do tempo de execução definidos pela *PS* (Nissanke, 1997).

Deferrable Server

Este algoritmo também é baseado na idéia da criação de uma requisição periódica (com alta prioridade) para atender as aperiódicas, e tem por diferencial conservar o tempo de execução destinado a requisição servidora quando não existem pedidos pendentes no momento da ativação dela. Desta forma, se alguma requisição aperiódica chegar durante o período destinado à *PS*, ela poderá ser atendida dentro do tempo de execução restante, que é pleno no início do período. Esse diferencial possibilita a melhora do tempo médio de resposta das requisições aperiódicas (Nissanke, 1997).

Slack Stealing

Este algoritmo tem por idéia principal não prover nenhum benefício para que as requisições periódicas terminem antes do tempo. Em vez de criar uma tarefa periódica para atender as requisições aperiódicas, ele cria uma requisição passiva, conhecida como *slack stealer*, que se apropria de todo tempo de processamento possível das periódicas, sem causar a perda de *deadlines* das mesmas. Assim, quando uma requisição aperiódica chega o *Slack Stealer* se apropria de todo tempo livre (que sobra) das periódicas, destinando-o para execução de requisições aperiódicas (Nissanke, 1997).

2.4 Simulação de Sistemas

Simulação é a reprodução da composição dinâmica (i.e. operações) de um **sistema** do mundo real sobre o tempo, descrita em termos de atividades, processos, e eventos (MacDougall, 1989). Pela observação do histórico artificial por ela gerada busca-se inferir características operacionais de um sistema real, estudando cientificamente seu comportamento. Isso é útil quando a complexidade de um sistema real inviabiliza seu estudo por experimentação; quando se pretende analisar em isolado a influência de mudanças

organizacionais ou ambientais no comportamento de um sistema, ou ainda para testar novos projetos ou políticas antes de serem implementados (Banks *et al.*, 2000).

A representação de comportamento é feita a partir de um **modelo** de simulação, construído a partir de um conjunto de suposições operacionais de um sistema real. Essas suposições expressam de forma matemática, lógica ou simbólica as relações entre as diversas **entidades**[10] do sistema. Uma vez construído e validado, o modelo pode ser utilizado para investigar uma ampla variedade de questões do tipo "o que aconteceria se" a cerca de um sistema real. A solução desses modelos pode ser feita de forma analítica, em que se utiliza a lógica dedutiva da matemática, ou por meio de simulação (geralmente computacional), que emprega métodos numéricos na análise (Law & Kelton, 2006).

Alguns conceitos e nomenclaturas do tema são necessários para a estruturação e análise do modelo de simulação a ser empregado. Sendo assim, define-se como **estado** de um sistema a coleção de variáveis necessárias para descrevê-lo em um dado período estudado. Em um sistema bancário, por exemplo, o número de caixas ocupados, de clientes esperando em fila, representam variáveis de estado. De acordo com o tipo de estados que predominam no sistema, ou no foco da análise, pode-se classificá-lo em **discreto** e **contínuo**. Um sistema discreto é aquele cujo estado sofre mudanças instantâneas em pontos distintos de tempo, como por exemplo o número de clientes em uma fila. Um sistema contínuo é aquele cujas mudanças ocorrem continuamente ao longo do tempo, como o volume de água em uma hidrelétrica.

Como modelar significa abstrair características de um sistema real, incorporando apenas as vistas como fundamentas pela ótica de interesse do estudo, constitui como etapa inicial a especificação do nível de detalhe que se pretende com esse estudo. Sistemas computacionais, em específico, podem ser modelados em vários níveis de detalhe, de acordo com os elementos de abstração utilizados: circuito, portas-lógicas, registradores, e sistema. Assim, pode-se melhor formalizar o tipo de simulação utilizado neste trabalho: **simulação de eventos discretos no nível de sistema**.

A modelagem de um sistema não se restringe à descrição da estrutura estática de um sistema, mas abrange também a representação de sua **composição dinâmica**[11], que pode ser descrita em termos de:

- **Atividades**: menor unidade de trabalho na visão que se tem do sistema, tendo um tempo de execução a ela associada;

- **Processos**: conjunto de atividades relacionadas logicamente, com o tempo de execução (ignorando concorrência) formado pela soma das execuções e tempos de atrasos de cada uma dessas atividades;

[10] Objetos de interesse que atuam e interagem em conjunto com algum intuito lógico.
[11] Modo que o sistema realiza trabalho.

- **Eventos**: representam a mudança de estado de algum objeto do sistema, como por exemplo uma *CPU*, que passa do estado ocioso para ocupado. Essa mudança é resultado da ação de alguma atividade. A inicialização das atividades é acionada pelos eventos.

Essas entidades são utilizadas para estruturar o modelo de simulação a ser construído. Dependendo em qual delas a descrição do sistema se baseia, as linguagens de simulação são classificadas em: orientadas a atividade, a evento, ou a processo. Dessas, as orientadas a processo e a evento são as mais utilizadas. Linguagens orientadas a processo, como *ASPOL*(MacDougall & McAlpine, 1973), *CSIM*(Schwetman, 1986), e *SIMULA*(Birtwhistle *et al.*, 1973), são mais recomendadas para implementar modelos de simulação de grande-escala, devido a grande similaridade entre modelo e sistema. Enquanto que linguagens orientadas a evento, como *NS2*(Heidemann *et al.*, 2007) e *SIMPACK*(Fishwick, 1992), são indicadas para modelos de pequena e média-escala, cuja representação em um único nível e a visão geral do sistema não significam problemas (MacDougall, 1989).

Representação de Trabalho

Além da construção do modelo de sistema, a representação da carga de trabalho a ser submetida a ele é outra tarefa importante. Esses dados de entrada descrevem o trabalho a ser desempenhado por um sistema. Eles podem ser gerados probabilisticamente dentro do programa de simulação, ou podem ser gerados externamente, como em simulação orientada a *trace*, na qual se utiliza o registro da seqüência de execução de um sistema real. Para as simulações executadas neste trabalho utilizou-se dados gerados probabilisticamente.

Para utilizar tais dados faz-se uma abstração no mesmo nível de detalhes do modelo de sistema desenvolvido. Assim, além de analisar se o nível de detalhe adotado é o realmente necessário para resolver o problema, é importante também verificar se existem dados de entrada disponíveis para o nível escolhido.

A descrição desses dados de entrada é denominada caracterização de carga (*workload characterization*), e consiste em examinar parâmetros comportamentais para determinar se existe, e de que forma se dá, correlação entre as variáveis de estado. A descrição desse comportamento é feita por meio de uma distribuição de probabilidade, que nem sempre é fácil de ser encontrada. Uma discussão detalhada das várias distribuições de probabilidade, seus parâmetros, e estimativas para esses parâmetros é feita em (Law & Kelton, 2006).

Dentre a distribuições utilizadas para descrever a carga de trabalho em sistemas geradores de fila, a exponencial negativa (ou simplesmente exponencial) é a mais importante. Utiliza-se ela para descrever variáveis contínuas como valores de intervalos de chegada e tempos de serviço. Essa distribuição tem como característica a ausência de memória

(*memoryless*[12]), essencial para obter soluções analíticas de vários problemas de fila. Para a análise de sistemas computacionais essa propriedade é importante pois evita a necessidade de armazenamento todos os eventos passados, facilitando a solução.

Simulação de Filas

Uma técnica amplamente utilizada para modelagem de sistemas com ocorrência de filas (servidores *Web*, por exemplo) é a de Redes de Fila. A representação se dá por meio de entidades denominadas centros de serviços e um conjunto de entidades chamadas de usuários[13], os quais recebem serviços desses centros e circulam pelo sistema.

Um centro de serviço consiste de um ou mais servidores, correspondentes aos recursos do sistema modelado, e uma área de espera (i.e. fila), de capacidade finita ou infinita, para usuários provenientes de uma dada população que estão requisitando serviços. A *CPU* de um computador é um exemplo de recurso. Esses recursos podem ser classificados em ativos ou passivos, de acordo com a capacidade de realizar ou serviços, respectivamente. A figura 2.5 ilustra os diferentes tipos de centros de serviço (Soares, 1990).

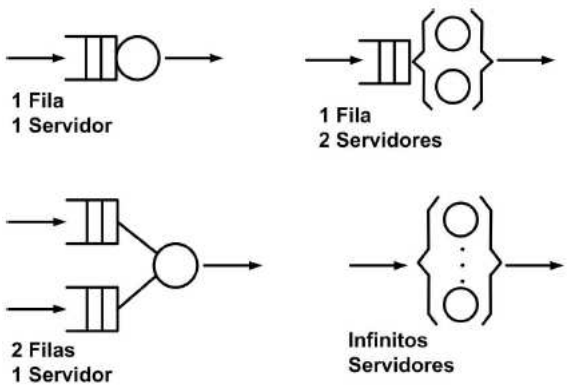


Figura 2.5: ***Centros de Serviço.***

De acordo com o comportamento de entrada e saída dos usuários, os modelos de Redes de Filas podem ser classificados em:

- **Modelos Abertos**: Os usuários entram no sistema e, obrigatoriamente, saem do mesmo;
- **Modelos Fechados**: O número de usuários que circulam pelo sistema é fixo, de forma que nem entram e nem saem usuários;
- **Modelos Mistos**: Para certas classes de usuários o modelo é aberto, enquanto que para outras é fechado.

[12]Qualquer probabilidade derivada de um conjunto de amostras aleatórias é distinta e não tem informação (i.e. "memória") das amostras anteriores. A probabilidade de uma chegada em um intervalo t, por exemplo, depende apenas da duração de t, e não de quando a chegada anterior ocorreu.

[13]O termo usuário é usado de uma forma genérica e pode designar tanto uma pessoa, uma requisição *Web*.

A figura 2.6 ilustra os três modelos de Rede de Filas (Lazowska *et al.*, 1984).

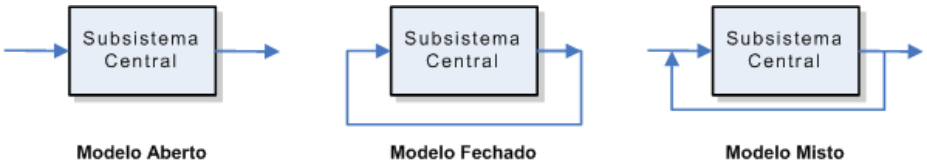


Figura 2.6: ***Tipos de Modelo.***

A descrição das características desses modelos é feita pela notação Kendall, representada pela tuple[14] $A/S/c/k/m/D$, cujos parâmetros significam:

A Distribuição que descreve o intervalo de chegada dos clientes;

S Distribuição do tempo de serviço;

c Número de servidores;

k Número máximo de clientes suportados pelo sistema;

m Número total de clientes disponíveis na fonte;

D Disciplina de fila a ser utilizada.

Geralmente omitem-se os parâmetros k ou m quando assumem valores infinitos. O parâmetro D também é omitido quando representa a disciplina de fila *FIFO*. Alguns símbolos normalmente utilizados para descrever distribuições são:

D Distribuição determinística (valores de intervalos de chegada e tempo de serviço constantes);

M Distribuição exponencial;

E_k Distribuição Erlang de estágio k;

H_k Distribuição hiper-exponencial de estágio k;

G Distribuição arbitrária.

2.4.1 Teoria de Filas

A técnica de simulação Redes de Fila se baseia na Teoria de Filas, que é um ramo da probabilidade que estuda sistemas geradores de filas (i.e. Sistemas de Filas). Com essa teoria é possível fazer a análise matemática de diversos processos relacionados, como os de chegada em fila, espera em fila, e os de oferecimento de serviço pelos servidores.

[14]Conjunto de variáveis.

Essa análise permite derivar e calcular várias medidas de desempenho, incluindo tempo de espera em fila ou no sistema, comprimento de fila, número de elementos recebendo serviço, e a probabilidade de ocorrência de certos estados do sistema, como ocioso, ocupado, tendo algum servidor disponível ou ter que esperar certo instante de tempo para ser servido. Essas medidas de desempenho podem ser obtidas por intermédio de algumas relações algébricas, que serão apresentadas nessa seção (Lazowska *et al.*, 1984; Jain, 1991).

A seguir são apresentadas a notação e a definição de algumas variáveis utilizadas em rede de filas:

τ: tempo entre chegadas;

λ: taxa média de chegada, definida por $\frac{1}{E[\tau]}$, onde $E[\tau]$ é o tempo médio entre chegadas;

s: tempo de serviço por usuário;

μ: taxa média de serviço por servidor, definida por $\frac{1}{E[s]}$, onde $E[s]$ é o tempo médio de serviço. A taxa de serviço total para c servidores é $c\mu$;

n: número de usuários no sistema, também chamada de comprimento da fila. Essa variável inclui tanto os usuários que estão recebendo serviço quanto os que estão esperando na fila;

n_q: número de usuários esperando para receber serviço. Sempre n_q é menor que n;

n_s: número de usuários recebendo serviço;

r: tempo de resposta ou tempo no sistema. Inclui tanto tempo de espera quanto tempo de recebimento de serviço;

w: tempo de espera ou tempo de fila, isto é, o intervalo de tempo entre a chegada e o instante em que inicia o serviço;

T: período de tempo observado;

A: número de chegadas de usuários no tempo T;

C: número de serviços completados no tempo T;

B: tempo em que o servidor permaneceu ocupado durante T;

Algumas equações válidas para sistemas de fila única são apresentadas a seguir:

- O número de usuários no sistema é dado pela soma dos usuários que estão na fila esperando pelo recebimento de serviço e aqueles que estão recebendo serviço, isto é, $n = n_q + n_s$(Jain, 1991);

- O tempo de resposta é o tempo gasto pelo usuário no sistema. Ele é dado pela soma do tempo de espera do usuário na fila e o tempo de serviço recebido por esse usuário, isto é, $r = w + s$(Jain, 1991);

Algumas relações algébricas que, por serem de grande relevância à avaliação de desempenho de modelos de rede de filas, são chamadas de **Leis Operacionais**, são apresentadas a seguir. Essas **Leis Operacionais** são válidas tanto para sistemas de fila única quanto para sistemas com uma rede de várias filas.

Taxa de chegada: $\lambda = \frac{A}{T}$;

Throughput: $X = \frac{C}{T}$;

Utilização: $\rho = \frac{B}{T}$;

Tempo médio de serviço: $S = \frac{B}{C}$;

Lei de Utilização: $\rho = XS$, derivada das equações acima;

Lei de *Little*: Seja L o número médio de usuários no sistema durante um período de observação, e R o tempo médio que esses usuários gastam no sistema. Define-se r_i como o tempo gasto pelo i-ésimo usuário, então $R = \frac{\sum r_i}{C}$. O número médio de usuários no sistema é então $L = \frac{RC}{T}$. Como $X = \frac{C}{T}$, a Lei de *Little* pode ser expressa como $L = XR$(MacDougall, 1989);

Lei do tempo de Resposta: Essa lei é comumente utilizada no contexto de um sistema de tempo compartilhado (*timesharing*). Seja N o número de terminais no sistema, Z e R, respectivamente o tempo de pensar e o tempo médio de resposta de um terminal, e seja X o processamento do sistema. Tem-se então $R = \left(\frac{N}{X}\right) - Z$(MacDougall, 1989);

Se o período de observação é suficientemente longo, o número de chegadas deve ser aproximadamente igual ao número de serviços completados ($A \approx C$), sendo assim, pode-se assumir que $\lambda \approx X$ (isto é, a taxa de chegada do sistema é igual ao *throughput* do sistema). Isso é chamado de Balanceamento de Fluxo(MacDougall, 1989).

2.4.2 Probabilidade e Estatística para Simulação

Para uma análise adequada de um modelo de simulação utiliza-se uma série de conceitos e ferramentas pertencentes à probabilidade e estatística, que são apresentados de forma sucinta a seguir.

Variável Aleatória

Uma variável aleatória é um mapeamento do resultado de um evento em um número. O estado de um servidor, por exemplo, pode ser modelado como uma variável aleatória que pode assumir o valor 1 quando o servidor estiver funcionando ou zero caso contrário. Neste exemplo, a variável mapeia o evento estado do servidor (desligado ou funcionando) em um número (0 ou 1). Uma variável aleatória pode ser discreta ou contínua. Uma variável é discreta quando o conjunto de possíveis valores é enumerável, como no exemplo acima. Outro exemplo de variável discreta é o número de pacotes *IP* descartados por um roteador em um intervalo de tempo. Já uma variável contínua pode assumir qualquer valor dentro de um determinado intervalo, ou seja, o conjunto dos possíveis valores não é enumerável. O atraso observado no estabelecimento de uma conexão entre um usuário e um servidor em uma rede pode ser modelado como uma variável contínua(Kamienski *et al.*, 2002).

Eventos Independentes

Dois eventos são chamados independentes se a ocorrência de um evento não afeta a probabilidade de ocorrência do outro. Durante a realização de uma simulação é extremamente importante identificar a relação de dependência entre as variáveis envolvidas. O número de pacotes descartados por um roteador e o número de pacotes que chegam em um intervalo de tempo são variáveis dependentes. Já o número de chamadas telefônicas que chegam a uma central num intervalo de tempo e o tempo de duração das chamadas são variáveis independentes.(Kamienski *et al.*, 2002)

Distribuição de Probabilidade e Função densidade

Considere o exemplo da variável X que modela o estado de um servidor, e que $P[X = 1] = p_1 0 P[X = 1] = p_2$. O conjunto $p1, p2$ é chamado de distribuição de probabilidade da variável discreta X. A distribuição de probabilidade determina completamente o comportamento da variável, uma vez todos os possíveis valores da variável têm suas probabilidades especificadas(Kamienski *et al.*, 2002).

No caso de uma variável contínua X , define-se uma função de distribuição acumulada $F_X(x)$ que determina a probabilidade da variável assumir um valor menor ou igual a um determinado valor x:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(u)du, \tag{2.5}$$

onde $f_X(x)$ é a função densidade de probabilidade (ou *probability density function - PDF*) de X . Assim, como a distribuição de probabilidade define completamente uma variável discreta, a função densidade define o comportamento de uma variável contínua.

Média ou Valor Esperado

A média ou valor esperado de uma variável aleatória X é dado por:

$$E(X) = \sum_{i=1}^{n} p_i x_i = \int_{-\infty}^{+\infty} x f_X(x) dx. \tag{2.6}$$

O somatório é utilizado para o caso de uma variável discreta. A média é um parâmetro muito utilizado na prática para obter estimativas do comportamento de um sistema. Por exemplo, em uma rede de computadores parâmetros comumente utilizados para avaliação de desempenho são vazão média e atraso médio. Aspectos de como obter esse parâmetro e qual a significância do seu valor são abordados mais adiante na seção sobre estimação de parâmetros(Kamienski *et al.*, 2002).

Variância e Desvio Padrão

Em determinadas situações a descrição do comportamento do sistema utilizando apenas a média não é suficiente. Na realidade, a média $E[X]$, não contém nenhuma informação sobre o espalhamento dos dados, ou seja, quão distante as amostras estão do valor médio. A média dos dois conjuntos $A = 5, 10, 15$ e $B = 0, 10, 20$ é a mesma, mas o espalhamento é diferente. No caso de aplicações multimídia na *Internet*, por exemplo, medir o atraso médio em uma conexão não é suficiente para avaliar a qualidade do serviço, pois um valor médio razoável pode ser resultado da combinação de atrasos bastante elevados e atrasos pequenos. Essa grande variação do atraso pode ser mais prejudicial para aplicação do que um atraso médio elevado. Nesse caso, um parâmetro para medir a dispersão dos dados em relação ao valor médio é extremamente importante(Kamienski *et al.*, 2002).

A variância e o desvio padrão são parâmetros que medem a dispersão dos dados em relação à média. A quantidade $(X - E(X)_2)$ representa o quadrado da distância entre X e sua média e o valor médio dessa quantidade é chamado variância de X:

$$Var(X) = E[(X - E(X))_2] = \sum_{i=1}^{n} p_i (x_i - E(X))_2 = \int_{-\infty}^{+\infty} (x_i - E(X))_2 f_X(x) dx. \tag{2.7}$$

O quadrado da distância é utilizado para podermos computar a dispersão tanto de valores acima, como abaixo da média. A variância é tradicionalmente denotada por σ_2. A raiz quadrada da variância é chamada de desvio padrão e é representado por σ.

Amostragem de dados e estimação de parâmetros

As características do conjunto de dados obtidos através das ocorrências de uma variável aleatória são representadas por parâmetros como valor esperado, variância e desvio padrão, por exemplo. Na maioria dos problemas reais, estes parâmetros são desconhecidos

e para obtê-los, precisaríamos repetir o experimento um número infinitamente grande de vezes, o que é impossível. Na prática, estes parâmetros são aproximados por valores obtidos através da amostragem de alguns valores entre o conjunto das ocorrências da variável. Estas estimativas são chamadas de estatísticas. A média aritmética (média amostral) de um conjunto $x_1, x_x, \ldots, x_n$ de ocorrências de uma variável X , representada por $\overline{x}$, é uma estimativa (estatística) para o valor esperado $E(X)$. É importante a distinção entre um parâmetro e uma estatística, porque o parâmetro é um valor fixo, enquanto uma estatística é uma variável aleatória.

Considere o exemplo de uma variável aleatória T que representa o tamanho dos pacotes *IP* que trafegaram em um segmento de rede em um determinado intervalo de tempo. Pela especificação do protocolo *IP*, sabe-se que um pacote tem um tamanho mínimo de 48 *bytes* e um máximo de 65535 *bytes*, logo T pode assumir qualquer valor inteiro no intervalo $48, \ldots, 65535$. No entanto, não sabemos a distribuição de T , nem muito menos o seu valor médio $E[T]$. Mas, coletando alguns pacotes e medindo seus tamanhos formamos uma amostra, por exemplo: $100, 124, 64, 75, 340, 240, 294$. Através dessa amostra podemos calcular a média amostral que serve de estimativa para $E[T]$.

Do exemplo acima, podemos concluir que se obtivermos k amostras de tamanho n da variável aleatória T, teremos k médias amostrais diferentes que são estimativas para a média de T. O próximo passo é obter uma única estimativa para a média a partir das k estimativas(Kamienski *et al.*, 2002).

Intervalo de confiança

Na realidade não é possível encontrar uma estimativa perfeita para a média a partir de um número finito de amostras de tamanho finito. Neste caso, a melhor opção é obter limites probabilísticos, por exemplo, c_1 e c_2 , de forma que seja alta a probabilidade $(1-\alpha)$ de a média exata pertencer ao intervalo $[c_1, c_2]$:

$$P(c_1 \leq E(X) \leq c_2) = 1 - \alpha \tag{2.8}$$

O intervalo (c_1, c_2) é chamado intervalo de confiança para a média, α é chamado nível de significância e $1 - \alpha$ é chamado coeficiente de confidência. Tipicamente, o intervalo de confiança é expresso em termo de um percentual próximo de 100%, por exemplo, 90% ou 95%; enquanto o nível de significância α é expresso como uma fração e é geralmente próximo de zero, por exemplo, $0,05$ ou $0,1$(Kamienski *et al.*, 2002).

2.5 Considerações Finais

Neste capítulo mostrou-se que embora o modelo atual de serviços da *Internet* (o de "melhor esforço") tenha funcionado e ainda funcione bem para vários tipos de aplicações,

tecnologias de serviços *online* emergentes descaracterizam hoje esse cenário. Assim, para oferecer novos níveis de confiabilidade exigidos com o crescimento e diversificação das aplicações suportadas pela *Internet*, o modelo de serviços *QoS* tem sido utilizado.

Esse modelo introduz mecanismos de controle que realizam um gerenciamento dinâmico de recursos, buscando-se com isso oferecer garantias de cumprimento dos requisitos operacionais das novas aplicações. Dentre os principais componentes desse modelo de serviços estão: garantia de desempenho e diferenciação de serviço. Esses componentes embasam as duas principais arquiteturas com suporte à *QoS* propostas pela *IETF*: Serviços Integrados (*IntServ*) e Serviços Diferenciados (*DiffServ*). Ainda no contexto de *QoS*, mostrou-se a importância de oferecimento de *QoS* também no nível de aplicação, em específico servidores *Web*.

Para estabelecer parâmetros de serviço em termos temporais utilizam-se conceitos de Tempo Real. Assim definiu-se alguns conceitos preliminares para o entendimento dessa área, outros importantes para definição e classificação de sistemas do gênero, além de apresentar alguns algoritmos mais utilizados por pesquisas e desenvolvimento em Tempo Real.

Por fim, para o estudo científico de um sistema *Web* utilizou-se a teoria de modelagem representá-lo, e a simulação como técnica de análise do modelo e da política proposta. Conceitos gerais de Sistemas e Redes de fila foram apresentados, como nomenclaturas, relações algébricas e classificação de modelos. E como ferramentas oferecidas pela probabilidade e estatísticas são de importância para uma análise adequada de sistemas, seus principais conceitos também foram abordados.

Com essa fundamentação teórica pode-se entender melhor a área e as técnicas necessárias para uma melhor fundamentação do trabalho proposto, como os conceitos de Qualidade de Serviço, que embasa o projeto; teoria de Tempo Real, importante devido o enfoque temporal dos parâmetros de serviço; e conceitos de Simulação de Sistema, utilizada para análise da política proposta.

Capítulo

3

Trabalhos relacionados

3.1 Considerações Iniciais

Como discutido na seção 2.2.3, a provisão de qualidade de serviço será mais abrangente e propiciará melhores resultados se houver a cooperação de todos os elementos do sistema, inclusive do nível de aplicação, como servidores *Web*, foco desta pesquisa. Sendo assim, alguns trabalhos abordando *QoS* na *Web* merecem destaque, seja pelo embasamento teórico para o desenvolvimento de alternativas que venham a complementar as soluções já propostas, como também para orientar sobre a necessidade de pesquisas em campos pouco explorados. Trabalhos utilizando conceitos de outras áreas, como Tempo Real e Controle Realimentado, também são analisados com, o intuito de fornecerem mais ferramentas e embasamento no desenvolvimento de novas abordagem de *QoS* na *Web*.

3.2 Trabalhos Relacionados

Dentro do tema *QoS* na *Web*, têm sido preponderantes pesquisas abordando: controle de admissão e diferenciação de serviço. A primeira tem sido amplamente empregada como mecanismo de proteção contra degradação de desempenho durante sobrecarga do sistema, visando com isso assegurar a qualidade dos serviços oferecidos às diversas classes do sistema. A segunda, basicamente tem por objetivo garantir diferenciação e isolamento entre as classes de serviço para uma distribuição mais justa dos recursos do sistema, tentando oferecer uma fração mínima de recursos independentemente de outras classes.

Kanodia & Knightly (2003), por exemplo, propõem um modelo de serviço *Web* fundamentado na diferenciação de classes e controle de admissão como forma de gerenciar latências de resposta contratadas, e ao mesmo tempo maximizar a utilização de recur-

sos. Além de tentar manter valores individuais das classes, a técnica também realiza um controle adaptativo de latências entre as classes. O modelo proposto é validado por meio de simulação orientada a *traces*. Os resultados mostraram que o algoritmo, por realizar abstração de recursos em alto nível, pode ser utilizado em conjunto com sistemas operacionais dotados de mecanismos de *QoS*, e explorando suas funcionalidades oferecer altas taxas de utilização de servidor, bem como satisfazer as especificações de serviços das várias classes.

Chen & Heidemann (2003) propõem um algoritmo denominado *SFD* (*Short Flow Differentiating*) que prioriza tráfegos mais curtos, como os de rajada, e "preenche" as lacunas existentes com tráfego longo, pois estudos mostraram que tráfegos curtos fluem mais rapidamente pela rede. O intuito é melhorar o desempenho de tráfego *Web* interativo, que correspondem a uma grande parcela do tráfego da *Internet*. A priorização de tráfego foi feita com o uso de *DiffServ*. O algoritmo foi avaliado por meio de simulação, e os resultados mostraram melhora de mais de 30% na latência de resposta das transmissões mais curtas. Apesar de ser um trabalho que enfoca *QoS* no nível de rede, pode-se perceber a importância e o impacto da priorização de serviços mais curtos, pois esse tipo de tráfego corresponde a mais de 80% das respostas *Web*.

Outro trabalho que merece destaque é o de Elnikety *et al.* (2004), no qual propõem um método transparente para controle de admissão e escalonamento de requisições em servidores *Web* multicamadas[1]. Implementado externamente ao sistema, em um *proxy* entre o servidor de aplicações e a base de dados, o método mantém estimativas de custo de execução por tipos de requisições. Com essas estimativas, obtidas por meio de medições *online*, mais o conhecimento da capacidade total do sistema, é possível controlar a admissão de requisições. Para obter melhores tempos de resposta o método propõe ainda ordenação da fila de admissão na forma da política *SJF* (*Short Job First*). Elnikety avalia sua proposta por meio de experimentos orientados a *traces*.

Pesquisas abordando *QoS* em termos relativos têm sido preponderantes. O trabalho de Teixeira *et al.* (2005), por exemplo, propõe uma arquitetura de servidor *Web* capaz de fornecer serviços diferenciados a seus clientes de acordo com suas características de demanda. Nela são propostos mecanismos de diferenciação de serviços baseados em prioridades e um módulo de controle de admissão com o objetivo de garantir estabilidade na qualidade de serviço oferecida. A arquitetura basicamente é composta por um módulo classificador de requisições, por um controle de admissão e um *cluster* de servidores *Web*. Quando admitida, a requisição é atribuída a um dos nós do *cluster* segundo o algoritmo de escalonamento ou diferenciação de serviços vigente. Outra contribuição desse trabalho é a proposta e análise de um mecanismo de diferenciação de serviços denominado *PRIAdap* (Prioridades Adaptativo), que percorre certo número k de posições na fila em busca de

[1]Servidor formado por um *front-end*, responsável pela formatação e manipulação de requisições; servidor de aplicações, o qual trata requisições dinâmicas; e um *back-end*, representando a base de dados.

requisições de uma dada classe. Esse parâmetro é denominado *look-ahead*, o qual permite regular o nível de priorização do sistema, determinando assim o quão rigoroso será o esquema de prioridades empregado. A arquitetura e os mecanismos são validados por meio de simulação orientada a *traces*.

O trabalho de Estrella *et al.* (2006) estende o de Teixeira ao incluir um mecanismo de negociação de serviço no módulo de controle de admissão, proporcionando com isso melhores médias de tempo de resposta e menores taxas de descarte de requisições, melhorando a qualidade de serviço oferecida.

Em (Traldi *et al.*, 2006) também se aborda *QoS* em termos relativos, propondo dois novos algoritmos de diferenciação de serviços com o intuito de prover qualidade de serviço no nível de aplicação: *RSVAdap* e *WFQ*. O *RSVAdap* é um algoritmo de particionamento de recursos proposto a partir do trabalho de Teixeira (2004), com a diferença de que realiza a alocação de recursos de forma dinâmica, ou seja, sob demanda, segundo a carga de trabalho vigente. Busca-se com isso uma melhor utilização dos recursos do sistema. O controle da alocação se baseia no número de requisições de cada classe presente no sistema e no nível de diferenciação pretendido. O algoritmo *WFQ*, por sua vez, é uma adaptação do existente no nível de rede só que aplicado para o nível de aplicação. Além de oferecer diferenciação entre classes, esse algoritmo de escalonamento tem como característica a ausência da negação de serviço, como pode ocorrer nos mecanismos de Prioridade Rigoroso e Adaptativo (Teixeira *et al.*, 2005). Ambos os algoritmos são implementados em um modelo de servidor *Web* com diferenciação de serviços (Teixeira *et al.*, 2005) e avaliados através de simulação orientada a *traces*.

Ainda no nível de aplicação, mas com uma nova proposta de controle de admissão, pode-se citar o trabalho de (Cherkasova & Phaal, 2002), que propõe fazer um controle baseado em sessão, em vez de requisições individuais. Uma sessão é uma série de requisições feitas por um usuário dentro de um período pré-definido. O intuito é aproveitar a natureza transacional da maioria dos serviços *Web*, como transação bancária, comércio eletrônico, os quais mantêm estado de uso dos clientes, como operações de autenticação, consulta, pagamento, e saída do sistema. Em situações de sobrecarga, um controle de admissão baseado em sessão permite a continuidade das pré-estabelecidas, redirecionado novas sessões para outros *sites*, ou negando acesso ao serviço. Experimentos mostraram um significativo aumento de sessões completadas se comparado a um servidor com controle de admissão por requisição individual, o que garantiu melhor qualidade de serviço.

Nessa mesma linha está o trabalho de Barbato *et al.* (2006), no qual é proposto um algoritmo de escalonamento para servidores *Web* baseado em sessão: *SBSA*. A idéia básica do algoritmo é priorizar as requisições pertencentes a sessões mais longas, assim como o faz Cherkasova & Phaal (2002), só que durante a fase de admissão. A priorização é feita por meio de um parâmetro denominado *look-ahead*, atribuído a cada sessão existente no sistema. Esse parâmetro indica o número de posições em fila que a busca por requisições

de uma dada sessão se limitará. Ele aumenta à medida que o número de requisições de uma sessão cresce, priorizando assim as sessões mais longas. A validação do algoritmo é feita por meio de simulação orientada a traces, e assim como no trabalho de Cherkasova & Phaal (2002) os resultados mostraram aumento do número de sessões concluídas, e conseqüentemente melhora de desempenho.

Cherkasova (1998) propõe uma estratégia de escalonamento não preemptiva que permite o ajuste entre um atendimento imparcial ou melhor tempo de resposta das requisições. Para isto, toma-se por base um coeficiente ajustável α que controla o balanço entre a política *FIFO* ou *SJF*. Quanto mais próximo de zero for seu valor, mais justo será o atendimento oferecido. Quanto maior o valor de alpha, mais otimizado será o tempo de resposta das requisições, pois proverá uma melhor utilização dos recursos compartilhados, como *CPU* e interface de rede. A fim de evitar situação de *starvation*, típica no *SJF* original, um parâmetro que leva em conta tempo de fila foi acrescentado à fórmula de atribuição de prioridade, para que todas as requisições sejam eventualmente atendidas. No trabalho também é sugerida que a estratégia possa ser aplicada ao escalonamento de classes com diferentes prioridades. Mas, talvez pelo fato de seu enfoque principal ser a otimização da latência de sistema, a diferenciação de classes é tratada de forma superficial e simples, com o uso de fila de prioridades. Essas podem levar a situações de *starvation*, já que filas de menor prioridade podem ser indefinidamente preteridas se sempre chegar requisições de alta prioridade.

Harchol-Balter *et al.* (2000) propõe um escalonamento baseado na política *SRPT* como forma de melhorar o desempenho do atendimento de requisições estáticas *HTTP* em servidores *Web*. Essa política prioriza requisições menores, ou as que têm o menor tempo restante de processamento, objetivando com isso minimizar os atrasos em fila das demais requisições, pois o impacto dos atrasos das menores é bem menor do que o causado por grandes requisições. Apesar de o foco ser servidores *Web*, a política é implementa no nível de rede, no qual controla o estabelecimento de conexão das requisições de serviço destinadas ao servidor. Esse controle se dá priorizando os *sockets* correspondentes a requisições de arquivos menores ou que tenham o menor tempo restante de processamento. Os resultados mostraram que o aumento de desempenho se dá na ordem de 200% sob baixa carga de trabalho e vai até 500% quando sob altas cargas, além de dificilmente penalizar requisições maiores. Os experimentos foram feitos com implementação no nível do *kernel* e a carga de trabalho obtida a partir de *traces*. Apesar do enfoque principal desse trabalho ter sido desempenho, não abordando diferenciação de classes, os resultados mostram a viabilidade da utilização de uma política que prioriza a requisições menores como forma de diminuir atrasos em fila, e assim exercer certo controle sobre a qualidade de serviço oferecido.

Outro trabalho que merece destaque é o que apresenta um estudo da viabilidade da utilização de disciplinas de escalonamento de Tempo Real associado com o conceito de *QoS*.

Nesse trabalho, Ye *et al.* (2005) utiliza três disciplinas de escalonamento aplicadas em sistemas de manufatura para prover *QoS*, por meio de diferenciação de serviços, também em servidores *Web*. A primeira política, denominada *WSPT* (*Weighted Shortest Processing Time*), é similar a *SJF*, só que considera prioridades em seu escalonamento, tendo por objetivo minimizar o tempo ponderado de resposta para um conjunto de requisições. A segunda política, denominada *ATC* (*Apparent Tardiness Cost*), combina *WSPT* e a *MS* (*Minimum Slack*), que prioriza as requisições com menor "folga" de processamento (i.e. tempo de serviço restante antes de descumprir o contratado). A terceira disciplina, denominada *EDD* (*Earliest Due Date*), trabalha no sentido de minimizar o atraso máximo aceitável para um conjunto de requisições, dado os deadlines e tempo de processamento das requisições. Por meio de simulação obteve-se como resultado que essas políticas, propostas a princípio para ambientes de manufatura, podem ser utilizadas como forma de oferecer *QoS* em servidores Web, e alcançam níveis de desempenho bem superiores ao de serviço de melhor esforço empregado nos servidores convencionais. Dessas três políticas a *WSPT* e a *ATC* mostraram melhores resultados de desempenho que a *EDD*, com desempenho bastante similar entre as duas primeiras.

Conceitos de teoria de controles também têm sido aplicados juntamente com os de *QoS*, com o objetivo de oferecer ajustes dinâmicos da qualidade de serviço. O trabalho de Sha *et al.* (2002), por exemplo, combina a predição oferecida pela teoria de redes de fila com a robustez proporcionada pela auto-regulagem (*feedback control*) como meios para se garantir atrasos em termos absolutos em servidores *Web*. O trabalho de Lu *et al.* (2003) estende a abordagem seguida por Sha *et al.* (2002) e passa a oferecer garantias de *QoS* em termos relativos para tratar diferenciação de serviço quando o servidor está saturado. Ambos os trabalhos são validados por meio de implementação do modelo em um servidor *Web* real.

3.3 Considerações Finais

Todos esses trabalhos apresentam contribuições importantes para aumentar o nível de confiabilidade[2] dos serviços oferecidos na *Web*, e conseqüentemente na *Internet*. Como se pode perceber, o enfoque maior das pesquisas tem sido a especificação de *QoS* em termos relativos, expondo um cenário de menor desenvolvimento de trabalhos com provisão de *Qos* em termos absolutos, oferecendo assim um amplo campo de pesquisa a ser explorado. Especificações em termo absolutos oferecem garantias mais concretas de cumprimento de requisitos de serviço, cada vez mais exigidos por aplicações de crescente importância, como tele-medicina e comércio eletrônico. Outro ponto que pode ser observado nesses trabalhos é a viabilidade da utilização de políticas de priorização baseadas no menor serviço para a redução de atrasos em fila, constituindo assim um bom mecanismo para controle do

[2]Quanto ao cumprimento de requisitos operacionais.

nível de qualidade de serviço oferecido. A utilização de conceitos de auto-regulagem e de tempo real também podem ser devidamente explorados para o desenvolvimento de uma nova disciplina de escalonamento que trabalhe com parâmetros de serviço quantitativos (absolutos), como realizado pelos trabalhos de Sha *et al.* (2002); Ye *et al.* (2005). E como mostrou o trabalho de Kanodia & Knightly (2003), a utilização de latência de resposta (ou sistema) é uma boa opção de parâmetro de serviço, dada sua ampla representatividade dos atrasos em fila a serem devidamente controlados pela disciplina de escalonamento.

Capítulo

4

Política EBS – Exigency-Based Scheduling

4.1 Considerações Iniciais

Neste capítulo são apresentadas as abordagens e procedimentos utilizados para o desenvolvimento do projeto, a fundamentação analítica definida, para melhor estruturação e avaliação da política desenvolvida, e a descrição da política de escalonamento em si. Para isso, apresenta-se na seção 4.2 a metodologia adotada para a realização do trabalho, em seguida, na seção 4.3 discutem-se os conceitos e métricas criados a partir da modelagem analítica realizada, e por fim na seção 4.4 trata-se das várias fases de desenvolvimento da política de escalonamento com suporte a *QoS* proposta.

4.2 Metodologia

Para avaliar a eficiência e o desempenho da disciplina *EBS* utilizou-se como abordagem a modelagem e simulação orientada a eventos (seção 2.4). Além de permitir maior flexibilidade para o estudo, a simulação também constitui um bom caminho para obter boas estimativas de comportamento do sistema após modificações. A partir dos resultados, modelos mais completos e representativos do problema real podem ser construídos, pois a maior facilidade oferecida pela simulação para a solução dos modelos propicia melhores previsões de desempenho e mais alternativas de comparação, graças à possibilidade de variações menos custosas do modelo.

Assim, utilizando-se os conceitos de redes de filas (seção 2.4) construiu-se um modelo que representa um servidor *Web* com qualidade de serviço. Basicamente, esse modelo

descreve os principais eventos que ocorrem em um sistema real do gênero, tais como: eventos para tratar chegada de requisições, solicitação de serviço e liberação de recurso, além de oferecer suporte à *QoS*. Como ilustrado pela figura 4.1, o servidor modelado é do tipo mono-processado, com uma fila única de espera para processamento. Nele foram implementadas a política de escalonamento *EBS*, além da usual disciplina *FIFO*, com intuito de comparação, amplamente utilizada nos servidores convencionais. A política *SJF* também foi implementada, pois oferece margem de comparação quanto ao desempenho da disciplina *EBS* proposta, permitindo discutir se existe alguma vantagem de implantação de *QoS* em vez de simplesmente aplicar uma política clássica de escalonamento, possivelmente ótima em outros cenários. Outra política implementada foi a *EDF*, pois a *EBS*, assim como aquela, relaciona o *deadline* das requisições no mecanismo de escalonamento.

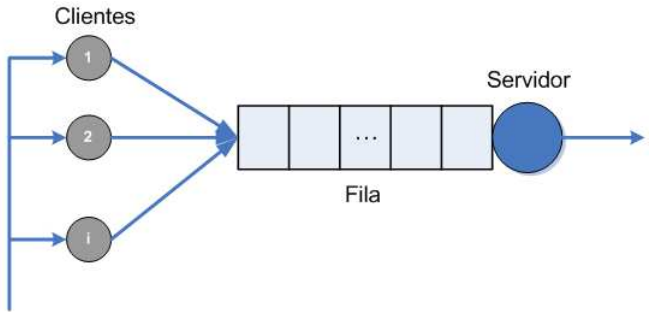


Figura 4.1: ***Modelo de servidor** Web **seqüencial com QoS.***

Para a validação do modelo e da política de escalonamento utilizou-se o pacote de simulação *SimpackJ* (Fishwick, 2005), o qual oferece um conjunto de ferramentas próprio para simulação orientada a eventos. Esse pacote é constituído por uma biblioteca de programação com uma ampla gama de métodos voltados a simulação de redes de filas. Ela foi desenvolvida a partir da biblioteca *SMPL* (MacDougall, 1989), com o acréscimo de diversas outras rotinas e a mudança do paradigma de programação para o orientação a objetos. O suporte atual é para a linguagem *Java*. A escolha dessa ferramenta de simulação se deu devido à larga experiência do Grupo de Sistemas Distribuídos e Programação Concorrente em sua utilização, o que facilitou o aprendizado e, principalmente, pelo fato do simulador ser distribuído livremente com seu código fonte completo, permitindo com isso que modificações fossem feitas quando necessário para adequar às necessidades do projeto.

A fim de incluir a nova disciplina de escalonamento, algumas modificações e extensões foram realizadas no *SimpackJ*. Definiu-se novos campos para descrever de modo mais completo as entidades que representam as requisições *Web*, incluiu-se suporte a *QoS*, e alterou-se procedimentos que implementam eventos, como os de requisição e liberação de recurso, além do acréscimo de alguns mecanismos de instrumentação.

Para controlar a qualidade do serviço oferecido escolheu-se como parâmetro a **Média de Latência de sistema**, que representa o tempo médio de residência no sistema das

requisições de um usuário (seção 4.4). Como em seu cálculo (equação 4.21) estão inclusos os tempos em fila das requisições, esse é um bom parâmetro de serviço, pois ao controlar esses tempos influencia-se diretamente o nível de qualidade oferecido.

Por fim, definiu-se como métrica de avaliação a satisfação de cumprimento contratual (seção 4.3.3) oferecida aos usuários do sistema. Ao relacionar o número de vezes que a média contratual foi cumprida, pelo número total de requisições submetidas ao sistema consegue-se analisar a confiabilidade da qualidade de serviço oferecida, e assim avaliar a eficácia da disciplina de escalonamento.

4.3 Modelagem Analítica

A diferenciação de serviços é importante para acomodar expectativas e requisitos heterogêneos de usuários e aplicações, permitindo que planos de atendimento distintos sejam fixados para serviços na *Internet* (Blake *et al.*, 1998). No nível de rede, por exemplo, agrupados em classes de serviço, fluxos de requisições/pacotes com necessidade de menores tempos de resposta e melhores taxas de entrega dos dados, como *VoIP*[1] e videoconferência, podem ter seus requisitos operacionais garantidos, enquanto usuários com requisitos simples de conectividade, como transferência de dados hipertexto, recebem serviço de melhor esforço padrão (Xiao & Ni, 1999).

No nível de aplicação a utilização de mecanismos de diferenciação permite um maior controle, por exemplo, sobre o atendimento de tráfego especulativo na *Web*, resultante de operações de *cache* antecipado (*prefetching*). Este tipo de tráfego não é primordial e pode ser descartado se necessário. A *QoS* diferenciada poderia ser resultado de políticas externas ao servidor, no caso favorecendo-se usuários pagantes em detrimento dos não pagantes (Eggert & Heidemann, 1999).

A diferenciação entre classes pode ser realizada perante alguma política de escalonamento, baseada em certos critérios para distinção de serviços oferecidos, como prover melhor qualidade aos usuários que arcam com os maiores custos (Vasiliou, 2000). Essas classes podem oferecer serviços simples, ou planos que podem ser classificados em *Gold*, *Silver*, e *Bronze* (qualidade decrescente).

Para que essa diferenciação seja oferecida é preciso implantar mecanismos de controle capazes de gerenciar recursos e manter a qualidade de atendimento. No caso de *QoS* no nível de aplicação, abordagem deste trabalho, o emprego de tais mecanismos se faz por meio de algoritmos de escalonamento, que se utilizam de parâmetros e métricas que ofereçam informações de desempenho e qualidade de atendimento para tomar decisões, comparar resultados e tentar cumprir especificações de serviço. Esta seção tem como finalidade definir os conceitos e as métricas utilizados para estruturar a política de

[1] *Voice Over Internet Protocol.*

escalonamento proposta neste trabalho, visando facilitar o entendimento e a análise dos resultados obtidos.

4.3.1 Peso da Classe de Serviço

O algoritmo de escalonamento adotado, influenciado pelas condições do ambiente, acaba impondo certa carga (peso) ao sistema para cumprir as exigências das diversas classes de serviço por ele oferecidas. Uma maneira de definir o peso que cada uma dessas classes impõe ao sistema é relacionando os seguintes parâmetros: custo médio de processamento de suas requisições e especificações de qualidade de serviço.

Denotado por C_i, esse custo especifica o tempo esperado para o processamento das requisições da i-ésima classe. Quanto maior for seu valor, maior será a carga imposta ao sistema. O parâmetro de qualidade de serviço, por sua vez, refere-se neste trabalho à média de **latência de sistema** a ser garantida para cada classe (L_{c_i}), de forma que seu impacto é inversamente proporcional ao valor especificado. Quanto menor for o requisito latência de sistema contratada, maior será o peso imposto ao mesmo.

Pode-se formalizar as relações acima discutidas estabelecendo-se o peso que a i-ésima classe impõe ao sistema por W_i (4.1).

$$W_i = \frac{C_i}{L_{c_i}} \tag{4.1}$$

As cargas W_i impostas ao sistemas formam um conjunto W (4.2) das classes por ele gerenciadas. Cada uma delas mantém certa quantidade de usuários fazendo requisições ao servidor, de forma que acrescentam pesos à carga total de processamento. Seja p_i a proporção de requisições submetidas pelos usuários da i-ésima classe ao sistema. Assim, a equação (4.3) define a carga total de um sistema (W_T) como a soma ponderada das cargas individuais de W.

$$W = \{W_1, \ldots, W_i, \ldots, W_n\} \tag{4.2}$$

$$W_T = \sum_{i=1}^{n} W_i \cdot p_i \tag{4.3}$$

A variação de p_i reflete diretamente na reconfiguração da carga do sistema, já que o impacto da exigência de uma dada classe é ponderado pela proporção de suas requisições. Assim, pelo remanejo de uma porção l de requisições da k-ésima classe para outra j descreve-se um novo cenário de carga para o sistema. Por meio deste fator de desbalanceamento de carga (l) pode-se representar diferentes cenários de simulação, como definido na equação (4.4). Estabelecendo p_{min} como a menor proporção de todas as classes

do conjunto W, ou seja, o máximo valor possível para o remanejamento, restringe-se os valores de l em: $l \in [0, p_{min}]$.

$$W'_T = \sum_{i=1}^{n-2} W_i \cdot p_i + W_j \cdot (p_j + l) + W_k \cdot (p_k - l) \tag{4.4}$$

A alteração de peso do sistema se deve ao rebalanceamento do peso relativo entre as duas classes rearranjadas. Isto pode ser observado reescrevendo a equação anterior (4.4) em função da diferença de pesos entre duas classes, como estabelece a equação 4.5. Esta diferença, ou discrepância de pesos entre duas classes, será denominada dW_{jk}. A equação 4.6 apresenta esta reformulação, que descreve a influência do peso relativo para a carga do sistema.

$$dW_{jk} = |W_j - W_k| \tag{4.5}$$

$$W'_T = W_T + l \cdot (W_j - W_k) \tag{4.6}$$

Representando W^+ como o peso da classe mais exigente e W^- como o da menos exigente, define-se dW_{MAX} como a diferença máxima entre os pesos das classes. Desta forma, os valores possíveis de dW_{jk} são limitados por: $dW_{jk} \in [-dW_{MAX}, +dW_{MAX}]$.

Com todas estas definições é possível analisar o espaço de variação da carga de um sistema. Para facilitar a análise inicial restringe-se o sistema a apenas duas classes: a mais e a menos exigente. Mantém-se ainda para cada cenário os mesmos valores de W'_T de um sistema com N classes, ou seja, remaneja-se o peso das demais classes para essas duas de forma a manter o peso do sistema. Com isso o intervalo sob o qual a carga total do sistema varia pode ser descrito pela inequação (4.7).

$$W_T - p_{min} \cdot dW_{MAX} \;\leq\; W'_T \;\leq\; W_T + p_{min} \cdot dW_{MAX} \tag{4.7}$$

Apenas rearranjando-se a inequação (4.7), os possíveis cenários de carga representados pela variação do fator de rebalanceamento l são mais convenientemente visualizados em (4.8), onde

$$-1 \;\leq\; \frac{l}{p_{min}} \cdot \frac{(W_j - W_k)}{dW_{MAX}} \;\leq\; 1 \qquad \text{, para } dW_{MAX} > 0 \tag{4.8}$$

mostra-se que por meio da variação de um fator l, que rebalanceia o peso entre duas classes, é possível descrever todos os cenários de carga do sistema. Os conceitos estabelecidos são igualmente aplicáveis ao rebalanceamento dos pesos para mais de duas classes, já que a carga imposta a um sistema de N classes pode equivaler a outro de apenas duas. Essa relação é resumida em (4.9), e discutida nos próximos parágrafos. Na equação

assume-se que $W_{i^*} \neq W^+ \neq W^-$, e que p^+ e p^- representam a proporção de requisições de uma classe mais e menos exigente, respectivamente.

$$W_T \;=\; p^+ \cdot W^+ + (\textstyle\sum_{i^*=1}^{n} W_{i^*} \cdot p_{i^*}) + p^- \cdot W^- \;=\; p \cdot W^+ + (1-p) \cdot W^- \tag{4.9}$$

Seja W_R (equação 4.10) o somatório dos pesos ponderados de todas as classes, exceto o daquelas mais e menos exigentes (i.e. peso das classes restantes). Pode-se rearranjar a carga total do sistema apenas em função do rebalanceamento de pesos das classes mais e menos exigentes, como apresentado em (4.11). Uma divisão pode ser feita definindo-se W_A como o peso ponderado da classe menos exigente e W_B o da mais exigente, após o rebalanceamento. Já p_r representa a proporção das requisições das classes restantes remanejadas para compor o peso equivalente W_A, enquanto que p_x corresponde à parcela equivalente dos menos exigentes e p_y a parcela equivalente mais exigente.

$$W_R = \sum_{i^*=1}^{n} W_{i^*} \cdot p_{i^*} \tag{4.10}$$

$$W_T = W_A + W_B \begin{cases} W_A = p^- \cdot W^- + p_r \cdot W_R \;=\; p_x \cdot W^- \\ \\ W_B = p^+ \cdot W^+ + (1-p_r) \cdot W_R \;=\; p_y \cdot W^+ \end{cases} \tag{4.11}$$

Isolando-se os parâmetros p_x e p_y na equação 4.11, e rearranjando-se os pesos apenas em função da classe mais exigente, como mostra a equação 4.12, pode-se verificar que a carga total de um sistema com N classes pode também ser representada apenas em função dos pesos de apenas duas classes.

$$\left.\begin{array}{l} p_x = p^- + p_r \cdot \frac{W_R}{W^-} \\ \\ p_y = p^+ + (1-p_r) \cdot \frac{W_R}{W^+} \end{array}\right\} \text{considere-se } p=0 \;\Rightarrow\; W_T = \sum_{i=1}^{n} W_i \cdot p_i = p^- \cdot W^- + \left(p^+ + \frac{W_R}{W^+}\right) \cdot W^+ \tag{4.12}$$

A razão $\frac{W_R}{W^+}$, que faz parte da equação (4.12), pode assumir valores maiores que 1. Isto acontece apenas quando o remanejamento da proporção das classes não é suficiente para manter a carga do sistema inalterável. Nesses casos, comum no trabalho com mais de duas classes, faz-se necessário ainda o acréscimo/remoção de requisições no sistema equivalente.

O conceito de sistema equivalente (4.13) é importante pois existem muitas permutações das diferentes proporções que as diversas classes podem assumir, evitando a execução de experimentos com desbalanceamentos repetitivos (permutações com cargas iguais), o que facilita o estudo.

$$W = p \cdot (W^+ - W^-) + W^- \tag{4.13}$$

Estando bem definida a exigência imposta ao sistema, em relação à proporção das classes mais e menos exigentes (4.13), é importante estabelecer outra métrica de análise do sistema, ou seja, a taxa de utilização.

4.3.2 Taxa de utilização

Da estatística, para grandes quantidades amostrais de uma variável aleatória discreta, seu valor esperado indica a média ponderada de suas ocorrências (Meyer, 2003). Por meio dele é possível ter uma impressão geral do comportamento da variável, independentemente dos detalhes de sua função de distribuição de probabilidade. O valor esperado de uma variável aleatória X discreta é simbolizado por (4.14), com $p(x_i)$ representando a probabilidade de cada uma de suas ocorrências: $P(X = x_i)$.

$$E(X) = \sum x_i \cdot p(x_i) \tag{4.14}$$

Desta forma, por meio do valor esperado dos tempos de serviço (C) e intervalos de chegada (A) observados para as classes que compõem o sistema, é possível descrever a taxa de utilização (U) do sistema por meio da equação (4.15). Nela observa-se que quanto maior for o intervalo entre as chegadas de requisições mais ocioso o sistema ficará. Em contrapartida, a utilização do sistema cresce proporcionalmente ao aumento do tempo de processamento esperado para as requisições que chegam.

$$U = \frac{C}{A} \tag{4.15}$$

Uma vez que o custo de processamento C_i dos usuários (ou classes) está relacionado à exigência imposta ao sistema pela equação (4.1), a taxa de utilização também pode ser definida de acordo com a relação (4.16).

$$C = W_T \cdot \overline{L} \;\; \Rightarrow \;\; U = \frac{W_T \cdot \overline{L}}{A} \tag{4.16}$$

Como tempo esperado de serviço (C) é diretamente proporcional à taxa de utilização (U), verifica-se que a utilização do sistema também é afetada pela ação ponderada da exigência (W_T) vezes a latência média ($\overline{L}$) por ele oferecida. Tal relação oferece ao escalonador um meio de monitoramento e controle da taxa de utilização pela alteração de valores de latência garantidas às classes. Este controle da taxa de utilização pelo contrabalanceamento da latência ou exigência possibilita ainda, mesmo que de forma limitada,

a diminuição da necessidade de *dropping*[2] de requisições (pacotes) por parte do controle de admissão.

4.3.3 Satisfação do cliente

Considera-se que um usuário estará satisfeito com o serviço a ele oferecido quando obtiver uma alta porcentagem de requisições atendidas em média abaixo do limiar de qualidade contratado. Logo, analiticamente, o escalonador pode tomar como índice de satisfação do usuário a equação (4.17), em que são relacionadas a quantidade total de requisições submetidas pelo i-ésimo usuário (R_i) e o número de vezes em que a média de latência de sistema dessas requisições ficou abaixo do limiar contratado (N_i).

Uma requisição é bem atendida quando sua latência de sistema observada (real) for menor ou igual a latência somada a um desvio padrão aceitável contratados. Quanto mais próximo de R_i for o valor de N_i, maior será a satisfação proporcionada ao usuário i. Esta é uma métrica suficiente para avaliar o desempenho de um escalonador no cumprimento dos níveis de QoS por ele oferecidos.

$$S_i = \frac{N_i}{R_i} \tag{4.17}$$

Como R_i nada mais é que a proporção de requisições submetidas pelo i-ésimo usuário em relação ao total de requisições do sistema (R), pode-se também reescrever a satisfação em termos de R (4.18), e utilizar isto para representar a satisfação em função da exigência. Para tanto se toma como base o resultado definido pela equação de pesos equivalentes (4.13). A partir dela isola-se o parâmetro p (um valor variando de 0 a 1) que representa a proporção (porcentagem) de requisições de alta exigência. Com ele pode-se definir as satisfações S^+ e S^-, proporcionadas aos usuários mais e menos exigentes, respectivamente, como apresentado na equação (4.19).

$$S_i = \frac{N_i}{p_i \cdot R} \tag{4.18}$$

$$p = \frac{W_T - W^-}{W^+ - W^-} \quad \Rightarrow \quad \begin{cases} S^+ = \frac{N_+}{R} \cdot \frac{(W^+ - W^-)}{(W_T - W^-)} \\ S^- = \frac{N_-}{R} \cdot \frac{(W^+ - W^-)}{(W^+ - W_T)} \end{cases} \tag{4.19}$$

A equação (4.18) retrata a relação entre a satisfação de um dado usuário e a proporção de suas requisições no sistema. Quanto maior o número de requisições de um usuário presentes no sistema, pior deverá ser sua satisfação, uma vez que um sistema sobrecarregado tende a enfileirar mais requisições pendentes, afetando inclusive no número das requisições bem atendidas. Outra relação importante pode é apresentada na equação (4.19), onde se

[2]Ação para controle de requisições no sistema feita por meio da eliminação (negação atendimento) quando um número máximo suportado for atingido.

observa a dependência do peso imposto por uma dada classe sobre a satisfação oferecida à outra. Quanto menor for a carga imposta por uma classe menos exigente ao sistema (W^-), maior será a satisfação da mais exigente(S^+).

Caso um módulo para controle de admissão venha fazer parte desse sistema, na equação que rege a satisfação de um dado usuário pode ser conveniente incluir ainda a taxa de descarte de requisições, como por exemplo em (4.20). Esta taxa (d_i) representa a porcentagem de requisições do i-ésimo usuário que foram descartadas pelo controle de admissão, tendo seu valor variando entre 0 e 1. Fixar $d_i = 0$ significa ausência desse tipo de controle. Conforme a taxa de descarte aumenta, a satisfação do cliente diminui, como é o esperado.

$$S_i = \frac{N_i}{p_i \cdot R} \cdot (1 - d_i) \tag{4.20}$$

Com estes conceitos e métricas bem definidos pode-se estruturar melhor a política de escalonamento proposta, além de possibilitar ao algoritmo algumas ferramentas de controle para monitorar e tomar decisões sobre a qualidade de serviço oferecida. A formalização apresentada nesta seção auxiliará ainda em uma melhor análise dos resultados, permitindo validar os dados coletados pela simulação do modelo implementado.

4.4 Política Desenvolvida

Um modo de especificar a qualidade de serviço oferecida por um sistema é quanto ao nível de desempenho oferecido a uma classe de usuários do sistema. Uma métrica geralmente utilizada para quantificar tal desempenho é o tempo de resposta experimentado pelo usuário final, que é definido como o intervalo entre a submissão e o completo recebimento do resultado da requisição (Jain, 1991).

Sendo diretamente perceptível no nível de aplicação, o tempo de resposta permite ampla representatividade de desempenho, pois inclui valores como latência de rede e de sistema (servidor). Por latência de rede entendam-se atrasos causados pela situação atual da rede e os introduzidos pelos seus protocolos, como *TCP*, por exemplo. A latência de sistema engloba tempos de processamento das requisições e atrasos em fila. Minimizar qualquer destes atrasos significa melhora no desempenho de aplicações *Web*, e uma forma de oferecer melhores níveis de serviço.

Trabalhar no sentido de diminuir a influência dos atrasos em fila no tempo de resposta é algo mais suscetível a controle, se comparado ao estado atual da rede, e que impõe uma menor complexidade do que alterar, ou mesmo propor um novo protocolo de rede. Assim, a estratégia inicialmente adotada para o desenvolvimento da política de escalonamento foi a de manipular os tempos de fila das requisições de acordo com o parâmetro contratual de cada classe. As requisições que estão mais próximas ou foram descumpridas receberão

maior prioridade do escalonador, ao contrário daquelas que toleram maiores tempos de fila.

Para definir uma classe escolheu-se como parâmetro o tempo médio de latência de sistema a ser garantido às requisições de um dado usuário, pois nele está incluso o tempo de fila gerenciado pelo escalonador, o que permite certo controle sobre a qualidade do atendimento. Este parâmetro, neste texto denominado latência contratada pela i-ésima classe (L_{c_i}), é especificado previamente por um acordo entre o provedor de serviços e o cliente, e é utilizado pelo escalonador como base para atribuição de prioridades. O valor que será comparado com L_{c_i} é o da Latência Real (L_u), que é a latência média de sistema atualmente oferecida para o usuário u.

Sempre que o servidor terminar de atender uma dada requisição j do usuário u o valor de L_u será recalculado. Como mostra a equação 4.21, este cálculo é a média entre a antiga latência real de u (L'_u) e o tempo de residência da requisição j recém atendida. O valor de $time()$ representa o tempo atual, $timeStamp_j$ o tempo de chegada da requisição j e R_u o número de requisições anteriormente submetidas por u.

$$L_u = \frac{(L'_u \cdot R_u) + (time() - timeStamp_j)}{R_u + 1} \tag{4.21}$$

Como o algoritmo não é preemptivo, as novas requisições que chegam ao sistema e não encontram servidor disponível são transferidas para uma fila única de espera. Ao término da execução de uma requisição, o escalonador recalcula a latência real daquele usuário, e em seguida busca na fila a requisição mais urgente, ou seja, aquela que tolera o menor tempo de espera dentre todas as existentes no sistema. Essa então terá acesso ao servidor, e o ciclo se repete.

O tempo de espera máximo, também denominado *deadline*, representa o quanto uma requisição ainda pode esperar na fila antes de começar a descumprir seu contrato. Para obtê-lo, isola-se na inequação 4.22 a variável D_j, que representa o *deadline* da requisição j. Como o interesse é obter o máximo tempo de espera aceitável pelo usuário u antes que o valor de sua latência real (L_u) ultrapasse o de sua latência contratada, igualam-se os termos. Os valores R_u e T_{w_j} expressam respectivamente o número de requisições feitas pelo usuário u e o tempo de espera em fila da requisição j, até o momento.

$$\frac{(L_u \cdot R_u) + T_{w_j} + D_j}{R_u + 1} \leq L_{c_i} \tag{4.22}$$

Quanto menor o valor de D_j, maior a prioridade atribuída à requisição, pois tem maior urgência. Basicamente este é o mesmo princípio utilizado pela política de escalonamento *EDF* (seção 2.3.3), de sistemas de tempo real, porém tratada de forma mais flexível, pois visa manter valores médios de latência de sistema, e não preemptiva.

Observe que uma requisição pertencente a uma classe mais exigente (com menor valor de latência contratada) nem sempre terá prioridade sobre requisições de classes menos

exigentes. Isto ocorre devido ao comportamento dinâmico do algoritmo na priorização das requisições. Se um usuário apresentar valores reais de latência melhores que a contratada, as requisições de usuários menos exigentes, mas prejudicados no cumprimento dos seus parâmetros de serviço, receberão maior prioridade, compensando assim por posposições cometidas ao longo do escalonamento.

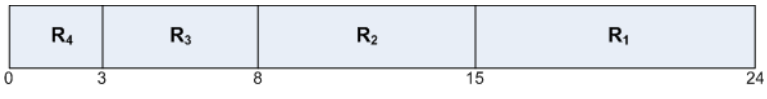
Dentre as requisições classificadas como mais urgentes podem existir algumas com os mesmos valores de *deadline*. A abordagem de escalonamento descrita anteriormente se restringe apenas a marcar como mais prioritária a primeira dessas requisições encontradas. No entanto, apesar de apresentar a mesma urgência, as requisições desse conjunto podem impor pesos distintos ao sistema, uma vez que, além do parâmetro de serviço (requisição operacional), o custo de processamento é outro fator impactante sobre a exigência imposta ao sistema, como rege a equação (4.1). Minimizar esse impacto é importante pois um sistema sob menor carga terá melhores condições para lidar com os requisitos de serviço de suas requisições.

Para ilustrar melhor esse caso, tem-se o seguinte exemplo com quatro requisições de mesma *deadline* em uma fila:

- a primeira requisição (R_1) demanda $9ut$[3] para ser servida;
- a segunda requisição (R_2) demanda $7ut$ para ser servida;
- a terceira requisição (R_3) demanda $5ut$ para ser servida;
- a quarta requisição (R_4) demanda $3ut$ para ser servida;

Se essas requisições fossem processadas na ordem de chegada (*FIFO*), como ocorre na primeira abordagem desenvolvida, os tempos de espera em fila das requisições seriam dados da seguinte forma: o da primeira seria $0ut$, $9ut$ para a segunda requisição, $16ut$ para a terceira requisição, e $21ut$ para a quarta. Com isso, a média do tempo de resposta proporcionado por este escalonamento seria $(0 + 9 + 16 + 21)/4 = 11.5ut$.

Uma forma de proporcionar tempos médios de espera (foco da qualidade de serviço a ser oferecida) menores do que aqueles obtidos com o *FIFO* é com o uso da política de escalonamento *SJF*. Nela as requisições em espera por atendimento são organizadas em uma fila segundo o tempo de processamento (T_p), sendo colocados à frente os menores *jobs*, isto é, os que serão processados em intervalos de tempo menores. Aplicando-se o conceito da política *SJF* se teria o seguinte escalonamento:



Desta forma o tempo de espera da primeira requisição seria $15ut$, o da segunda seria $3ut$, o da terceira seria $8ut$, e o da quarta requisição seria $0ut$, com um tempo médio de espera $(15+3+8+0)/4 = 6.5ut$. O *SJF* provê um tempo de resposta ótimo[4] pelo fato de

[3]Unidades de tempo.

[4]Considerado ótimo quando o conjunto de requisições a ser escalonado é conhecido previamente.

oferecer mínimos tempos de espera ao conjunto de requisições escalonadas. Realocar as requisições mais curtas na frente das que exigem maior tempo de processamento permite que o tempo de espera das mais curtas diminua mais do que aumenta o das requisições mais longas, possibilitando uma diminuição na média de tempo de resposta oferecida pelo sistema.

Assim, dentre as mais urgentes faz sentido também reordenar as requisições pelos seus valores esperados de processamento. Isso possibilita melhor utilização de recursos de sistema, como menores tamanhos de fila[5], e o que é mais importante, menores médias de latência de sistema, contribuindo para obtenção de melhores níveis de qualidade de serviço.

Com esse intuito, uma segunda abordagem de escalonamento foi desenvolvida, a qual atribui prioridades baseada na *deadline* e no valor esperado do tempo de processamento da requisição. Como ilustra a equação (4.23), a prioridade de uma dada requisição j em fila, do usuário u, é dada por P_j. Quanto menor for o valor de P_j maior será a prioridade de escalonamento da requisição j, conseqüentemente as requisições que apresentarem maior urgência (menores valores de D_j) e menor custo esperado de processamento (T_{p_j}) serão classificadas como mais prioritárias.

$$P_j = D_j \cdot T_{p_j} = \left(\Big(L_{c_u} \cdot (R_u + 1) \Big) - \Big(L_u \cdot R_u \Big) - T_{w_j} \right) \cdot T_{p_j} \tag{4.23}$$

Diferentemente da política *SJF*, esta não apresenta problema de *starvation*, pelo fato de utilizar também o *deadline* como critério de priorização. Assim, mesmo requisições mais longas não serão indefinidamente preteridas com a chegada de outras mais curtas, pois quanto mais tempo elas ficarem no sistema, menor se tornarão seus *deadlines*, o que permite serem eventualmente atendidas. O aumento do tempo em fila (T_{w_j}) contribui para o aumento da priorização das requisições.

Um artifício geralmente empregado em fórmulas de controle de escalonamento é a potencialização (ou não) de certos fatores que a compõe, elevando-os a um coeficiente *alpha* (α), como ocorre em (Cherkasova, 1998). Isto possibilita um melhor espalhamento entre os valores de prioridade, já que valores antes muito próximos uns dos outros ficam mais bem diferenciados e devidamente escalonados. Na tentativa de oferecer tal nível de otimização à segunda abordagem de escalonamento, desenvolveu-se uma terceira versão para disciplina, incluindo tal coeficiente no parâmetro T_{p_j} (4.24).

$$P_j = D_j \cdot T_{p_j}^{\alpha} \tag{4.24}$$

Uma análise do efeito do coeficiente α sobre a atribuição de prioridades é importante para a escolha de valores que otimizem o escalonamento. Aqueles entre 0 e 1 diminuem

[5]Requisições menores tendem a sair mais rapidamente do sistema, diminuindo o tempo de espera das demais, afetando assim a quantidade de requisições aguardando por atendimento.

a influência do tempo de processamento esperado no cálculo da prioridade de uma requisição. Se α for igual a 0 o escalonamento terá o mesmo comportamento do proposto pela primeira versão, enquanto que valores de α iguais a 1 fazem com que o escalonador defina as prioridades de acordo com o explicado para a segunda versão de política desenvolvida. Valores de *alpha* maiores que 1 potencializam o espalhamento dos valores de *deadline* proporcionado pelo fator T_{p_j}.

Por meio de simulações, detalhadas nas seções a seguir, observou-se que em cenários com sobrecarga de requisições de alta prioridade, valores de *alpha* mais próximos de 0 proporcionam melhores taxas de satisfação para contratos de serviço. Em contrapartida, cenários com predomínio de requisições menos exigentes apresentam melhores porcentagens de satisfação para valores de *alpha* maiores que 1. Para entender melhor a razão de tal comportamento é necessária uma análise mais detalhada do objetivo proposto e o real funcionamento da segunda versão da disciplina de escalonamento.

Inicialmente o objetivo era de priorizar requisições mais urgentes e que demandam menores custos de processamento, em razão dos critérios de serviço e desempenho discutidos anteriormente. No entanto, tal resultado esperado não ocorre para todos os casos. Naqueles onde existem valores negativos[6] de *deadline*, como nos cenários onde o sistema está saturado por requisições mais exigentes, o resultado do escalonamento é completamente o oposto quando aplicado às requisições que estão em descumprimento de contrato. Nesses casos, simplesmente multiplicar o *deadline* pelo tempo de processamento causa uma inversão de prioridades em relação ao propósito da política *SJF*.

Por exemplo, se existirem duas requisições mais urgentes, e com *deadline* negativo:

- uma requisição R_1 com $D_1 = -1$, e $T_{p_1} = 7.5ut$;
- uma requisição R_2 com $D_2 = -2$, e $T_{p_2} = 2.5ut$;

Se o escalonamento proposto pela segunda abordagem for aplicado nesse caso, haveria uma inversão de prioridades, pois a $P_1 = -7.5ut$ e a $P_2 = -5.0ut$. Mesmo R_2 sendo mais urgente, mas pelo fato de $P_1 < P_2$, a requisição R_1 receberia maior prioridade e seria inicialmente escalonada. Voltando à análise do coeficiente *alpha*, pode-se concluir que nos cenários com predomínio de requisições mais exigentes a escolha de valores de *alpha* tendendo a zero minimizam a deficiência de escalonamento apresentada pela segunda abordagem, pois o valor de T_{p_j} tenderia a 1.

De modo a garantir que o objetivo almejado com a segunda versão do escalonador seja aplicável a todos os casos é proposta uma última abordagem para a disciplina. Como definido na equação (4.25), para as requisições mais urgentes que ainda não tiveram seus *deadlines* descumpridos, a prioridade é diretamente proporcional ao seu *deadline* e ao custo esperado de processamento. Para aquelas com descumprimento de *deadline* a prioridade é inversamente proporcional a seu custo esperado de processamento, já que quanto menor

[6]Um valor negativo de *deadline* significa que o contrato foi violado, i.e. o tempo que uma requisição pode aguardar na fila é menor que zero.

for esse custo menor será o valor de P_j resultante, e assim maior será sua prioridade de escalonamento. Dessa forma garante-se que as requisições mais urgentes, independente de terem descumprido ou não seus *deadlines*, e com menores custos esperados de processamento sejam escalonadas primeiro.

$$P_j = \begin{cases} D_j \cdot T_{p_j} & \text{se } D_j \geq 0 \\ D_j \cdot \frac{1}{T_{p_j}} & \text{se } D_j < 0 \end{cases} \tag{4.25}$$

A abordagem de escalonamento proposta baseia-se na exigência imposta ao sistema como critério de controle da qualidade de serviço a ser oferecida. Requisições mais exigentes tendem a ter maior urgência de atendimento, devido a especificações contratuais mais estritas, como estabelecido pela equação (4.1). Com isso, geralmente elas recebem maior prioridade de escalonamento, para manter o nível de qualidade de serviço exigido, pois requisições menos exigentes tendem a ser mais tolerantes com o tempo de espera em fila, não demandando esforço imediato.

No entanto, dentre as mais urgentes não é interessante atender inicialmente as que impõem maior peso ao sistema, pois o impacto dessas geraria uma degradação significativa sobre as médias de latência das demais. Postergar o escalonamento dessas requisições visa aliviar o peso imposto ao sistema e às médias de latência de maneira geral, de forma que o servidor realmente tenha condições de garantir os requisitos contratuais das mais exigentes. A meta continua sendo atender primeiro as requisições que demandam ações imediatas para que se possa garantir a qualidade de serviço contratado, só que isso é feito de forma a otimizar a qualidade como um todo, inclusive para as demais mais urgentes. Assim, para expressar tal semântica e objetivo da política proposta, o algoritmo de escalonamento é denominado *EBS*: *Exigency-Based Scheduling*.

Estas sucessivas melhorias possibilitaram obter bons resultados de confiabilidade na qualidade de serviço oferecida, além de melhor desempenho e significativa diferenciação de serviço. O reflexo dessas melhorias foi um considerável aumento da satisfação contratual dos usuários do sistema como um todo. Tais constatações podem ser observadas a partir da análise dos resultados obtidos com a simulação do modelo proposto, apresentados nas seções que seguem.

4.5 Considerações Finais

Neste capítulo foi discutida a metodologia empregada para o desenvolvimento do projeto, a modelagem analítica realizada para melhor estruturação e análise dos resultados, bem como a política proposta neste trabalho.

Na seção que descreve a metodologia (4.2) apresentou-se a abordagem adotada para estudar a eficácia da política, que foi a utilização de técnicas de modelagem e simulação de sistemas; a composição geral do modelo desenvolvido, como os principais eventos e

políticas implementadas; a ferramenta de simulação empregada para validação do modelo e política propostos, discutindo o motivo de sua escolha e as adequações necessárias que foram realizadas; e por fim o parâmetro de serviço utilizado para controle da qualidade de serviço, bem como a métrica de avaliação proposta para estudar a eficácia da disciplina desenvolvida.

Em seguida, na seção 4.3, apresentou-se uma modelagem analítica desenvolvida para estruturar melhor a política e auxiliar na análise dos resultados. Definiu-se que a exigência de cada classe de serviço está relacionada com seu custo de processamento e com os requisitos de serviço exigidos, os quais acabam impondo certo peso sobre o sistema. Relacionou-se também a taxa de utilização com a exigência das classes, e a taxa de satisfação com a qualidade de serviço oferecida aos usuários do sistema.

Por fim, na seção 4.4, descreveu-se a política de escalonamento proposta neste trabalho: a *EBS* (*Exigency Based Scheduling*). Foram discutidas as várias etapas de sua evolução, com as necessidades observadas e as melhorias realizadas para alcançar melhores níveis de qualidade de serviço.

Com a metodologia bem definida pode-se entender melhor a forma e as decisões tomadas para o desenvolvimento do projeto. Os conceitos e métricas obtidos a partir da modelagem analítica possibilitam maior estudo e entendimento do funcionamento e objetivo da política proposta. Com esse embasamento pode-se fazer uma melhor análise dos resultados apresentados na próxima seção.

Capítulo

5

Experimentos, Resultados e Análise

5.1 Considerações Iniciais

Nesta seção são apresentados e discutidos os resultados obtidos pela simulação do modelo especificado no Capítulo 4. A fim de fundamentar e tornar o estudo da política proposta mais abrangente, serão utilizados os conceitos e métricas definidos na Seção 4.3. A abordagem analítica apresentada possibilita uma melhor compreensão e generalização dos resultados.

Para um melhor direcionamento deste estudo, além da escolha de métricas e técnicas de análise é importante salientar a relevância dos seguintes pontos:

- verificação de quais fatores do sistema influenciam diretamente no comportamento do escalonador, e de que forma;
- análise das situações em que se consegue ganhos consideráveis de desempenho com o emprego da política proposta;
- determinar se o emprego de uma política com suporte a *QoS* oferece alguma vantagem sobre a utilização de uma política clássica de escalonamento, possivelmente ótima em outros cenários (como *SJF* e *EDF*);
- inferir as taxas de satisfação dos usuários das diversas classes perante o uso dessa e outras políticas de escalonamento.

Para abranger todas essas questões organizou-se os dados coletados em três tipos de gráficos.

O primeiro tipo tem por objetivo ilustrar o comportamento das políticas testadas, mostrando para cada cenário qual a média real de latência de sistema oferecida para

um dado usuário ao longo do tempo. Além de possibilitar a análise da eficácia de cada política em cumprir os contratos estabelecidos, este tipo de gráfico apresenta também um comparativo do desempenho entre elas, pois valores de latência oferecidos para um mesmo usuário, sob as mesmas condições do ambiente, são plotados para os diferentes escalonadores empregados.

O segundo tipo de gráfico representa a variação da satisfação média dos usuários do sistema perante a aplicação das diferentes disciplinas de escalonamento, sob a variação de diversos fatores do ambiente, como proporção de requisições entre as classes e valores de exigência contratual.

Por fim, o terceiro tipo de gráfico tem por objetivo mostrar a qualidade dos valores individuais de latência oferecidos, em média, para todos os usuários do sistema, apresentando a porcentagem média de latências de sistema que ficaram abaixo da contratada para cada usuário.

Para uma melhor organização da discussão os resultados estão apresentados em duas seções: comportamento do escalonador (seção 5.3) e satisfação e variação do atendimento (seção 5.4).

5.2 Planejamento de Experimentos

Com o modelo de simulação e a métrica de avaliação bem definidos (seções 4.2 e 4.3.3), seguiu-se para o planejamento e execução dos experimentos. Para isso, foram especificadas duas classes de serviço, uma classe A mais exigente e outra B menos exigente. É válido lembrar que exigência está relacionada com a média de latência de sistema especificada para cada classe e, no caso, quanto menor for essa média, maior será a exigência da classe.

Para verificar a capacidade do escalonador de gerenciar um sistema com diversos usuários, e mesmo assim manter suas respectivas especificações de serviço, estabeleceu-se que cada classe (A e B) atende 10 usuários, totalizando assim 20 usuários submetendo requisições ao sistema. Dentro de uma mesma classe, todos os usuários têm a mesma chance de fazer requisições *Web*, descrita por uma distribuição uniforme.

Sintetizou-se uma carga de trabalho com distribuição exponencial para descrever tanto os intervalos de chegada quanto o tempo de execução das requisições. Essa é uma distribuição amplamente utilizada para analisar sistemas de filas (MacDougall, 1989). Os intervalos de chegada são descritos com média $4u.t.$, enquanto que o tempo de execução média tem $3u.t.$, equivalendo com isso a um sistema com uma taxa de utilização de 75%, como rege a métrica (equação 4.15).

Para analisar sob quais fatores a política de escalonamento se apresenta ou não adequada, foram definidos alguns cenários através da variação de alguns parâmetros. Um deles é a porcentagem de variação de contrato (P) em relação ao valor de latência média oferecida durante uma simulação com a disciplina FIFO (L_{FIFO}). Ao se utilizar como

base médias de latência que seriam oferecidas por um servidor convencional sem suporte à *QoS* tem-se por objetivo definir contratos de serviço viáveis, e não contratos com tempos impossíveis de serem garantidos com qualquer tipo de algoritmo de escalonamento.

Dessa forma o cálculo do contrato A (L_{c_A}) é obtido pela equação (5.1) e o da classe B (L_{c_B}) pela equação (5.2). Durante as simulações, atribuiu-se à P os seguintes valores: $5\%, 10\%, 20\%, 30\%, 40\%$, e 50%. Assim, quanto maior for a variação dos contratos em relação ao escalonamento *FIFO*, mais e menos exigente se tornarão os usuários das classes A e B, respectivamente.

$$L_{c_A} = L_{FIFO} - (L_{FIFO} \cdot P) \tag{5.1}$$

$$L_{c_B} = L_{FIFO} + (L_{FIFO} \cdot P) \tag{5.2}$$

Outro parâmetro que define um cenário é a proporção de requisições de classe A no sistema. Por meio deste parâmetro especificou-se duas situações extremas, uma com um sistema sobrecarregado de requisições de baixa prioridade ($10\%A$ e $90\%B$) e outra com um sistema sobrecarregado com requisições de alta prioridade ($90\%A$ e $10\%B$). Definiu-se também uma situação de igualdade no número de requisições de ambas as classes ($50\%A$ e $50\%B$).

Para se ter uma ampla amostragem dos dados fixou-se como 100.000 o número de requisições submetidas em cada cenário a ser simulado, e com o intuito de alcançar confiabilidade estatística realizou-se os experimentos 30 vezes (Bhattacharyya, 1977), simulando os 18 cenários em cada uma das vezes.

5.3 Análise do comportamento do Escalonador

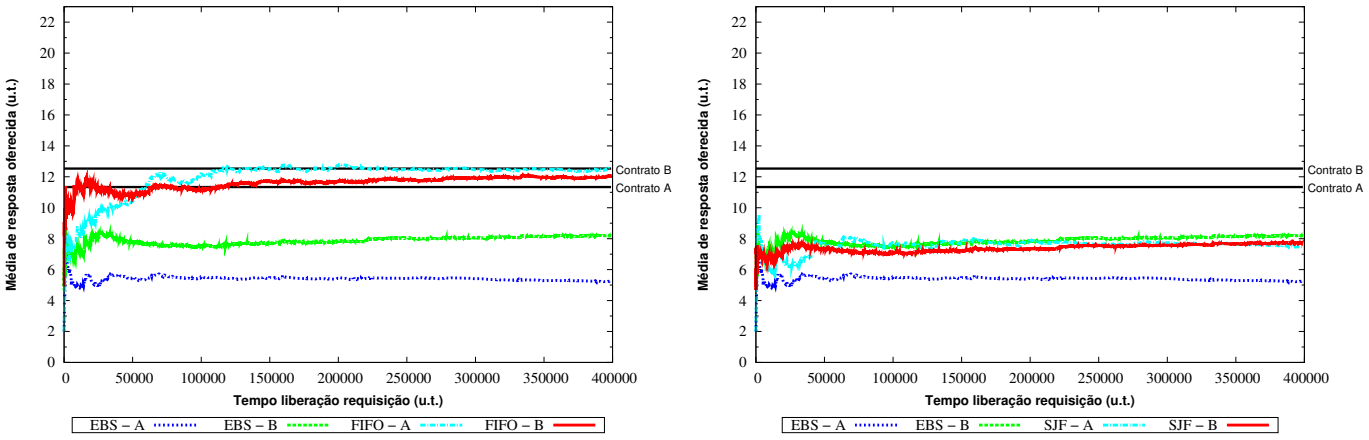
As figuras desta seção apresentam experimentos de simulação que permitem um comparativo de comportamento entre políticas: *EBS*, *FIFO*, *SJF* e *EDF*. Em seus gráficos são plotadas as médias de latência de sistema oferecidas ao longo do tempo para as classes de serviço mais e menos exigentes, classes (A) e (B), respectivamente. O eixo das abscissas informa o término de atendimento de uma requisição e o momento em que a média de latência oferecida àquele usuário é atualizada. Essas médias são representadas no eixo das ordenadas. Os contratos que definem as classes de serviço são representados por duas retas horizontais, intituladas: *Contrato A* e *Contrato B*.

Foram selecionados alguns cenários principais de variação contratual dentre todos os estudados a fim de estudar, para casos extremos, a influência deste e de outros fatores sobre a eficácia dos algoritmos em manter as médias de latência dentro do limiar de qualidade especificado para cada classe. Como discutido na seção anterior, a variação contratual é a porcentagem de discrepância dos serviços (qualidade de latência) oferecidos por cada algoritmo em relação ao tempo de residência em um sistema de escalonamento

convencional: *FIFO*. Os detalhes dos demais cenários especificados pela seção (4.2), podem ser conferidos no apêndice A ao final deste documento.

5% Variação Contratual

No primeiro cenário, apresentado pela Figura 5.1, é feito um comparativo entre o comportamento da *EBS* com as outras duas políticas sem suporte a *QoS* (*FIFO* e *SJF*). Nesse cenário há pouca proporção de requisições de classe *A*, e a discrepância entre os valores contratuais é mínima, apenas 5% em relação a um escalonamento convencional. Como se pode observar, a qualidade das médias oferecidas a uma classe mais exigente pela *EBS* é melhor se comparada com as oferecidas pelas outras duas disciplinas. Os ganhos de desempenho e a qualidade de serviço oferecida em relação ao *FIFO* são bem consideráveis. Comparadas com a *SJF*, a média oferecida a um usuário menos exigente ficou próxima das médias oferecidas pela *EBS*, enquanto que a oferecida a um usuário mais exigente teve uma considerável diminuição dos valores de latência de sistema. É válido ressaltar que a *SJF* tem desempenho ótimo quando tem a disposição todo o conjunto de requisições *a priori*. No entanto, mesmo no escalonamento dinâmico[1] o *SJF* consegue baixas médias de latência no sistema, como ilustra o gráfico 5.1(b).



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.1: ***Comparativo entre escalonadores - Cenário 10% A e 90% B com 5% de variação contratual***

Em ambos os gráficos, 5.1(a) e 5.1(b), nota-se também a capacidade de garantia dos contratos de serviço por parte da política proposta, pois as médias oferecidas para ambas as classes ficaram em níveis significativamente melhores do que os contratados. Os baixos tempos de residência de sistema, devido ao menor impacto da exigência das classes sobre as médias de latência, proporcionam menores tempos de resposta percebidos pelos clientes.

Outro fato que se pode notar pela análise da Figura 5.1 é que as médias oferecidas tanto pela disciplina *FIFO* quanto pela *SJF* não apresentam significativa diferença de

[1]Decisão sobre a ordenação das requisições feita em tempo de execução.

qualidade entre as classes, pois o parâmetro de *QoS* não está relacionado ao mecanismo de priorização implementado por elas.

Ainda nesse mesmo cenário, com baixa discrepância contratual e pequena proporção de requisições mais exigentes, são comparadas as duas políticas com suporte a *QoS*. Como mostrado na Figura 5.2, ao utilizar apenas o *deadline* como critério de controle, a política *EDF* consegue administrar as médias de latência oferecidas abaixo das especificadas em cada contrato, mas com qualidade inferior se comparada com a oferecida pelo escalonamento proposto. A *EBS*, por levar em conta a exigência imposta ao sistema, realiza um melhor escalonamento ao considerar tanto a urgência de atendimento das requisições quanto o impacto dos seus pesos sobre a qualidade do atendimento como um todo. No caso da *EDF*, pela baixa discrepância entre os pesos das classes, e grande proporção de requisições classe B no sistema, as médias de latência delas tendem a serem maiores que as oferecidas a um usuário classe A.

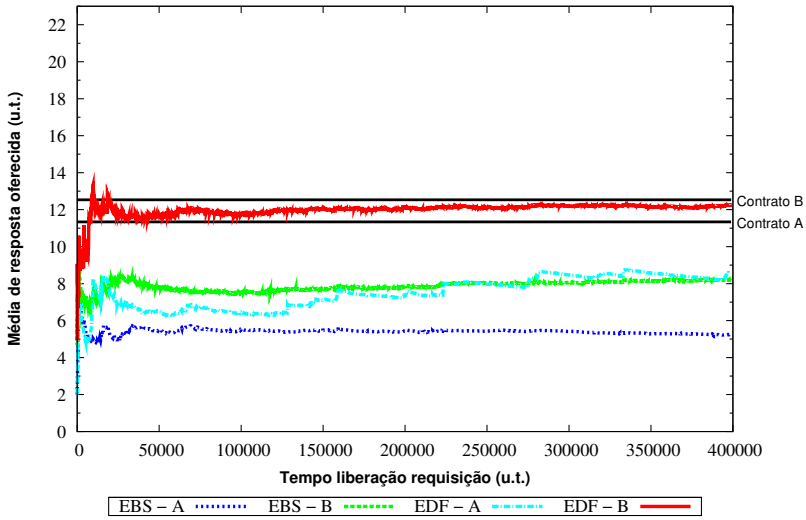
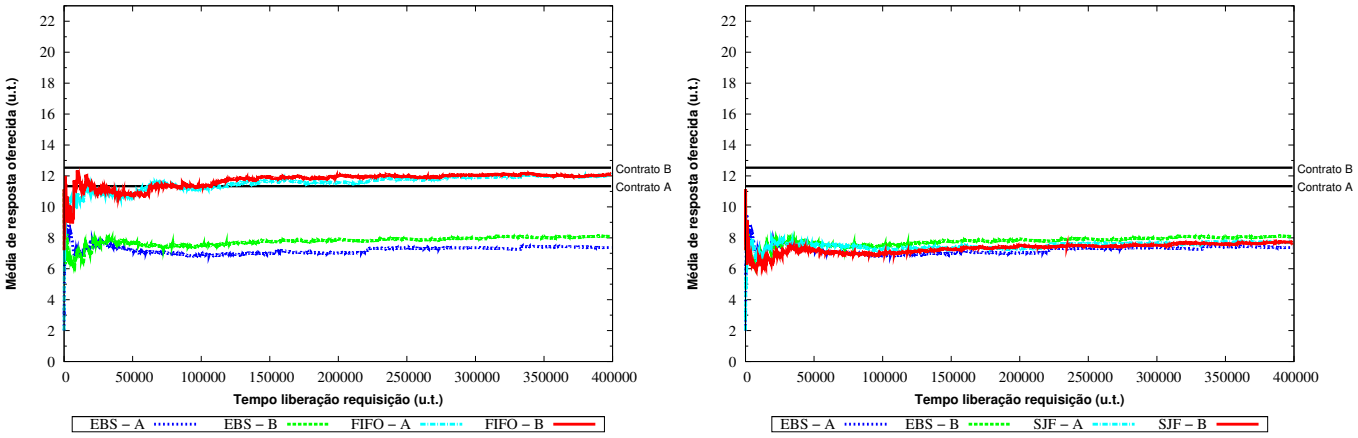


Figura 5.2: ***Comparativo entre EBS e EDF - Cenário 10% A e 90% B com 5% de variação contratual***

Mantendo-se a mesma variação contratual e incrementando o número de requisições mais exigentes no sistema de 10% para 50% do total, percebe-se uma aproximação das médias de latências oferecidas tanto para usuários de classe A quanto aos de classe B, como ilustrado na Figura 5.3.

A pequena discrepância (dW_{AB}) entre as exigências é o fator principal que impede a garantia de boa diferenciação entre os serviços, deixando o nível de qualidade ser mais influenciado pela proporção de requisições de cada classe. No entanto, essa pouca diferenciação não significa descumprimento de contrato, pelo contrário, as médias reais de ambas as classes se encontram abaixo das estipuladas previamente.

Esse aumento de requisições do tipo A reflete em uma pequena elevação da média de latência oferecida aos usuários dessa classe, devido à maior carga imposta ao sistema,



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.3: ***Comparativo entre escalonadores - Cenário 50% A e 50% B com 5% variação contratual***

como rege a equação 4.13. Em contrapartida, a diminuição da proporção de requisições do tipo B influencia em um peso menor imposto por tal exigência de serviço, contribuindo para manter boa média das latências médias oferecidas para seus usuários.

Se comparar o escalonamento oferecido pela *EBS* com o da *EDF*, como ilustrado na Figura 5.4, pode-se notar que a preocupação com a exigência como um todo imposta ao sistema, por parte da disciplina *EBS*, proporcionou melhores níveis de qualidade de serviço para as classes. As médias oferecidas pela *EDF* ficaram mais próximas das contratadas pois esse é o objetivo da política: cumprir os *deadlines* de acordo com o especificado por cada contrato de serviço. Mas essa preocupação apenas com um fator pode significar uma dificuldade maior de gerenciamento dos *deadlines* quando o sistema apresentar grande número de requisições urgentes, como se pode perceber comparando-se as figuras 5.2 e 5.4.

Normalmente, em um sistema com suporte a *QoS*, é esperado que o número de requisições de alta prioridade não seja preponderante. Caso contrário, o peso imposto ao sistema poderá crescer de tal forma que mesmo o emprego de uma quantidade maior de recurso pode não ser suficiente para a garantia de bom atendimento a todas as classes de serviço, inclusive às mais prioritárias. Mas, para analisar qual seria o comportamento do escalonador em um cenário como este, com predomínio de requisições de classe alta mas com pouca exigência, simulou-se o experimento ilustrado pela Figura 5.5, onde o sistema está sobrecarregado por requisições do tipo A em 90%.

Pode-se observar que mesmo com menor exigência contratual, um usuário do tipo B obtém melhores valores médios de latência de sistema, o que sinaliza uma inversão da qualidade de serviço relativa. Isto ocorre pelo fato da diferença entre as exigências das classes ser muito pequena, o que dificulta a distinção do grau de urgência das requisições por parte do escalonador. Além disso, outro fator preponderante para as melhores médias

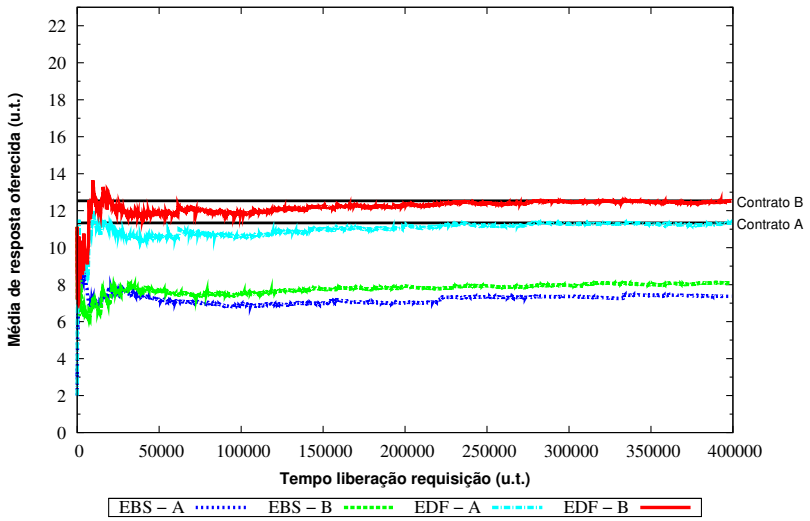
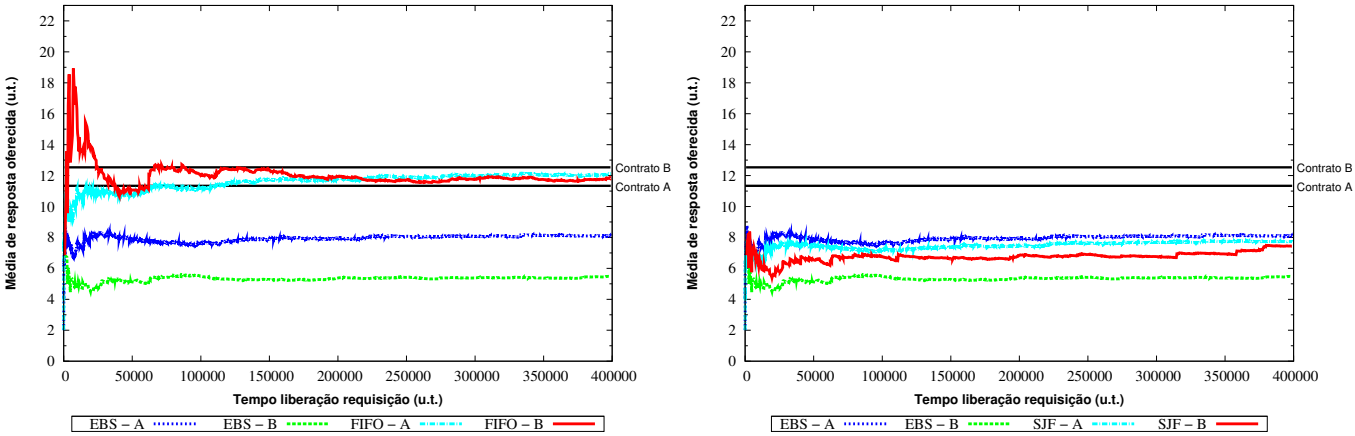


Figura 5.4: ***Comparativo entre EBS e EDF - Cenário 50% A e 50% B com 5% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.5: ***Comparativo entre escalonadores - Cenário 90% A e 10% B com 5% de variação contratual***

oferecidas às requisições do tipo B é que o número destas no sistema é menor se comparado aos das de classe A, o que está de acordo com a equação (4.18).

No entanto, esta inversão de prioridade relativa não influência negativamente no cumprimento dos contratos. Os valores reais de latência de sistema oferecidos para ambas as classes são melhores do que os exigidos, o que mostra o bom desempenho proporcionado pela disciplina *EBS*, que além de garantir a qualidade de serviço contratada, oferece também valores de serviço equiparáveis ou melhores que os proporcionados por políticas como *FIFO* e *SJF*.

Nesse cenário saturado por requisições mais exigentes, comparando-se as médias oferecidas pelos algoritmos *EBS* e *EDF* pode-se perceber a melhor estabilidade[2] e qualidade

[2]No gráfico, a pouca variação das médias de latência expressa maior estabilidade de uma política.

de serviço oferecido pela política proposta para ambas as classes. A preocupação estrita com o *deadline* da política *EDF* impõe maior dificuldade de escalonamento quando a fila de entrada do sistema está cheia de requisições urgentes, mesmo com baixa variação contratual dos contratos de serviço.

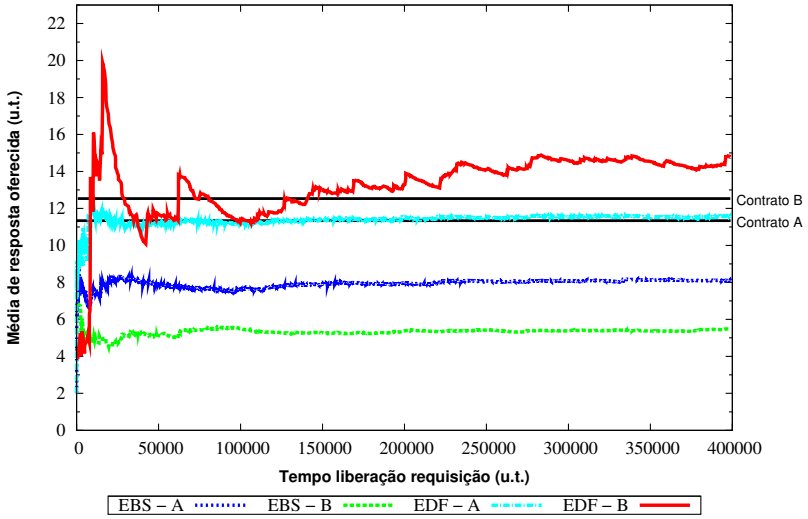


Figura 5.6: ***Comparativo entre EBS e EDF - Cenário 90% A e 10% B com 5% de variação contratual***

Como foi discutido anteriormente, e fica evidente pela Figura 5.6, para um escalonamento do tipo *EBS*, em casos de sobrecarga de requisições de alta prioridade, o fator **exigência** que define as classes de serviço, quando baixo, é preponderante para garantir boas médias de latência e considerável qualidade de serviço.

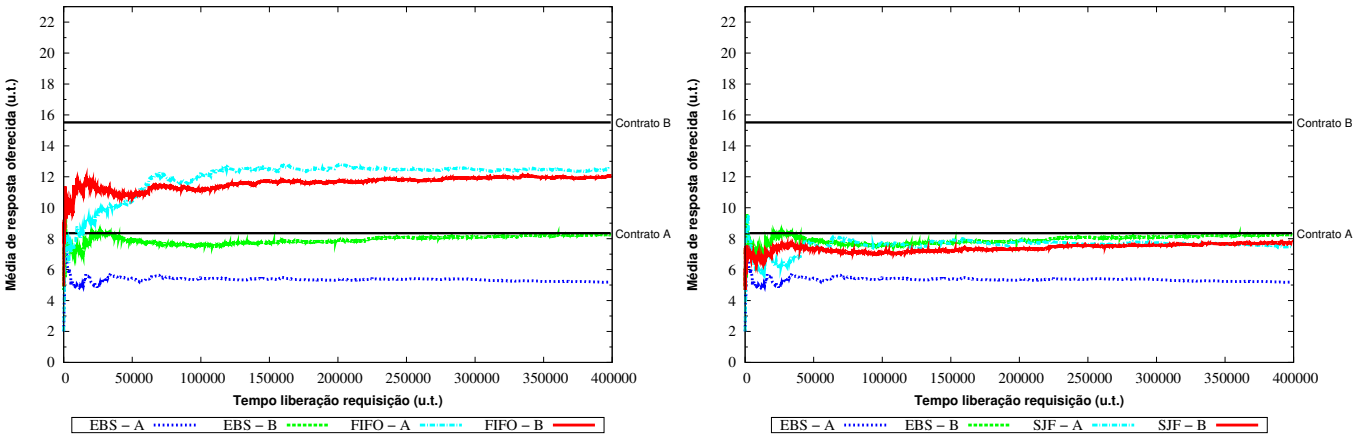
30% Variação Contratual

Estabelecer uma variação contratual de 30% em relação ao tempo de residência oferecido por um sistema de melhor esforço (*FIFO*) significa que o escalonador *EBS* pode trabalhar com a meta de garantir uma média de latência do sistema, 30% melhor para requisições mais exigentes (classe A). Para que isto ocorra, o contrato da classe B precisa oferecer uma tolerância maior em seus valores de residência média no sistema, de forma a contrabalancear a demanda de desempenho.

A fim de verificar como este aumento de tolerância é percebido pelos usuários da classe B, ou seja, quanto que suas médias de latência pioram para que usuários da classe A recebam melhor atendimento, simularam-se outros cenários propriamente ajustados, que podem ser visualizados a seguir. A partir das figuras 5.7, 5.9 e 5.11 mostra-se a capacidade do escalonador em garantir aumento da exigência e também a variação da qualidade do atendimento sob diferentes proporções entre classes.

Em cenários onde o número de requisições de alta prioridade é bastante inferior ao de baixa, como ilustra a Figura 5.7, pode-se constatar que as médias reais de latência do

sistema ficam melhores que as contratadas para ambos os tipos de serviço quando a política *EBS* é empregada. Isto ocorre, pois, apesar da exigência da classe A ser relativamente alta, sua baixa proporção de requisições contrabalanceia a carga total imposta ao sistema, como estabelece a equação **peso do sistema** (equação 4.13).



(a) FIFO X EBS (b) SJF X EBS

Figura 5.7: ***Comparativo entre escalonadores - Cenário 10% A e 90% B com 30% de variação contratual***

Esta menor carga se reflete em um melhor escalonamento, pois a baixa probabilidade de existir requisições urgentes em fila diminui as chances de atrasos das demais. Isto possibilita que todos os contratos de serviço sejam devidamente cumpridos e com valores bem abaixo dos especificados. As requisições da classe B apresentam uma qualidade de serviço melhor do que a especificada por contratos mais prioritários (como os da A). Mesmo quando comparada à política *SJF* o escalonamento proporcionado pela *EBS* também apresenta resultados equiparáveis ou melhores, possibilitando oferecer melhores tempos de resposta aos usuários.

A Figura 5.8 apresenta um comparativo de comportamento entre *EBS* e *EDF* para esse mesmo cenário. Pode-se perceber que a grande proporção de requisições mais tolerantes (classe B) diminuiu a dificuldade de gerenciamento de *deadlines* imposta a ambas as políticas. Mas, a melhor capacidade de gerenciamento da qualidade de serviço continua sendo da *EBS*, a qual ofereceu médias de latência de requisições da classe B melhores em relação às especificadas para a classe A.

Quando a proporção de requisições mais exigentes aumenta para 50%, como ilustra a Figura 5.9, percebe-se um aumento das médias de latências oferecidas se comparado ao cenário anterior, em razão da maior exigência imposta ao escalonador. Valores de exigência maiores, quando potencializadas pelo aumento da proporção de requisições do tipo A, resultam em maiores cargas impostas ao sistema, como estabelecido pela métrica (equação 4.13). No entanto, pode-se constatar também que os serviços oferecidos pela *EBS* tanto

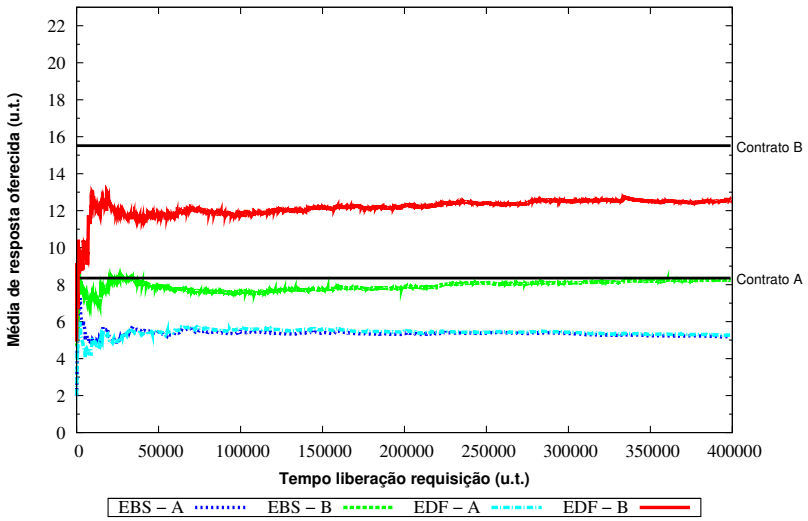
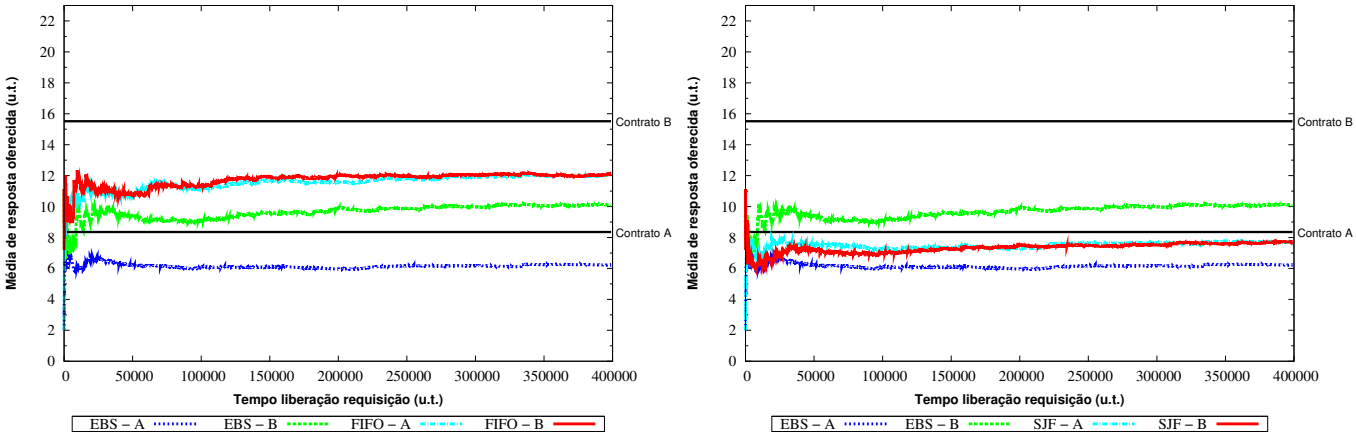


Figura 5.8: ***Comparativo entre EBS e EDF - Cenário 10% A e 90% B com 30% de variação contratual***

para usuários da classe A quanto da classe B continuam com qualidade consideravelmente superior aos especificados pelos seus respectivos contratos.



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.9: ***Comparativo entre escalonadores - Cenário 50% A e 50% B com 30% de variação contratual***

Quando comparado com a política *SJF* o serviço prioritário oferecido pelo escalonamento *EBS* continua sendo superior em termos de qualidade e garantia de diferenciação, com valores de médias bem menores dos que observados no outro escalonador. No entanto, para garantir tal grau de diferenciação, e qualidade, observa-se também um aumento das médias reais da classe B, devido ao aumento da tolerância contratual. Com isto o nível da média de latência oferecida para essas requisições aumenta um pouco se comparadas às proporcionadas pelo algoritmo *SJF*. Entretanto, mesmo um pouco mais elevados, seus valores ainda ficam abaixo do especificado a sua classe de serviço e próximo do especificado à classe de alta prioridade, garantindo bom nível de desempenho.

O aumento da proporção de requisições mais exigentes no sistema tem um impacto maior sobre o escalonamento oferecido pela política *EDF*, como ilustra a Figura 5.10. Enquanto a disciplina *EBS* apresenta melhores médias de latência, os valores oferecidos pela *EDF* ficam no limiar aceitável de cada classe. Isso evidencia maior dificuldade de cumprimento contratual quando a preocupação é unicamente com fatores tais como o *deadline*. A preocupação com as exigências impostas pelas demais requisições é importante para se obter melhores médias de serviço, e conseqüentemente para maior estabilidade da qualidade de serviço oferecida.

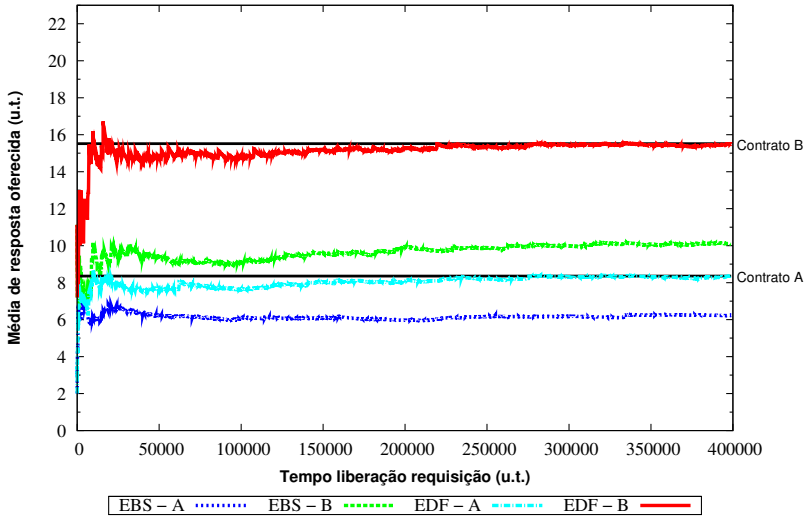
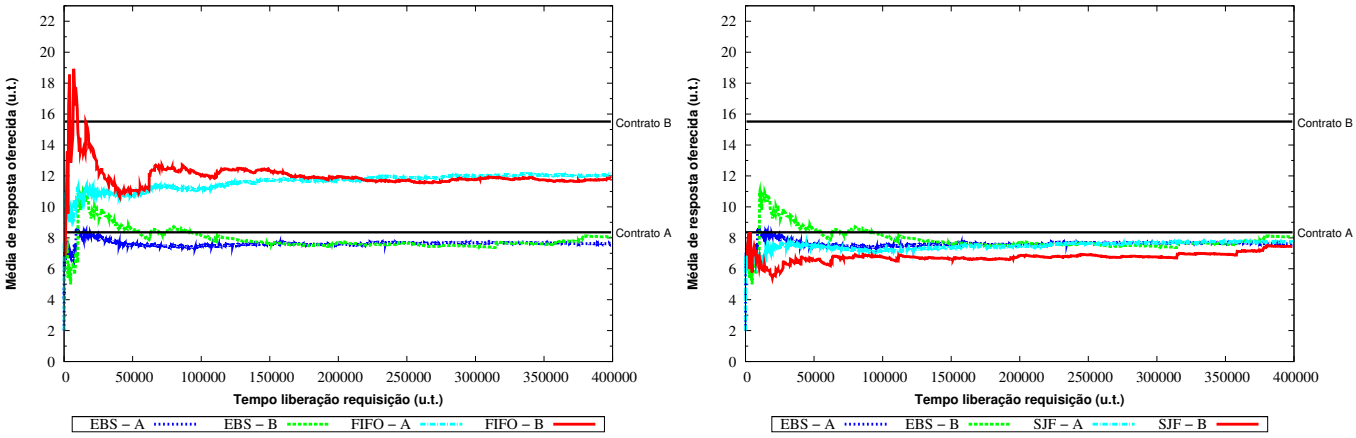


Figura 5.10: ***Comparativo entre EBS e EDF - Cenário 50% A e 50% B com 30% de variação contratual***

Como discutido anteriormente, o aumento da proporção de requisições de alta prioridade tende a influenciar negativamente nos valores de média oferecidos pelos diversos serviços do sistema, pois a exigência imposta ao sistema tende à máxima. Quando o grau de exigência não é tão alto, como mostrado nos cenários com 5% de variação contratual, o peso imposto ao sistema é amortizado.

No entanto, como ilustrado na Figura 5.11, um aumento considerável da exigência contratual, em um cenário saturado de requisições de alta prioridade, tende a causar certo aumento das médias de residência no sistema das requisições de ambas as classes. As do tipo *A* são as mais afetadas, uma vez que o número de requisições de baixa prioridade é insuficiente para oferecer "brechas" de reordenação às requisições mais urgentes, que predominam na fila de escalonamento.

Mas mesmo em cenários saturados, os quais não se espera que aconteçam com freqüência em um bom projeto de implantação de servidor *Web* com gerenciamento de qualidade, a política *EBS* se mostra eficaz em garantir as margens contratuais para ambas às classes de serviço. Comparado a um escalonamento *Web* convencional esse algoritmo oferece, além de diferenciação de serviço, altos níveis de qualidade para os valores de resposta de



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.11: ***Comparativo entre escalonadores - Cenário 90% A e 10% B com 30% de variação contratual***

seus serviços, semelhante a um escalonador possivelmente ótimo, em termos de oferecer mínimas médias latências de sistema: como o *SJF*.

Para ilustrar essa estabilidade na provisão da qualidade dos serviços, a Figura 5.12 apresenta um comparativo entre *EBS* e *EDF* quanto ao quesito médias de latência oferecidas. A grande proporção de requisições de classe A, e a alta variação contratual, dificultam o cumprimento de contratos por parte da política *EDF*, a qual faz o melhor possível para classes mais exigentes, afetando negativamente a qualidade de serviço oferecida à classe B.

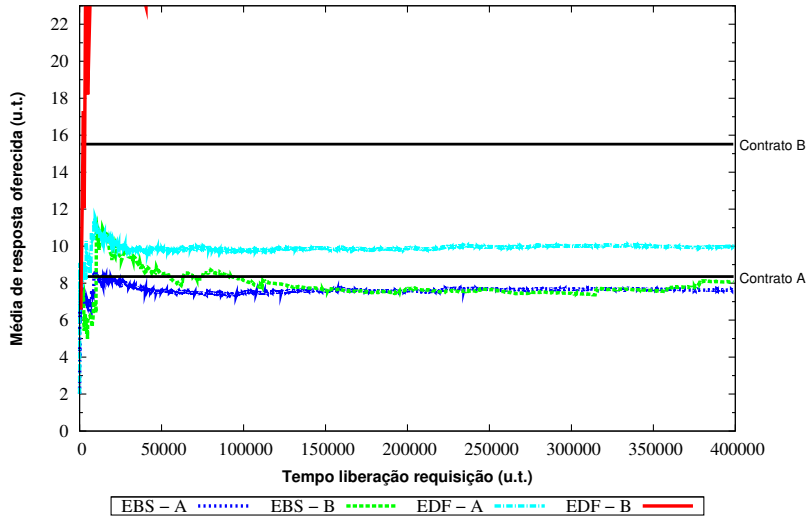


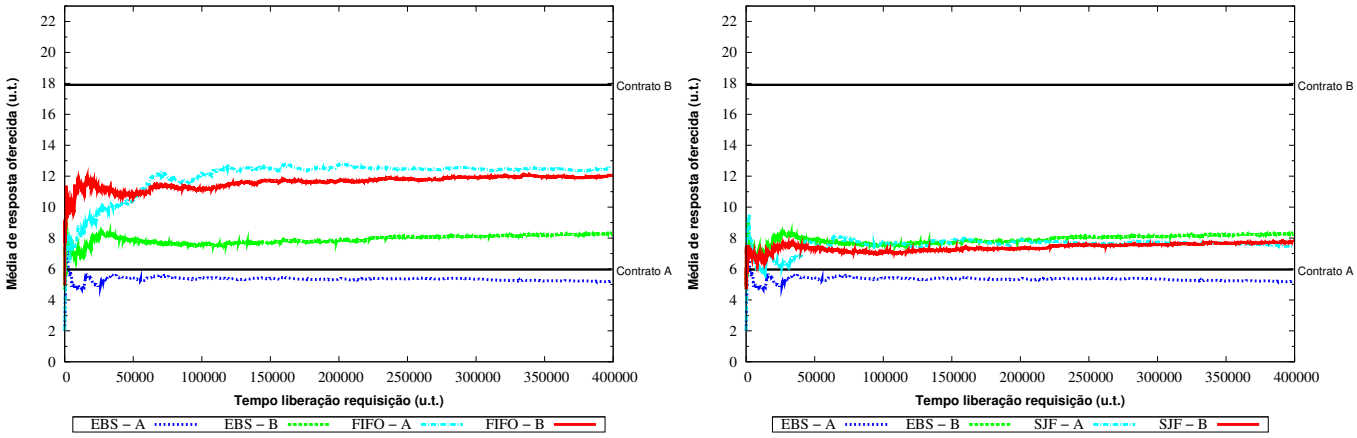
Figura 5.12: ***Comparativo entre EBS e EDF - Cenário 90% A e 10% B com 30% de variação contratual***

A política *EBS*, ao ponderar a urgência das requisições e o peso imposto ao sistema como um todo, conseguiu lidar melhor com a situação de sobrecarga, oferecendo médias de latência melhores que as especificadas para ambas as classes.

50% Variação Contratual

A fim de avaliar um caso extremo de influência das exigências dos serviços disponibilizados sobre a qualidade do escalonamento, estabeleceu-se variações contratuais de 50% em relação ao *FIFO*. Dessa forma requisições da classe A demandam um nível de qualidade de serviço 50% melhor, enquanto as da classe B toleram médias contratuais 50% maiores.

Assim como em cenários anteriores (figuras 5.1 e 5.7) também se fixou a quantidade de requisições de alta prioridade a uma taxa bem inferior ao das de classe B, de forma a representar apenas 10% do total, como ilustra a Figura 5.13.



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.13: ***Comparativo entre escalonadores - Cenário 10% A e 90% B com 50% de variação contratual***

Esse cenário, se comparado com os anteriores do tipo, apresenta pouca alteração das médias de latências de ambas as classes, mesmo com considerável variação contratual. Uma das razões que justificam isto é a baixa proporção de requisições de alta prioridade, que reduz o efeito da alta exigência sobre a carga total do sistema. Além desse, outro fator que contribui para a pequena alteração das médias é a grande tolerância especificada para contratos de classe B. Como visto pela equação (4.9), mesmo em situações em que a quantidade de requisições de baixa prioridade aumenta o peso total do sistema não sofre uma significativa alteração se a exigência dessa classe (B) diminuir, explicando assim a pouca variação das médias.

O aumento da tolerância do contrato B, e a grande proporção de requisições desse tipo no sistema, facilitam o gerenciamento dos *deadlines* por parte da política *EDF*, como ilustra a Figura 5.14. A maior flexibilidade contratual impõe menor peso ao sistema (como regido pelas equações (4.1) e (4.3)), facilitando o gerenciamento dos *deadlines*. Aproveitando indiretamente esse menor peso do sistema, a disciplina *EDF* apresenta um comportamento similar ao da *EBS*, evidenciando assim a importância de se considerar a exigência do sistema como um todo para se obter melhores níveis de qualidade de serviço.

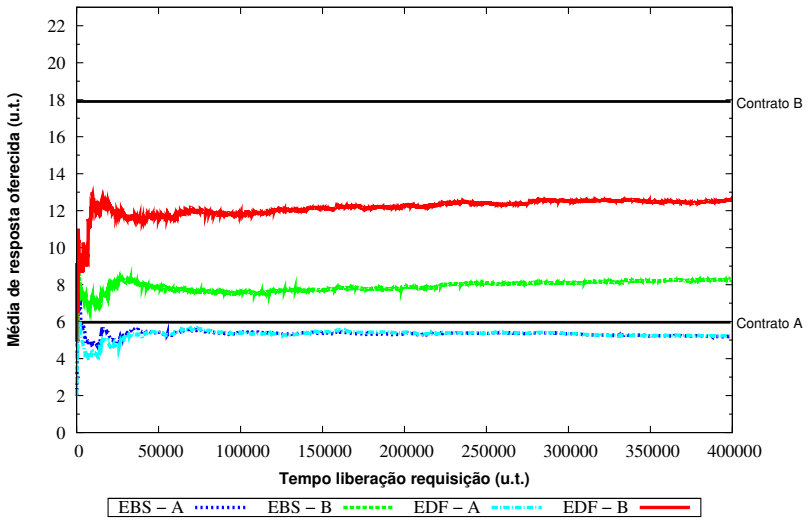


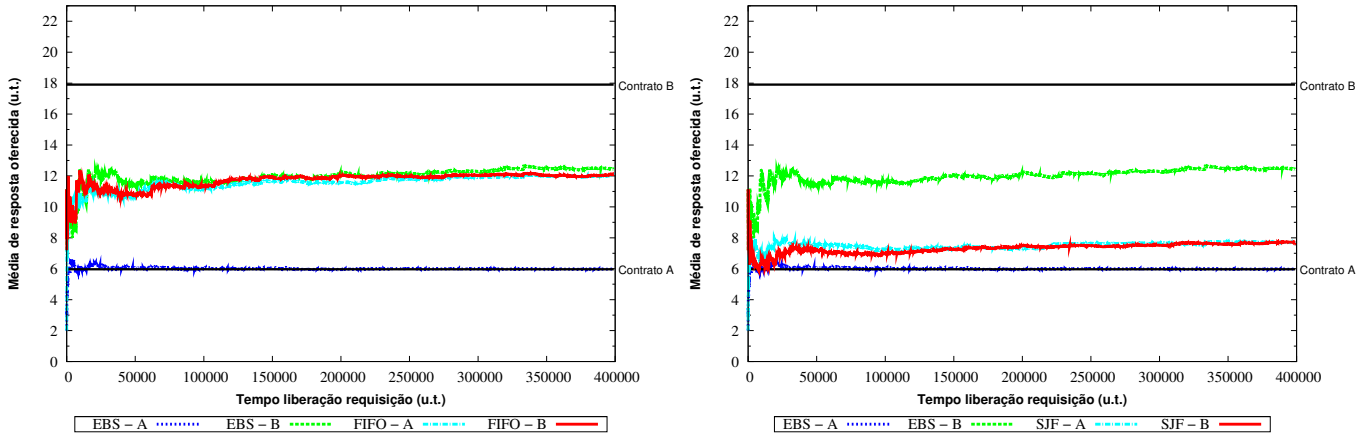
Figura 5.14: ***Comparativo entre EBS e EDF - Cenário 10% A e 90% B com 50% de variação contratual***

Cenários onde a proporção de requisições prioritárias são inferiores em quantidade ao das demais podem ser considerados mais realistas pois oferecem maiores garantias quanto ao gerenciamento de recurso e controle de qualidade de serviço. E, nesses casos, mesmo sob diferentes variações contratuais a política *EBS* tem mostrado boa capacidade de garantir os níveis de *QoS* contratados, além da melhora de desempenho para as médias de latências oferecidas.

Para verificar como essa capacidade de provisão de *QoS* se comporta quando a participação de requisições de alta prioridade sobe para 50% simulou-se outro cenário (Figura 5.15). Como ilustrado, o aumento da proporção de requisições com grande exigência de serviço provoca um aumento das médias de latências de sistema oferecidas pelo escalonador. No entanto, mesmo assim, a qualidade de serviço das classes é mantida, se considerado um limiar de tolerância para ambas as classes.

Apesar da média oferecida para requisições de classe B aumentar em relação a outros cenários de mesma proporção (figuras 5.3 e 5.9) o escalonamento *EBS* se mostrou eficiente em garantir os parâmetros contratuais de ambas às classes. O desempenho foi superior a um convencional (*FIFO*), mas com médias de latência não tão baixas quanto às do *SJF*. Essa diminuição de desempenho é justificada pelo aumento do grau de diferenciação, devido à maior discrepância contratual, definida opcionamente assim como forma de ser mais justa quanto ao nível de qualidade de serviço oferecido para cada classe. Requisições do tipo A, consideradas prioritárias, receberam um nível de qualidade de serviço melhor em relação aos oferecidos pelas outras políticas, mostrando notável eficiência na oferta de *QoS* por parte da disciplina *EBS*.

Comparada à política *EDF*, ilustrada na Figura 5.16, a *EBS* oferece melhores níveis de qualidade devido ao aumento de desempenho obtido com menores médias de serviço,



(a) FIFO X EBS (b) SJF X EBS

Figura 5.15: ***Comparativo entre escalonadores - Cenário 50% A e 50% B com 50% de variação contratual***

oferecidas para ambas as classes. Isso permite que mesmo com alta exigência, requisições de classe A não causem muita degradação de qualidade em uma classe menos exigente como a B.

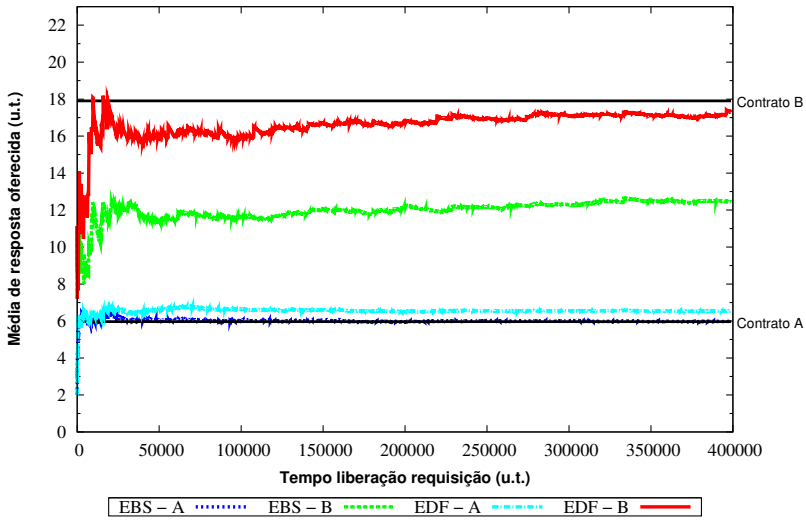
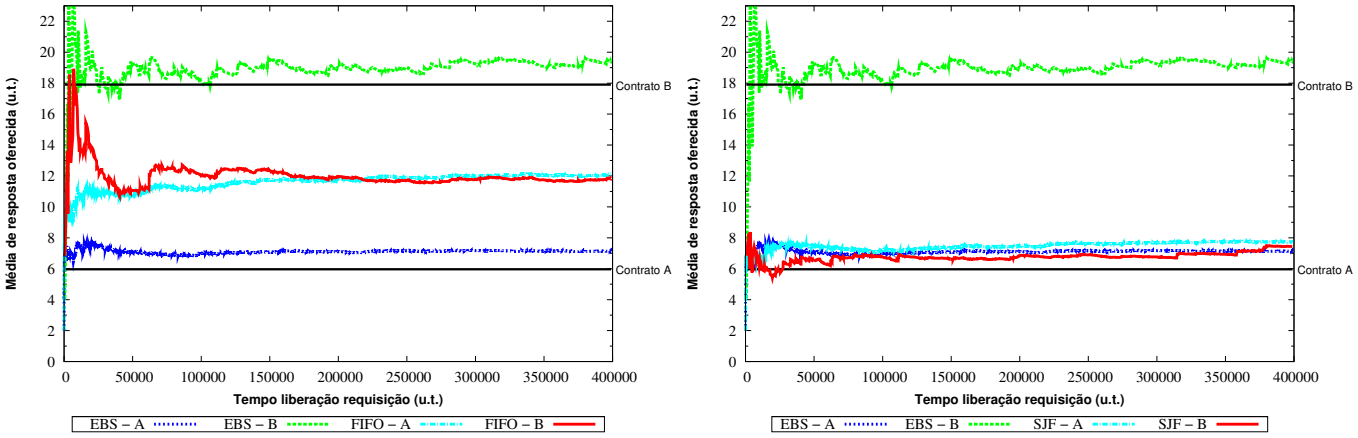


Figura 5.16: ***Comparativo entre EBS e EDF - Cenário 50% A e 50% B com 50% de variação contratual***

Por fim, para verificar o comportamento em um cenário totalmente saturado por requisições com baixa latência contratada e submetido à alta variação de exigência executou-se o experimento ilustrado pela Figura 5.17.

Como se pode observar, apesar de não conseguir garantir médias de latências abaixo dos patamares contratados, o escalonador *EBS* fez o possível para oferecer melhor atendimento às requisições do tipo A em relação às do tipo B, sem deixar de atender estas últimas para garantir diferenciação. Novamente, utilizando-se a equação (4.9) pode-se notar que o peso total de um sistema sofre grande impacto quando a exigência e a pro-



(a) FIFO X EBS

(b) SJF X EBS

Figura 5.17: ***Comparativo entre escalonadores - Cenário 90% A e 10% B com 50% de variação contratual***

porção das requisições de uma dada classe aumentam muito. A grande proporção de requisições de mais exigência potencializa o impacto da discrepância de pesos entre as classes (dW_{AB}), afetando os níveis de qualidade oferecidos.

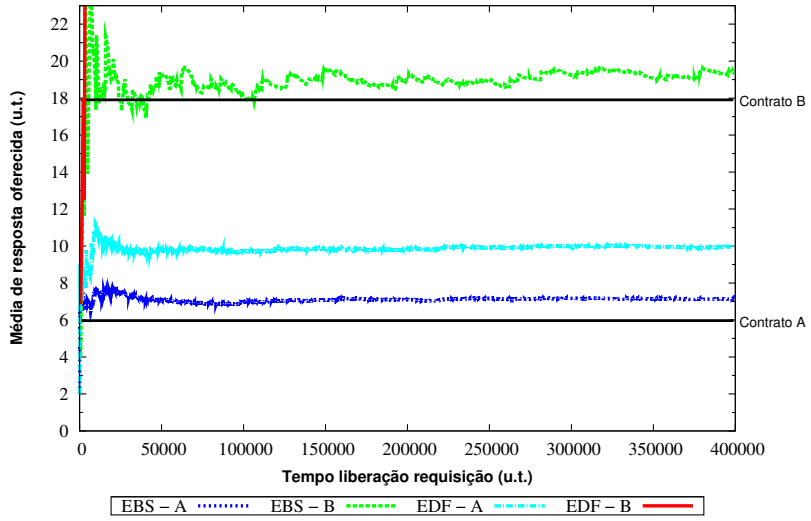


Figura 5.18: ***Comparativo entre EBS e EDF - Cenário 90% A e 10% B com 50% de variação contratual***

O melhor equilíbrio de qualidades entre os serviços oferecidos pela *EBS* pode ser percebido comparando-a com um escalonamento *EDF* do mesmo cenário. Como ilustra a Figura 5.18, ao preocupar-se somente com as urgências individuais das requisições, sem levar em conta o peso delas sobre a qualidade de serviço do sistema, degrada-se muito as médias de latência oferecidas para ambas às classes, principalmente a das menos exigentes.

Assim, quando sob alta presença de requisições prioritárias não é interessante elevar muito o grau de diferenciação oferecido aos serviços de um sistema pelo fato que estes dois fatores agem em conjunto sobre o peso imposto ao escalonador. Um fator poten-

cializa o efeito do outro, o que faz com que a exigência imposta ao escalonador torne-se excessivamente alta. Com isto, a fila de escalonamento fica saturada por requisições urgentes, presentes em grande número, o que dificulta a melhor escolha de atendimento se o escalonador precisar manter a diferenciação a todo custo.

5.4 Análise quanto a Satisfação do Usuário

A análise dos resultados por estudo de casos, como realizada no tópico anterior, é importante pois fornece detalhes do comportamento do escalonador e da qualidade das médias de latência oferecidas. No entanto, uma visão mais abrangente, com maior validade estatística também se faz necessária. Por ela pode-se constatar se a eficiência e desempenho vistos nos gráficos anteriores são ou não casos isolados.

Dados de satisfação dos usuários e da variação de suas latências individuais em relação às médias contratadas foram coletados, de acordo com o planejamento especificado na seção 5.2. Apresentados em termos percentuais, esses valores foram calculados a partir de uma faixa de tolerância de 3% em relação aos parâmetros contratuais.

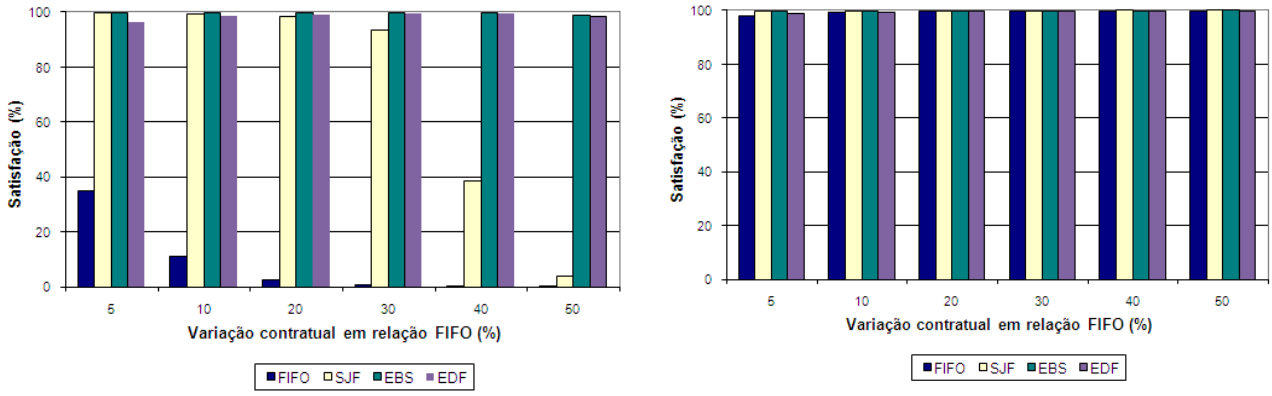
A satisfação de um dado usuário mostra o percentual de vezes que a média real oferecida ficou dentro da faixa de tolerância especificada por cada contrato de serviço. O número de latências oferecidas abaixo da média contratada quantifica a variação da qualidade de um dado atendimento, pois, em média, a satisfação pode se manter alta, mas os valores individuais de latência oferecidos podem variar muito. Esse percentual reflete a estabilidade da qualidade do escalonamento oferecido.

10% requisições prioritárias

A Figura 5.19 apresenta um comparativo da satisfação contratual obtida pelas diferentes disciplinas de escalonamento, num cenário onde a proporção de requisições no sistema é composta por 10% de classe A e 90% de classe B.

A taxa de satisfação para usuários da classe B é praticamente plena para todas as políticas. Quando analisada a satisfação para usuários da classe A verifica-se alto grau de satisfação para todas as variações contratuais quando as disciplinas *EBS* e *EDF* são aplicadas. A disciplina *SJF* oferece ótimas taxas para variações até 30%, a partir da qual seus percentuais sofrem forte decréscimo devido à incapacidade em garantir qualidade de serviço, pois sua meta é a otimização de desempenho. Isso mostra que além de oferecer bons níveis de desempenho, a política *EBS* também é eficiente em garantir amplas taxas de diferenciação de serviço quando a proporção de requisições prioritárias não é predominante no sistema.

A Tabela 5.1 apresenta os desvios-padrão para as médias de satisfações ilustradas pela Figura 5.19. Diferentemente das demais políticas, a *EBS* consegue oferecer maior esta-



(a) Classe A

(b) Classe B

Figura 5.19: ***Satisfação Contratual - Cenário 10% A e 90% B***

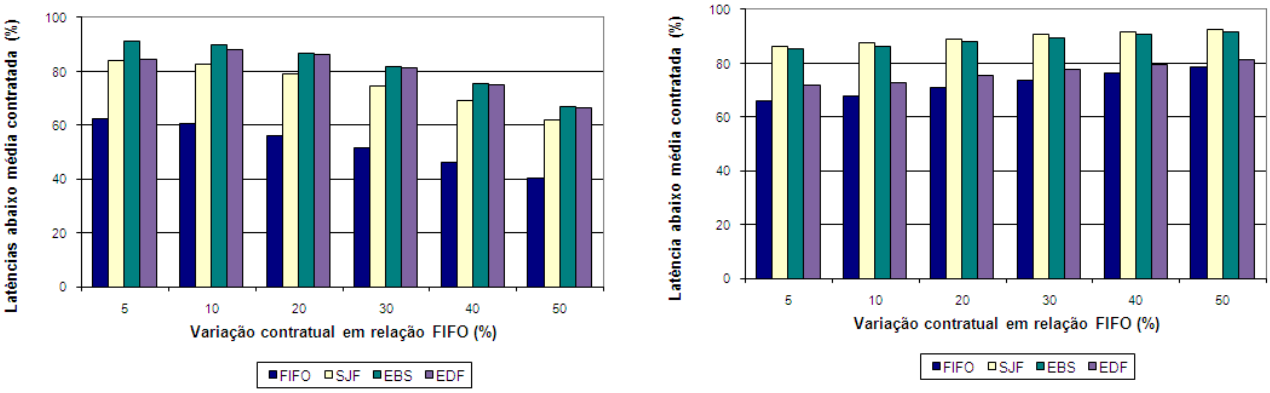
bilidade da qualidade de seus serviços para todas as variações de exigências contratuais, e para ambas as classes.

Variação Contratual(%)	Desvio Padrão (%)							
	Classe A				**Classe B**			
	FIFO	SJF	EBS	EDF	FIFO	SJB	EBS	EDF
5	30.50	1.18	0.19	4.88	2.73	0.08	0.01	1.46
10	15.40	1.61	0.18	2.63	1.39	0.06	0.07	0.81
20	2.75	4.21	0.23	1.43	0.54	0.03	0.06	0.41
30	1.34	15.83	0.19	1.14	0.27	0.02	0.03	0.23
40	0.67	32.92	0.39	0.76	0.15	0.01	0.02	0.13
50	0.32	4.77	2.23	2.32	0.08	0.01	0.02	0.07

Tabela 5.1: ***Desvio Padrão da Satisfação Contratual no Cenário 10% A e 90% B.***

Pela Figura 5.20 pode-se verificar como varia a qualidade individual de atendimento das requisições para este cenário de 10% de requisições de classe A. No escalonamento *EBS*, mesmo sob grandes variações contratuais das exigências, a qualidade do escalonamento se mantém em um bom nível para a maior parte das requisições. Para variações contratuais de até 30% a qualidade individual das latências reais é melhor que a contratada em mais de 80% dos casos, o que mostra que, mesmo sob graus mais altos de diferenciação, o algoritmo *EBS* consegue manter ótima qualidade de serviço, quando o sistema não está saturado por requisições prioritárias. Comparadas às demais políticas, a *SJF* consegue oferecer pouca oscilação de qualidade para classes prioritárias, mas com pouca diferenciação de serviço, apenas para contratos até 10% mais exigentes. A política *EDF* apresenta qualidade de latência similar à *EBS* quando analisada a oferecida para classe A, mas inferior quando comparada em um atendimento de classe B.

Para a classe B ambas as políticas (*EBS* e *SJF*) garantem pouca variação de qualidade, independente das taxas contratuais oferecidas, pois, à medida que essas aumentam, os contratos se tornam mais tolerantes.

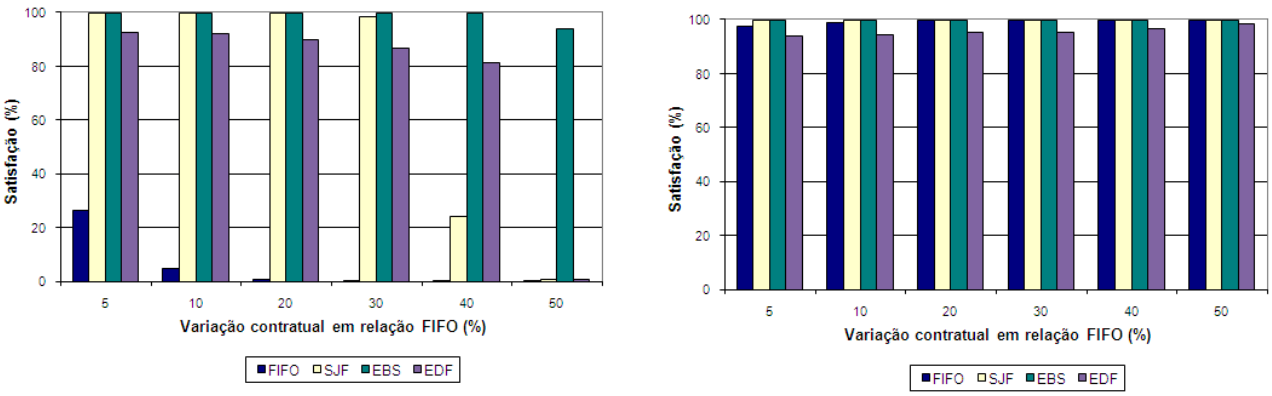


(a) Classe A (b) Classe B

Figura 5.20: ***Latências abaixo média contratada - Cenário 10% A e 90% B***

50% REQUISIÇÕES PRIORITÁRIAS

Em casos aonde o número de requisições prioritárias aumenta, mas não chegam a ser preponderantes no sistema, apresentando uma participação de 50% das requisições totais, pode-se verificar como fica a satisfação dos usuários para diferentes níveis de exigência contratual (Figura 5.21). A satisfação proporcionada pela política *EBS* se mantém estável e praticamente plena para todas as variações contratuais simuladas, para ambas as classes. Isso só é alcançado pelas outras disciplinas quando se avalia a satisfação de usuários da classe *B*. Para classe *A* a *SJF* e a *EDF* oferecem valores satisfatórios, mas limitados por uma variação contratual de, no máximo, 30% e 40%, respectivamente. Mesmo com o aumento da proporção de requisições prioritárias a satisfação oferecida por um escalonamento *EBS* se manteve estável, se comparado com o cenário apresentado pela Figura 5.19, onde o número de tais requisições é menor.



(a) Classe A (b) Classe B

Figura 5.21: ***Satisfação Contratual - Cenário 50% A e 50% B***

Essa maior estabilidade se deve ao fato da *EBS* se preocupar com o impacto da exigência de cada requisição a ser atendida sobre o sistema como um todo, influenciando assim na satisfação das demais requisições. Essa influência da exigência de uma dada classe sobre a satisfação das demais também é evidenciada pela equação (4.19), na qual a discrepância de pesos entre as requisições tem relação direta com a satisfação.

Os desvios-padrão para os dados da Figura 5.21 são apresentados pela Tabela 5.2, e como os apresentados anteriormente, estes também se mantêm baixos e com pouca variação para o escalonamento *EBS*, exceto para aumentos de exigência de 50% para classe A. Mas, mesmo este, não apresenta significativa variação de satisfação. Os valores apresentados pela *EDF* demonstram maior instabilidade da qualidade de serviço oferecida pelo escalonador, a qual fica evidenciada com os gráficos de comportamento que ilustraram maior dificuldade de gerenciamento dos *deadlines* em cenários de maior exigência. A maior instabilidade da qualidade de serviço oferecida pela política *EDF* é evidenciada pelos altos valores de desvios-padrão apresentados pela Tabela 5.2.

Variação Contratual(%)	Desvio Padrão (%)							
	Classe A				**Classe B**			
	FIFO	SJB	EBS	EDF	FIFO	SJB	EBS	EDF
5	24.65	0.18	0.19	9.61	3.81	0.17	0.17	8.61
10	5.53	0.23	0.20	10.96	1.66	0.13	0.16	7.90
20	0.95	0.60	0.23	11.77	0.71	0.07	0.14	5.74
30	0.41	2.98	0.33	13.71	0.37	0.05	0.14	5.65
40	0.21	26.76	0.50	16.96	0.20	0.03	0.12	3.92
50	0.09	1.24	4.20	0.62	0.11	0.02	0.11	2.02

Tabela 5.2: ***Desvio Padrão da Satisfação Contratual no Cenário 50% A e 50% B.***

90% REQUISIÇÕES PRIORITÁRIAS

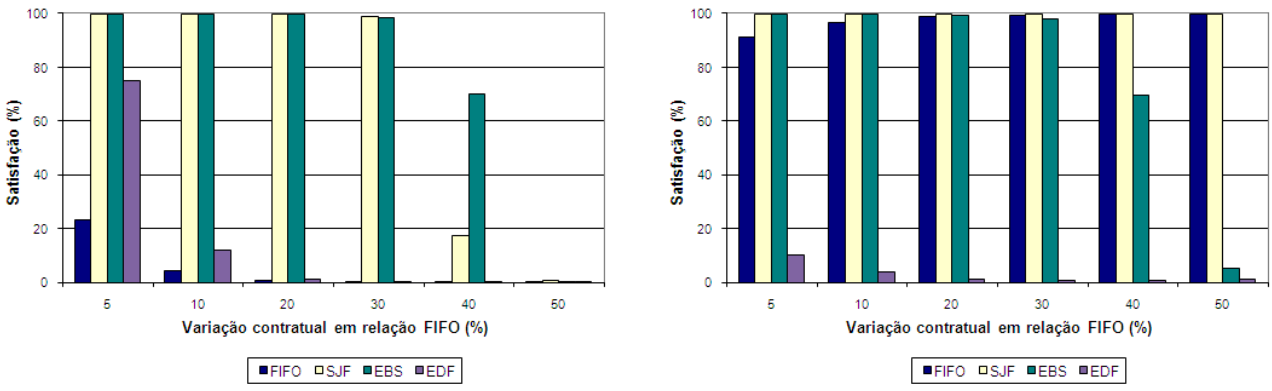
Quando o sistema fica saturado por requisições prioritárias, Figura 5.22, a exigência imposta ao escalonador para garantir as requisições de *QoS* aumenta excessivamente. Para taxas muito altas dessa variação contratual observa-se que o peso imposto ao sistema inviabiliza um bom escalonamento, oferecendo baixa satisfação contratual para usuários mais exigentes. Variações de até 40% oferecem uma satisfação aos usuários prioritários superior a 75%, além de uma sólida diferenciação de serviço. Em contrapartida, por priorizar a classe A, à medida que o grau de priorização cresce, a satisfação oferecida pela política à classe B decresce, principalmente a partir de 40%, quando a carga do sistema atinge níveis altos.

Pela métrica de satisfação definida pela equação (4.11) pode-se constatar este comportamento. A satisfação de classes mais exigentes (S^+) é inversamente proporcional ao peso imposto por elas ao sistema. Assim, quanto menor o valor de sua latência contratual, maior será esse peso, e menor tenderá a ser sua taxa de satisfação. Mesmo sendo alta a discrepância entre os pesos das classes mais e menos exigentes, ela não é suficiente

para influenciar em melhores valores de satisfação devido ao baixo valor do número de requisições bem atendidas dos usuários (N_i), devido ao alto peso imposto ao sistema.

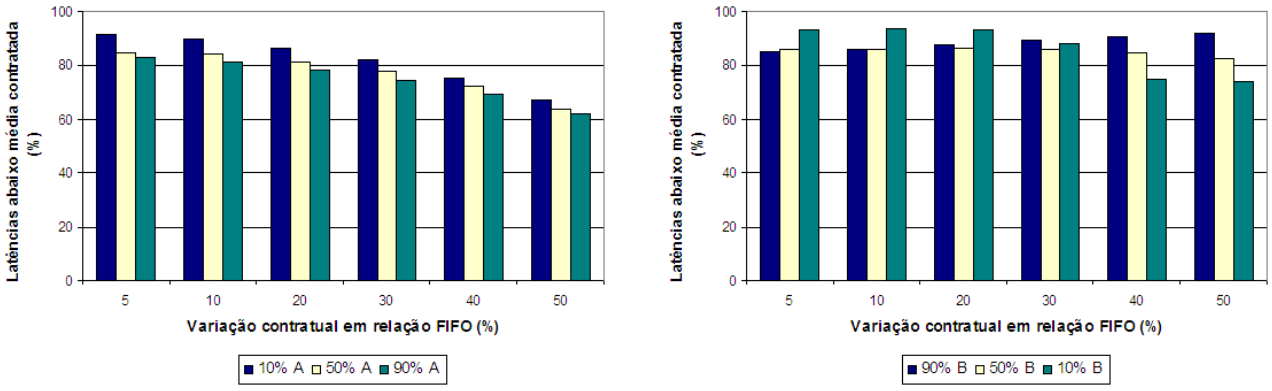
Outra relação importante, que ficou evidenciada pela análise dos gráficos e, pelas equações (4.18) e (4.19), é a influência da proporção das requisições de cada classe sobre o impacto da discrepância de pesos na satisfação das classes. Quanto maior a proporção de uma classe no sistema, maior será o impacto da discrepância de pesos na satisfação das demais. Nos cenários de pouca proporção de classe A, por exemplo, mesmo sob altas variações contratuais as satisfações dos clientes se mantêm mais estáveis.

Como as requisições de classes A e B diferem apenas quanto aos contratos de serviço, e políticas como *FIFO* e *SJF* não levam tais parâmetros em conta na hora de escalonar, diferentes proporções entre estas classes não influenciam significativamente as taxas de latências abaixo da média, apresentando assim valores similares aos apresentados pela Figura 5.20. Por isso a comparação de tais taxas para diferentes proporções de classes se faz relevante apenas para a disciplina *EBS*, como é apresentada pela Figura 5.23.



(a) Classe A (b) Classe B

Figura 5.22: ***Satisfação Contratual - Cenário 90% A e 10% B***



(a) Classe A (b) Classe B

Figura 5.23: ***Latências abaixo média contratada - Política EBS***

É possível notar que para os mesmos valores de exigência contratual, a variação da qualidade dos valores individuais de latência, em relação às diferentes proporções entre classes, é pequena. No entanto, isto não é razão suficiente para garantir pouca variação dos valores de satisfação quando o número de requisições prioritárias é grande e a exigência aumenta.

Quando a proporção de requisições da classe A é pequena (10%) nota-se que o aumento de exigência contratual causa diminuição da taxa de latências bem atendidas, enquanto que essa taxa aumenta para requisições da classe B. À medida que o número de requisições prioritárias aumenta para 50% do total, percebe-se o mesmo decréscimo de qualidade para taxas da classe A quando a exigência aumenta, mas certo equilíbrio para a classe B, sinalizando que o aumento de tolerância não age isoladamente sobre o nível de qualidade. Quando o sistema apresenta saturação de requisições prioritárias (90%), observa-se que as taxas de latências abaixo da média caem com o aumento da exigência para ambas as classes. Isto ocorre, pois, apesar dos contratos dos usuários do tipo B ficarem mais tolerantes, o que poderia significar números maiores de atendimento abaixo da média, verifica-se que o aumento da exigência da classe A é potencializado pela sua proporção maior no sistema. Com isso a probabilidade de requisições de classe B serem preteridas aumenta, causando diminuição da qualidade individual das latências oferecidas.

À primeira vista, com uma análise superficial da Figura 5.23 pode-se inferir que é preferível assinar um contrato do tipo B, pois estes oferecem melhores índices de latências abaixo da média que os de classe A, cujo plano exige maior custo. Entretanto, pela análise dos gráficos de comportamento anteriormente apresentados, constata-se que as médias oferecidas aos usuários da classe A melhoram significativamente à medida que o grau de diferenciação é elevado. Além disso, o maior número de requisições da classe B com latências abaixo da média contratada se deve ao fato dessa classe ser mais tolerante quanto à exigência contratual.

5.5 Considerações Finais

Neste capítulo apresentou-se o planejamento feito para a realização dos experimentos de avaliação da política proposta, bem como os resultados obtidos e a análise realizada. Para organizar a análise, representaram-se os dados obtidos em três tipos de gráficos, um primeiro, descrevendo o comportamento de cada política de escalonamento implementada, tendo por objetivo avaliar a qualidade das médias de latência de sistema oferecidas ao longo do tempo. Um segundo tipo descrevendo a satisfação dos usuários sob diferentes variações de parâmetros do sistema, como proporção de requisições e variação de contratos de serviço. Por fim, um terceiro tipo de gráfico descrevendo a estabilidade da qualidade de serviço oferecida, no qual apresentam-se as latências de sistemas que foram oferecidas dentro do limiar de serviço contratado. Na discussão desses gráficos ficou evidênciado que é

importante realizar um escalonamento baseado na exigência imposta por classe de serviço para a qualidade do atendimento oferecida pelo sistema como um todo, característica essa adotada pela política *EBS*. Os principais pontos de análise levantados nesse capítulo são aproveitados para uma conclusão mais concisa a ser apresentada no próximo capítulo.

Capítulo

6

Conclusão

6.1 Conclusões gerais

Pela análise dos resultados apresentados no capítulo 5 pode-se observar que a política *EBS* proposta, alcança com sucesso o objetivo inicialmente estabelecido: oferecer gerenciamento de *QoS* em termos absolutos para escalonamento no nível de aplicação. Como mostrado, para a grande maioria dos cenários simulados os parâmetros contratuais de serviços de um sistema foram garantidos ao longo do tempo em patamares iguais ou abaixo dos contratados, mostrando eficiência e desempenho no cumprimento das médias de latência.

Englobando características tanto do algoritmo *EDF* (em sistemas de tempo real), quanto do *SJF* (em escalonamento de processos visando baixa latência), a estratégia de escalonamento proposta permite um ajuste ponderado de comportamento entre o desempenho proporcionado pelo *SJF* e o maior grau de diferenciação de serviço oferecido pelo *EDF*, ao tentar escalonar as requisições mais urgentes primeiro. Como ilustrado pelos gráficos de comportamento este ajuste é realizado por meio da variação contratual que define as classes de serviço.

Pela análise dos experimentos pode-se concluir que pequenas variações contratuais proporcionam maior desempenho em um escalonamento *EBS*. Ao diminuir a variação contratual entre os serviços, o escalonador passa a ter uma visão mais homogênea das prioridades das requisições, no sentido de haver uma menor diferenciação da urgência dessas requisições. Com isto, a reordenação da fila de atendimento é regida muito mais em função do custo de processamento do que pela urgência de execução, pois para o escalonador a maioria dessas requisições apresentam *deadlines* bem próximos. Nesses

casos o comportamento do escalonador é moldado pelo desempenho característico do *SJF*.

Como discutido na figura 5.5, quando a variação entre os contratos é menor, a proporção de requisições de uma dada classe no sistema é muito mais decisiva na definição da qualidade de atendimento oferecida do que os parâmetros contratuais. Nesse cenário, por exemplo, requisições de classe B receberam médias de latência melhores que as requisições mais exigentes de classe A, pelo fato de estarem em número bem menores no sistema (10%), mesmo com contratos mais tolerantes quanto à qualidade, respeitando o comportamento esperado em concordância com a equação (4.18).

Assim, menores variações contratuais são preponderantes para alcançar melhor desempenho dos valores de resposta, no entanto diminuem substancialmente a garantia de diferenciação relativa entre classes[1], deixando a qualidade de atendimento de suas requisições ao arbítrio de suas proporções no sistema, e não em função dos contratos de serviço. Isso pode ser um problema, pois se um usuário não tiver um limite de requisições submetidas ao sistema ele pode causar degradação na qualidade de serviço oferecida aos demais. Assim um controle do comportamento do usuário é importante, podendo ser estabelecido por meio de contratos bilaterais de serviço, em que ao mesmo tempo o provedor garante direitos aos usuários, estes têm deveres com o provedor de serviços.

Por outro lado, o aumento desta variação contratual possibilita um controle muito mais rígido da diferenciação relativa entre classes, deixando com que a proporção de requisições de uma dada classe influencie apenas no cumprimento ou não de seu contrato de serviço. Nos cenários com variação contratual de 50% pode-se constatar que em nenhum momento ocorre inversão de prioridades[2], no entanto dependendo da proporção de uma dada classe no sistema o escalonador consegue ou não cumprir os contratos estabelecidos, maior variação contratual e proporção potencializam o efeito da exigência de uma classe sobre a carga total imposta ao sistema (equação 4.3).

Em momentos de sobrecarga de requisições pertencentes às classes mais exigentes, se a satisfação dos usuários diminuir consideravelmente em função do aumento do peso imposto ao sistema, pode-se usar esse mecanismo de ajuste como estratégia de controle/monitoramento de satisfação. Diminuir a variação contratual do ponto de vista do escalonador significaria ganho de desempenho, pois menores valores de médias de latência seriam obtidos, e como consequência, conseguiria-se melhorar a satisfação real dos usuários. Essa diminuição da variação contratual teria efeito virtual apenas para o escalonador, pois os valores reais oferecidos para os usuários estariam em níveis abaixo dos especificados, como bem mostrou os gráficos de comportamento (figuras 5.1, 5.3 e 5.5).

[1]Ressaltando-se que atender com menor latência efetiva a classe com menor latência contratada não é o objetivo da política. O inverso pode ocorrer desde que os contratos sejam atendidos.

[2]Se prioridades fossem definidas em função da latência contratada.

A partir desta análise, a intuição de que baixos valores contratuais são sempre preferíveis pode surgir, pois com eles consegue-se maior desempenho, o que facilita o cumprimento dos contratos de serviço. No entanto, além dessa medida não ser atrativa para usuários que gastam mais com planos que definem menores latências contratuais, acaba significando também um certo desperdício de recurso. Mesmo tendo seus contratos específicos garantidos, o fato de em certos momentos os usuários de classe A pagarem mais e receberem uma qualidade relativa inferior a outros que pagarem menos, pode ser percebido negativamente pelo cliente, causando assim um certo impacto econômico nos planos do provedor de serviço.

Além disso, garantir a maior parte do tempo médias de latência bem melhores às classes menos prioritárias do que seus contratos especificam, e ainda abaixo das mais prioritárias, significa também um desperdício de recurso, pois o excesso de satisfação oferecido a estes usuários poderia ser melhor aproveitado para garantir maior satisfação aos usuários da classe A quando sua proporção crescer no sistema. E baixa variação contratual não permite um controle mais rígido sobre a garantia dessa diferenciação de prioridades, pois a menor discrepância entre as classes torna a priorização mais homogênia.

No entanto, quando não utilizado de maneira indiscriminada, esse mecanismo pode ser utilizado como uma importante ferramenta de controle da qualidade de serviço do sistema como um todo. Como estabelece a equação (4.16), a alteração de valores de latência contratada tem reflexo sobre a taxa de utilização a qual o sistema está submetido, e um sistema sob menor exigência tem mais condições de oferecer um melhor atendimento. Assim, quadros de baixo desempenho podem ser, em algumas situações, contornados por um "sistema de promoção", diminuindo momentaneamente a discrepância contratual entre as classes para que se possa aumentar a satisfação geral. Isso permite um maior controle da qualidade de serviço oferecida aos usuários.

Assim, normalmente é aconselhável fixar uma variação contratual intermediária, não tão baixa quanto a estudada, mas nem tão alta de forma a impor um peso exagerado ao sistema e inviabilizar a garantia da qualidade dos serviços oferecidos. Em momentos de sobrecarga de requisições de alta prioridade, reduzir essa variação pode estabilizar o nível de satisfação oferecido aos diversos usuários.

6.2 Contribuições

A política de escalonamento proposta neste trabalho permitiu uma abordagem inovadora de provisão de *QoS* em serviços *Web* ao relacionar o gerenciamento da exigência imposta ao sistema com a possibilidade de se conseguir melhor qualidade de serviço de suas classes. Geralmente se considera apenas características individuais nos mecanismos de provisão de qualidade de serviço, sem se preocupar com o impacto das decisões de escalonamento na qualidade dos demais serviços. No entanto, com a análise dos resul-

tados obtidos pode-se perceber a importância do impacto da exigência imposta por uma classe sobre a qualidade de serviço oferecida para as demais. Ao escalonar no sentido de minimizar o impacto das exigências das classes de serviço sobre o sistema como um todo, a política *EBS* obtém menores médias de latência oferecidas, o que permite maior controle da qualidade de serviço.

Outro diferencial é que, sintetizando características de escalonamento de tempo real, com baixa latência e de modelo re-alimentado, a política *EBS* oferece uma abordagem multi-diciplinar para provisão de novos níveis de confiabilidade às aplicações *Web*. Ao tratar os requisitos de *QoS* em termos quantitativos (absolutos), a política proposta oferece garantias mais concretas de qualidade de serviço, contribuindo para um aprimoramento na infra-estrutura da *Web*, dado o fato que a maioria das pesquisas de *QoS* enfocam garantias relativas.

Além da metodologia, o trabalho contribui ainda com um conjunto de métricas e parâmetros que possibilitam maior caracterização e estudo do impacto de provisão de *QoS* no escalonamento de requisições (tarefas) de um sistema. Esse ferramental permite uma análise mais abrangente da qualidade de serviço e oferece um mecanismo de controle dinâmico de satisfação (pela variação contratual). Por meio da relação entre taxa de utilização e exigência contratada para uma classe, como estabelecida pela equação (4.16), a metodologia oferece ainda a possibilidade de uma menor dependência do controle de admissão para controle da garantia da qualidade dos serviços oferecidos pelo sistema.

O trabalho apresentou ainda um estudo comparativo entre políticas de escalonamento consagradas em outras áreas, destacando os pontos positivos e negativos observados quando aplicadas em mecanismos de controle para provisão de *QoS* na *Web*. A implementação de tais políticas exigiu a alteração da biblioteca de simulação *SimpackJ*, para adequar os parâmetros de *QoS* a serem oferecidos, contribuindo assim para o melhoramento do pacote.

6.3 Trabalhos Futuros

A abordagem de escalonamento e o conjunto de métricas e parâmetros abordados neste projeto de mestrado consistem em um estudo inicial, podendo, portanto, serem ampliados. Por isso, são apresentadas, a seguir, sugestões para trabalhos futuros:

- Estudo e implementação da política em um ambiente distribuído, verificando a viabilidade e os benefícios de estender o gerenciamento da exigência imposta a um *cluster* de servidores *Web*, visando oferecer menores médias de latência de sistema aos usuários dos serviços.

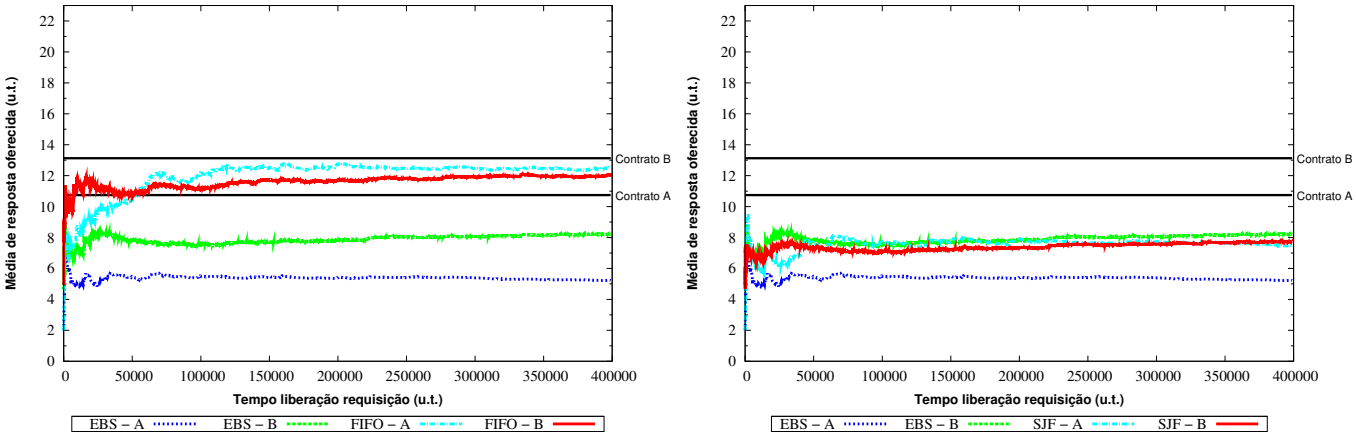
- Implementação e análise da viabilidade do "sistema de promoção" de classes para tratar contornar cenários de baixo desempenho do sistema, trabalhando com a alteração contratual como meio de prover melhores índices de satisfação aos usuários.
- Propor alternativas para a política proposta, verificando, por exemplo, se é melhor descartar requisições cujo valor de *deadline* tenha sido muito descumprido, em vez de mantê-la em fila com exigência cada vez mais crescente. Talvez computar esse atraso de atendimento na média de latência do cliente e esperar ele refazer a requisição pode impor uma menor exigência ao sistema, possibilitando um melhor atendimento quando a requisição for submetida novamente.
- Inclusão de um mecanismo para controle da variância das médias de latência oferecidas, objetivando garantias um pouco mais rígidas de qualidade de serviço.
- Estudar a viabilidade de se implementar um módulo de controle de admissão para oferecer melhores níveis de confiabilidade dos requisitos de *QoS* das diversas classes do sistema.

APÊNDICE

A

Demais Gráficos

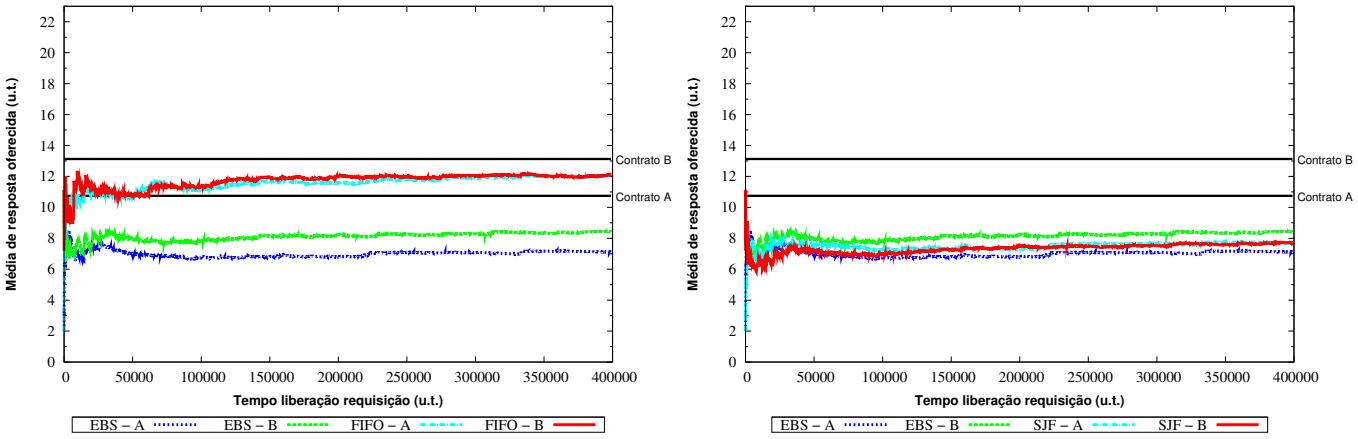
10% Variação Contratual



(a) FIFO X EBS

(b) SJF X EBS

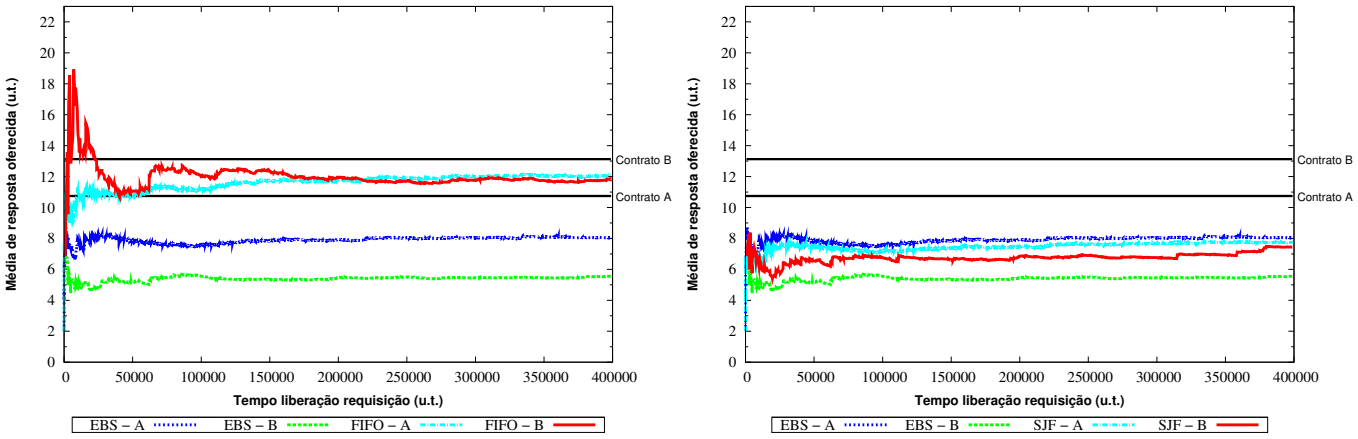
Figura A.1: ***Comparativo entre escalonadores - Cenário 10% A e 90% B com 10% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura A.2: ***Comparativo entre escalonadores - Cenário 50% A e 50% B com 10% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura A.3: ***Comparativo entre escalonadores - Cenário 90% A e 10% B com 10% de variação contratual***

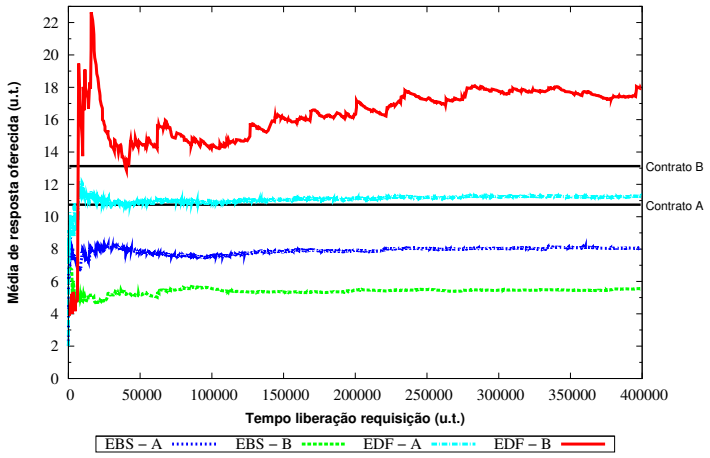
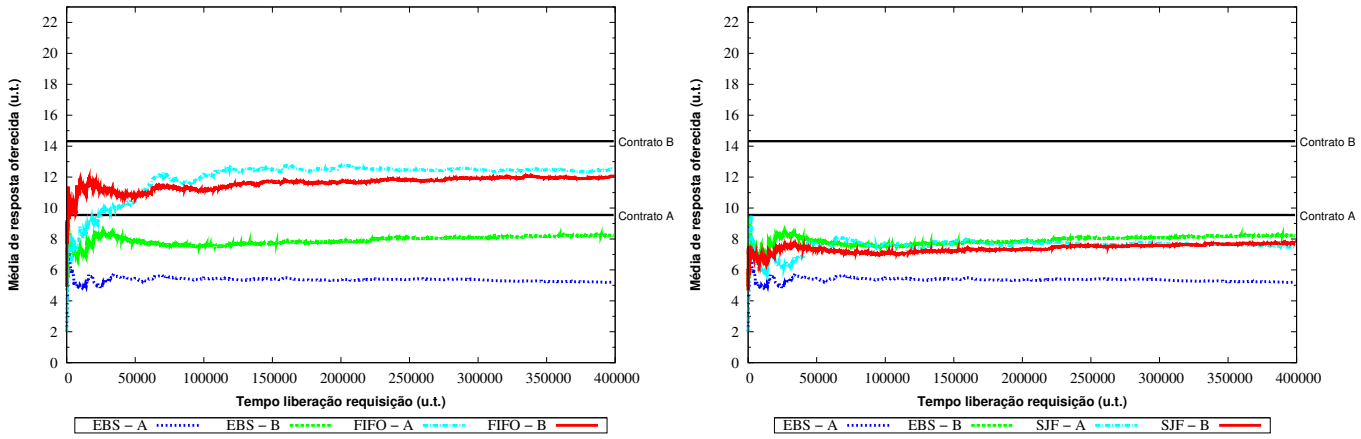


Figura A.4: ***Comparativo entre EBS e EDF - Cenário 90% A e 10% B com 10% de variação contratual***

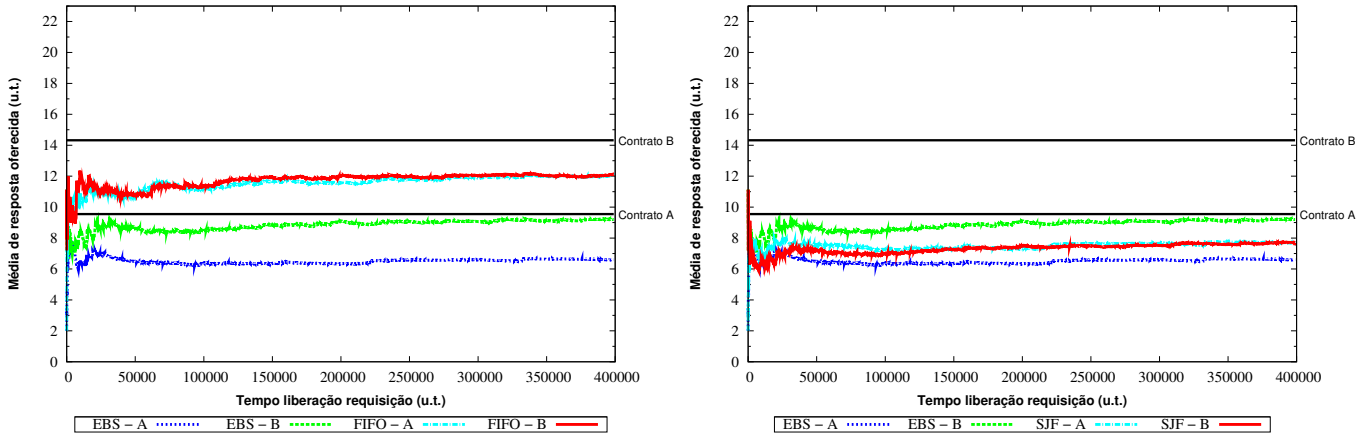
20% Variação Contratual



(a) FIFO X EBS

(b) SJF X EBS

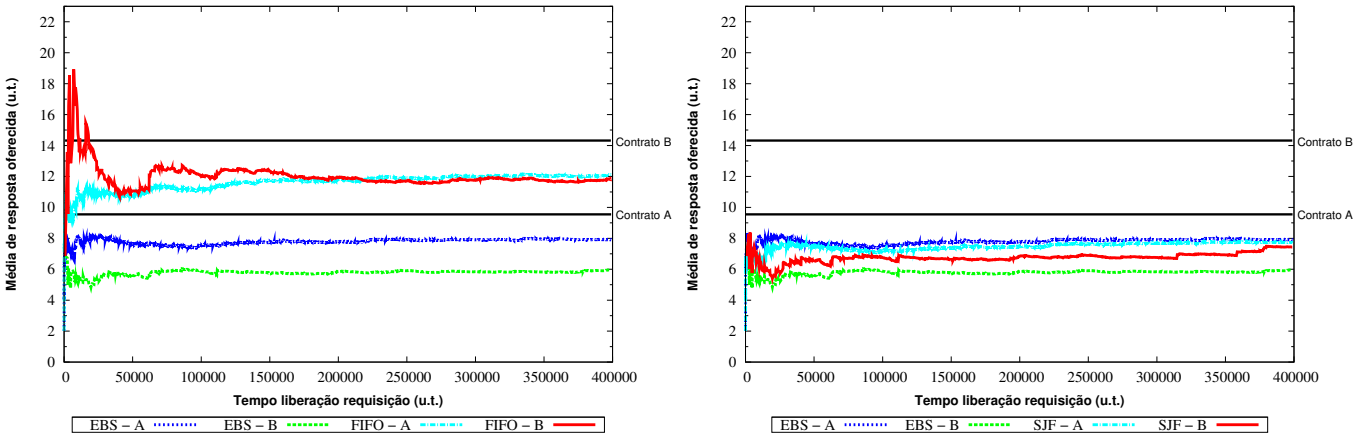
Figura A.5: ***Comparativo entre escalonadores - Cenário 10% A e 90% B com 20% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura A.6: ***Comparativo entre escalonadores - Cenário 50% A e 50% B com 20% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura A.7: ***Comparativo entre escalonadores - Cenário 90% A e 10% B com 20% de variação contratual***

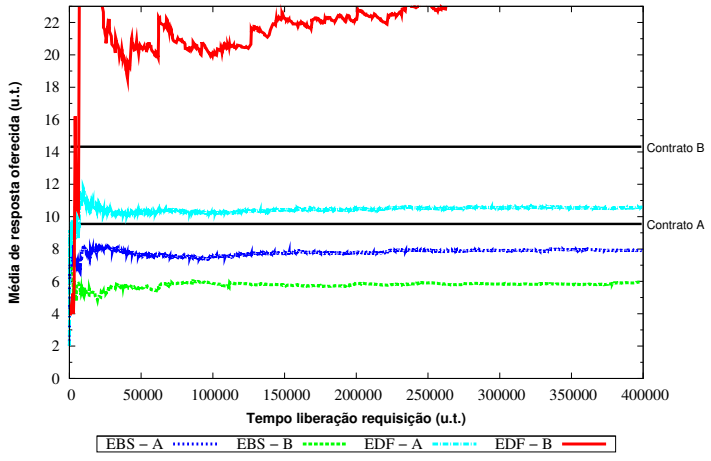
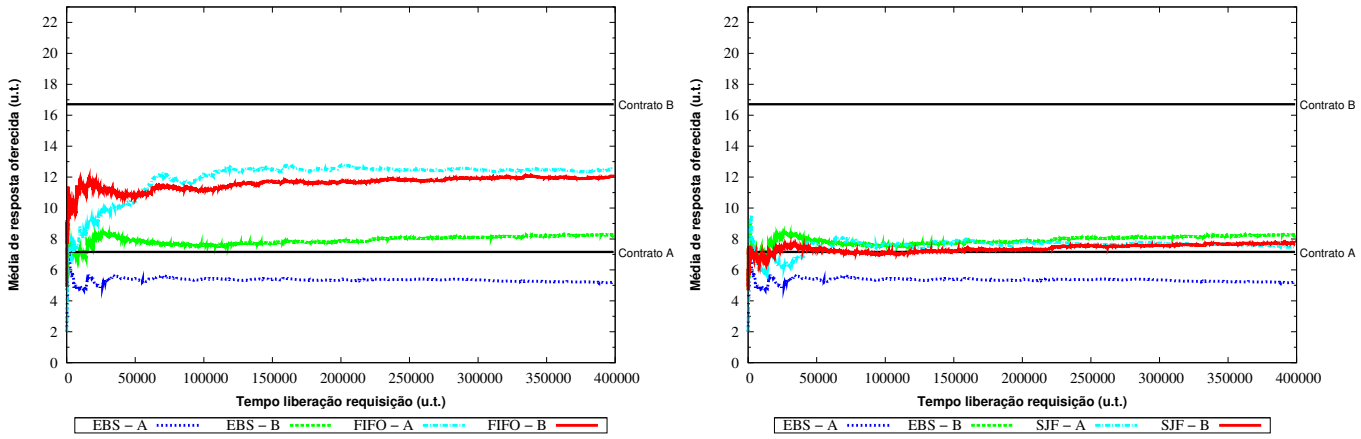


Figura A.8: ***Comparativo entre EBS e EDF - Cenário 90% A e 10% B com 20% de variação contratual***

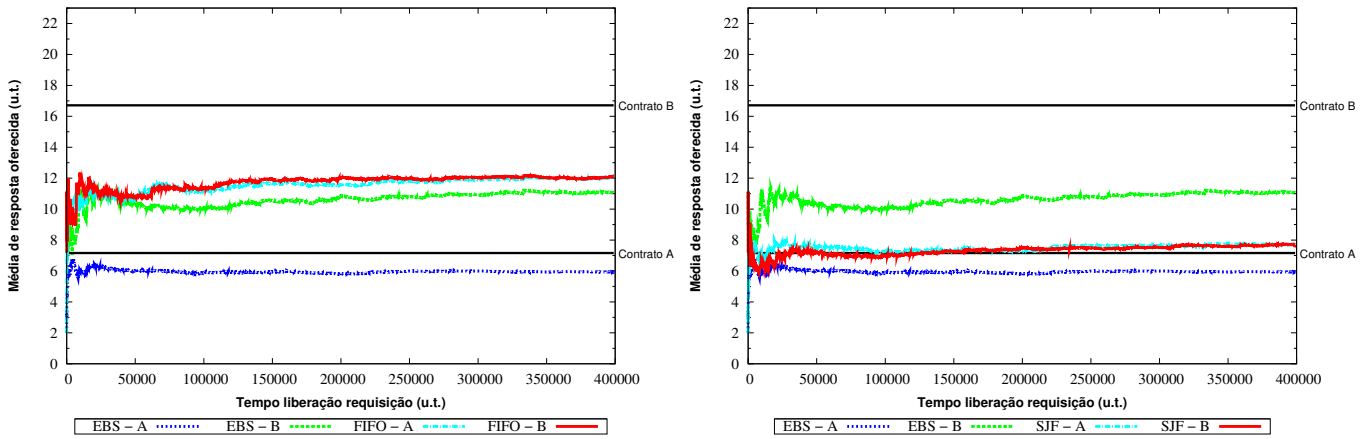
40% Variação Contratual



(a) FIFO X EBS

(b) SJF X EBS

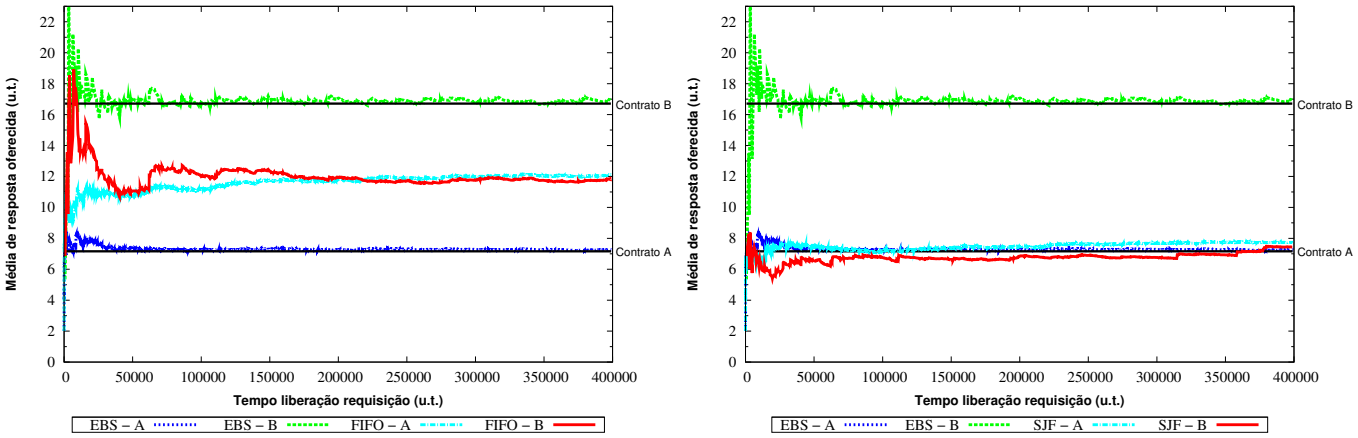
Figura A.9: ***Comparativo entre escalonadores - Cenário 10% A e 90% B com 40% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura A.10: ***Comparativo entre escalonadores - Cenário 50% A e 50% B com 40% de variação contratual***



(a) FIFO X EBS

(b) SJF X EBS

Figura A.11: ***Comparativo entre escalonadores - Cenário 90% A e 10% B com 40% de variação contratual***

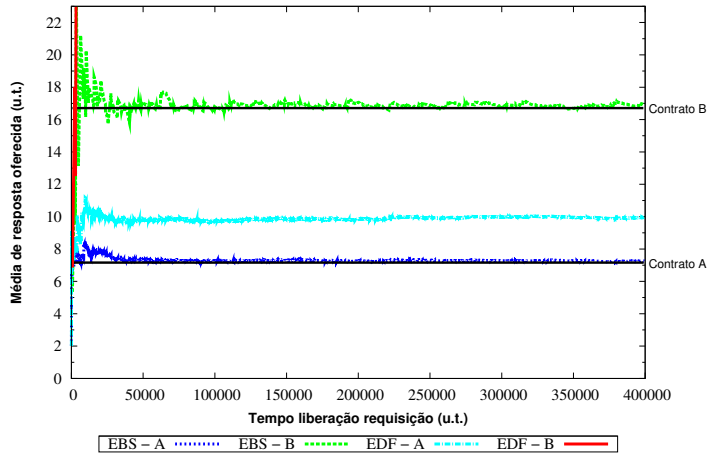


Figura A.12: ***Comparativo entre EBS e EDF - Cenário 90% A e 10% B com 40% de variação contratual***

Referências Bibliográficas

Banks, J.; Carson, J. S.; Nelson, B. L. (2000). *Discrete-Event System Simulation*. Prentice-Hall, Upper Saddle River, New Jersey 07458, 3 edição.

Barbato, A. K.; Traldi, O.; Santana, R. H. C.; Santana, M. J.; Teixeira, M. M. (2006). Algoritmo de escalonamento para servidores web baseado em sessão. *XII Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)*. Natal, RN.

Bhattacharyya, G. K. (1977). *Statistical concepts and methods*. New York : Wiley.

Bhatti, N.; Bouch, A.; Kuchinsky, A. (2000). Integrating user-perceived quality into web server design. *Computer Networks*, v. 33 de *1–6*, p. 1–16.

Birtwhistle, G. M.; Dahl, O.-J.; Myhrhaug, B.; Nygaard, K. (1973). Simula begin. Academic Press.

Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. (1998). Rfc 2475: An architecture for differentiated services.

Braden, E. R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S. (1997). Rfc 2205: Resource reservation protocol (rsvp).

Braden, R.; Clark, D.; Shenker, S. (1994). Rfc 1633: Integrated services in the internet architecture: an overview.

Chen, X.; Heidemann, J. (2003). Preferential treatment for short flows to reduce web latency. *Computer Networks*, v.41, n.6, p.779–794.

Cheng, A. M. K. (2002). *Real-Time Systems: Scheduling, Analysis, and Verification*. Wiley - Interscience, Hoboken, New Jersey.

Cherkasova, L. (1998). Scheduling strategy to improve response time for web applications. *HPCN Europe*, n. Vol. 1401, p. 305–314. Springer-Verlag.

Cherkasova, L.; Phaal, P. (2002). Session-based admission control: A mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers*, v.51, n.6, p.669–685.

Eggert, L.; Heidemann, J. (1999). Application-level differentiated services for web servers. *World Wide Web Jounal*, v.v. 2, n.3, p.133–142.

Elnikety, S.; Nahum, E.; Tracey, J.; Zwaenepoel, W. (2004). A method for transparent admission control and request scheduling in e-commerce web sites. *WWW '04: Proceedings of the 13th international conference on World Wide Web*, p. 276–286, New York, NY, USA. ACM Press.

Estrella, J.; Teixeira, M.; Santana, M.; Santana, R. H. C.; Bruschi, S. M. (2006). Negotiation mechanisms on application level: a new approach to improve quality of service in web servers. *Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. The Fourth IEEE Workshop on*, p. 5pp.

Estrella, J. C. (2005). Mecanismos de negociação no módulo de controle de admissão da arquitetura de servidor web com diferenciação de serviços (swds). Dissertação (Mestrado), Universidade de São Paulo.

Fishwick, P. A. (1992). Simpack: Getting started with simulation programming in c and c++. *Winter Simulation Conference*, p. 154–162.

Fishwick, P. A. (2005). Simpackj: Simpack toolkit. Acessado em Nov. de 2005.

Harchol-Balter, M.; Bansal, N.; Schroeder, B. (2000). Implementation of srpt scheduling in web servers. Technical report, Carnegie Mellon University, Pittsburgh, PA 15213.

Heidemann, J.; Shenker, S.; Dupuy, A.; Helmy, A.; Kumar, S. (2007). The network simulator - ns-2. Acessado em Maio de 2007.

Jain, R. (1991). *The art of computer systems performance analysis : techniques for experimental design, measurement, simulation, and modeling*. New York : Wiley.

Kamienski, C.; Sadok, D.; Cavalcanti, D. A. T.; Souza, D. M. T.; Dias, K. L. (2002). Simulando a internet: Aplicações na pesquisa e no ensino. *21ª JAI (Jornada de Atualização em Informática)*, p. 97–138, Florianópolis/SC.

Kanodia, V.; Knightly, E. (2003). Ensuring latency targets in multiclass web servers. *Parallel and Distributed Systems, IEEE Transactions on*, v.14, n.1, p.84–93.

Kurose, J. F.; Ross, K. W. (2006). *Redes de Computadores e a Internet: Uma abordagem top-down*. Pearson Addison Wesley, São Paulo, 3 edição.

Laplante, P. A. (2004). *Real-Time Systems Design and Analysis*. Wiley-IEEE Press, Hoboken, New Jersey, 3. edição.

Law, A. M.; Kelton, W. (2006). *Simulation Modeling and Analysis*. McGraw-Hill Education, USA, 4 edição.

Lazowska, E. D.; Zahorjan, J.; Graham, G. S.; Sevcik, K. C. (1984). *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Upper Saddle River, NJ, USA.

Liu, J. W.-S. (2000). *Real-time Systems*. Prentice-Hall, Upper Saddle River, New Jersey 07458, 1 edição.

Lu, Y.; Abdelzaher, T.; Lu, C.; Sha, L.; Liu, X. (2003). Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers. *Real-Time and Embedded Technology and Applications Symposium, 2003. Proceedings. The 9th IEEE*, p. 208–217.

MacDougall, M. H. (1989). *Simulating Computer Systems*. Computer Systems. MIT Press, Cambridge, Massachusetts London, England, 2 edição.

MacDougall, M. H.; McAlpine, J. S. (1973). Computer simulation with ASPOL. *Symposium on the simulation of computer systems*.

Meyer, P. L. (2003). *Probabilidade : aplicações à estatística*. LTC, 2ª edição. Tradução de Ruy de C. B. Lourenço Filho.

Nissanke, N. (1997). *Realtime Systems*. Prentice-Hall, Upper Saddle River, New Jersey 07458, 1 edição.

Santos, A. P. S. (1999). Qualidade de serviço na internet. Boletim bimestral sobre tecnologia de redes da RNP - Rede Nacional de Ensino e Pesquisa, v. 3, n. 6,.

Schwetman, H. (1986). Csim: A c-based, process-oriented simulation language. *Winter Simulation Conference*, p. 387 – 396.

Sha, L.; Abdelzaher, T.; Arzén, K.-E.; Cervin, A.; Baker, T.; Burns, A.; Buttazzo, G.; Caccamo, M.; Lehoczky, J.; Mok, A. K. (2004). Real time scheduling theory: A historical perspective. *Real-Time Syst.*, v.28, n.2-3, p.101–155.

Sha, L.; Liu, X.; Lu, Y.; Abdelzaher, T. (2002). Queueing model based network server performance control. *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, p. 81–90.

Soares, L. F. G. (1990). *Modelagem e simulacao discreta de sistemas*. Escola de Computacao 7. Ime-Usp, São Paulo.

Stardust (1999). Qos protocos & architectures. White paper, Stardust.com. Acessado em Maio 2007.

Strnadl, C. (2002). At your service: Qos for the internet. *IEEE Multimidia*, v. 09 de *Media Reviews*, p. 93–95.

Teixeira, M. A. M. (2004). Suporte a serviços diferenciados em servidores web: modelos e algoritmos. Dissertação (Mestrado), Universidade de São Paulo.

Teixeira, M. A. M.; Santana, M. J.; Santana, R. H. C. (2005). Servidor web com diferenciação de serviços: Fornecendo qos para os serviços da internet. In XXIII Simpósio Brasileiro de Redes de Computadores (SBRC), Fortaleza, CE.

Traldi, O. A.; Barbato, A. K.; Santana, R. H. C. (2006). Service differentiating algorithms for qos-enabled web servers. *WebMedia '06: Proceedings of the 12th Brazilian symposium on Multimedia and the web*, p. 263–272, Natal, Rio Grande do Norte, Brazil. ACM Press.

Vasiliou, N. (2000). Overview of internet qos and web server qos. Department of Computer Science, The University of Western Ontario, London, Ontario, Canada.

Xiao, X.; Ni, L. (1999). Internet qos: a big picture. *Network, IEEE*, v.13, n.2, p.8–18.

Ye, N.; Gel, E. S.; Li, X.; Farley, T.; Lai, Y.-C. (2005). Web server qos models: applying scheduling rules from production planning. *Computers & Operations Research*, v.32, p.1147–1164.

Zhao, W.; Olshefski, D.; Schulzrinne, H. (2000). Internet quality of service: an overview. Technical Report CUCS-003-00. Acessado em Maio de 2007.