Políticas de atendimento para servidores Web com serviços diferenciados baseadas nas características das requisições

Ottone Alexandre Traldi

Políticas de atendimento para servidores Web com serviços diferenciados baseadas nas características das requisições

Ottone Alexandre Traldi

Orientador: *Prof. Dr. Marcos José Santana*

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

USP – São Carlos
Novembro de 2008

* Agradecimento ao CNPq pelo apoio parcial dado a este trabalho.

Dedico este trabalho a minha família e amigos, que sempre estiveram presentes, dando-me a força e incentivo necessários para concluí-lo com sucesso.

Agradecimentos

Agradeço a Deus pela vida, saúde e inspiração necessários para a realização deste trabalho.

Agradeço aos meus pais e irmã pela motivação e apoio ao longo de toda a minha vida acadêmica.

Agradeço ao Buby por ter me ajudado a entender muito cedo o real valor da vida.

Agradeço ao Benedicto Geraldo Stoppa, Maria Aparecida Rosa Vianna, Fabio Vianna Stoppa e Aparecida Poiatti Rosa Vianna por terem atuado como uma segunda família.

Agradeço ao Professor Doutor Marcos José Santana pela orientação científica desde os tempos de graduação e à Professora Doutora Regina Helena Carlucci Santana pela co-orientação. Com ambos construí uma amizade que atravessa os muros da Universidade, amizade que foi determinante em diversas conquistas pessoais.

Agradeço aos professores que participaram do meu exame de qualificação e da minha defesa, pela leitura respeitosa dos meus textos, que contribuiu para o aprimoramento da pesquisa.

Agradeço à amiga Alessandra Kelli Barbato, com quem mantive uma parceria acadêmica da graduação ao Mestrado.

Agradeço ao amigo Marcelo Fila Pecenin, com quem mantenho uma parceria no esporte e na vida há mais de 10 anos.

Agradeço à Marjorie Mariano pelo apoio que chegou já no fim deste trabalho, mas de extrema importância.

Agradeço aos colegas e amigos com quem mantive contato durante o período do Mestrado e que, de alguma forma, contribuíram com a realização deste trabalho: José Mário Franzin, Vilma Franzin, Rafael Belli, Mateus Pecenin, Umberto Pecenin, Fátima Pecenin, Luiz Henrique Folster, Tiago Poiatti, Rosa Belli, José Armando Belli, Gustavo Belli, Adilson Renesto, Maria Ângela Renesto, Rodrigo Tortella, Gustavo Lazarini, Felipe Francisco, Murilo Francisco, Felipe Ferreira, Victor Molina, Maria Fernanda de Carvalho, Diogenes Medeiros Júnior, Michel Lacombe, Luiz Fernandes e Marlene Fernandes; Matheus Santos, Lucas Casagrande, Julio Estrella, Geraldo, Valter e demais colegas do LaSDPC; Renato Pimentel, Cibele Russo, Daniel Silva, Daniel Junqueira, Marcos Martins, Márcia Sena, Felipe Ladeira, André Freire e demais amigos da USP; Anne Carraretto, Leonardo Martins, Felipe Ricci e demais amigos do Unibanco; e à toda “molecadinha” do CERD.

Resumo

Este trabalho propõe mecanismos de diferenciação de serviços para servidores Web, visando a melhorar o desempenho desses sistemas quando são consideradas as características das requisições Web nas políticas de atendimento. Optou-se por adotar o contexto do comércio eletrônico para a realização das pesquisas, uma vez que esse ambiente é um dos mais impactados negativamente quando há um comportamento inadequado do servidor em situações de sobrecarga. Para isso, foi realizada uma investigação das características das requisições Web típicas do *e-commerce*, para que tais características pudessem ser usadas como diretrizes para os mecanismos e melhorar o desempenho dos servidores. Em seguida, foram propostos um modelo de carga de trabalho e um modelo de simulação para a realização dos experimentos. Com isso, foi possível avaliar os resultados obtidos com a inserção dos diversos mecanismos no Servidor Web com Diferenciação de Serviços (SWDS), um modelo de servidor cuja arquitetura o torna capaz de fornecer serviços diferenciados a seus usuários e aplicações. Foram propostos novos mecanismos de escalonamento de requisições bem como novos mecanismos de controle de admissão. Diversas simulações foram realizadas e os resultados obtidos mostram que a exploração das características das requisições Web, além de ser fundamental para um bom entendimento do comportamento do servidor, possibilita a melhoria de desempenho do sistema.

Abstract

This work proposes differentiated services mechanisms for Web servers, aiming at improving their performance when the features of Web requests are considered. The electronic commerce (e-commerce) context was adopted to develop the researches once this environment is one of the most negatively influenced when there is an inadequate behavior of the server under overload situations. Thus, it was realized an investigation on the features of e-commerce Web requests, so that these features could be used both as guidelines for the mechanisms and to improve the performance of the servers. Afterwards, a workload model and a simulation model were proposed to implement the experiments. Thus, it was possible to evaluate the results obtained with the insertion of several mechanisms in the Web Server with Differentiated Services (WSDS), a server model with an architecture that makes it capable of supplying differentiated services to its users and applications. New request scheduling mechanisms were proposed as well as new mechanisms for admission control. Several simulations were realized and the obtained results show that the exploration of the Web request features, besides being fundamental for a good understanding of the server behavior, makes possible to improve the system performance.

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

CA	Controle de Admissão
CGI	*Common Gateway Interface*
CPU	*Central Processing Unit*
DNS	*Domain Name Service*
FCFS	*First Come, First Served*
FTP	*File Transfer Protocol*
GPL	*General Public License*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
OSI	*Open Systems Interconnection*
PC	*Personal Computer*
QoS	*Quality of Service*
QoWS	*Quality of Web Services*
RFC	*Request for Comments*
RR	*Round-Robin*
RSVAdap	Reserva Adaptativa de Recursos
RSVP	*Resource Reservation Protocol*
SLA	*Service Level Agreement*
SMNP	*Simple Network Management Protocol*
SMTP	*Simple Mail Transfer Protocol*
SSL	*Secure Socket Layer*
SWDS	Servidor Web com Diferenciação de Serviços
TCP	T*ransmission Control Protocol*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locators*
W3C	*World Wide Web Consortium*
WFQ	*Weighted Fair Queuing*
WWW	*World Wide Web*

Sumário

Capítulo

1

Introdução

1.1 Contextualização

A Internet é hoje um dos mais importantes meios de comunicação, entretanto, não foi projetada para o uso atualmente observado.

Inicialmente, os dados transferidos pela Internet ofereciam pouca carga, porém, com o surgimento da *World Wide Web* (WWW), foi observado um grande aumento do tráfego na rede, pois ela passou a existir não apenas para fins de pesquisas, mas também para ser utilizada para fins informativos, educacionais, de entretenimento e comerciais. Portanto, ocorreram mudanças não apenas na quantidade do tráfego da Internet, mas também na natureza do mesmo (Xiao & Ni, 1999). Essa rede se tornou uma plataforma para aplicações de conteúdo dinâmico, de comércio eletrônico, telefonia, entre outras. No entanto, essas aplicações não requerem o mesmo tratamento, uma vez que não possuem as mesmas características ou mesmas prioridades. Contudo, o serviço oferecido pela Internet baseia-se em um modelo de melhor esforço (*best-effort*), em que a rede procura transportar os dados que lhe são confiados no menor tempo possível, de preferência sem erros ou inconsistências, mas sem dar às aplicações garantias de que isso realmente ocorrerá (Comer, 2000).

Uma das conseqüências da adoção desse modelo de melhor esforço é o fato de que todo o

tráfego é tratado de maneira uniforme, sem nenhum tipo de diferenciação ou de priorização. Assim, quando há congestionamento, os pacotes presentes na rede são descartados sem distinção, independentemente das necessidades das aplicações.

1.2 Motivação

Como as aplicações não possuem as mesmas características ou mesmas prioridades, é essencial que a Internet forneça diferenciação de serviços com diferentes níveis de QoS (*Quality of Service*) para os diversos tipos de requisições (Kant & Mohapatra, 2000).

Uma das soluções propostas atualmente pela comunidade científica baseia-se no aumento sob demanda da largura de banda da rede, buscando minimizar o descarte de pacotes. Outra abordagem adotada para a solução do problema é a inserção de QoS na Internet por meio do fornecimento de serviços diferenciados, ou seja, o tráfego não deve ser tratado de maneira uniforme, sem nenhum tipo de diferenciação ou de priorização. A inserção de QoS na Internet por meio dos Serviços Diferenciados tem se mostrado uma excelente alternativa ao modelo de melhor esforço em que o serviço oferecido pela rede se baseia.

Não é difícil perceber que qualquer esforço para o fornecimento de QoS na Web não poderá ter sucesso se apenas mecanismos em nível de rede e sistema operacional forem utilizados, pois, em última instância, são os servidores Web os responsáveis pelo atendimento das solicitações dos usuários. Dessa forma, é necessário que os servidores Web também sejam capazes de prover QoS, e, para isso, eles devem possuir políticas de atendimento que consigam realizar a diferenciação de serviços.

Estão disponíveis diversas especificações para a provisão de QoS sobre redes IP, com destaque para as arquiteturas de Serviços Integrados (*IntServ*) (Braden et al., 1994) e Diferenciados (*DiffServ*) (Blake et al., 1998). Em nível de aplicação, a utilização de Serviços Diferenciados tem sido proposta como uma solução eficiente para a provisão de melhores

serviços. Isso pode ser observado por meio dos vários trabalhos desenvolvidos na área, entre eles o de Cardellini et al. (2001), em que é introduzido o conceito de *Quality of Web Services (QoWS),* inspirado nos princípios de garantia de QoS em nível de rede, definidos por Kurose & Ross (2006).

Em Teixeira 2004, é proposta uma arquitetura para um servidor Web que o torne capaz de fornecer serviços diferenciados a seus usuários e aplicações, de acordo com suas características de demanda. Esse modelo foi denominado Servidor Web com Diferenciação de Serviços (SWDS).

Foram incluídos alguns algoritmos de escalonamento no modelo SWDS visando tanto ao balanceamento de carga entre os processos quanto à diferenciação de serviços. No primeiro caso, destacam-se os algoritmos *Round Robin* e *Shortest Queue First* (Teixeira, 2004). No segundo caso, destacam-se os algoritmos de Reserva de Recursos (Teixeira, 2004), Reserva Adaptativa de Recursos (RSVAdap) (Traldi et al., 2006), Baseados em Prioridades (Estrito e Adaptativo) (Teixeira et al., 2005) e *Weighted Fair Queuing* (WFQ) (Traldi et al., 2006).

Também foram incluídas, no modelo SWDS, algumas políticas de controle de admissão, conforme Teixeira (2004). Essas políticas, levando em consideração a carga do sistema e a classe a que pertence uma determinada requisição que chega ao servidor, decidem pela aceitação ou pela rejeição da solicitação.

Os resultados fornecidos pelas políticas inseridas no modelo SWDS são promissores e motivam a investigação de novas políticas de controle de serviços. Entretanto, tanto as políticas de escalonamento quanto as políticas de controle de admissão implantadas no modelo não levam em consideração as características das requisições Web como parâmetro na tomada de decisões. Essas políticas baseiam-se apenas na classe da tarefa, que é definida pela sua prioridade.

1.3 Objetivos

O objetivo central da pesquisa desenvolvida nesta dissertação de Mestrado foi avaliar a possibilidade de melhorar o desempenho do modelo de Servidor Web com Diferenciação de Serviços (SWDS), quando são consideradas as características das requisições Web nas políticas de atendimento. Entre os objetivos específicos, destacam-se:

- Investigação das características das requisições Web;
- Realização de um detalhamento do modelo SWDS considerando-se os recursos necessários para a execução das requisições Web;
- Desenvolvimento de novas políticas de escalonamento que considerem o tipo das requisições Web, e/ou alteração das já existentes;
- Desenvolvimento de novas políticas de controle de admissão que considerem os tipos das requisições Web, e/ou alteração das já existentes;
- Inserção das políticas desenvolvidas em um modelo de simulação do Servidor Web com Diferenciação de Serviços (SWDS).

1.4 Estrutura

No Capítulo 2, será apresentada a infra-estrutura da Internet, seus protocolos e principais serviços, com destaque para a Web.

No Capítulo 3, será abordado o tópico de diferenciação de serviços na Internet, e serão discutidas as limitações do seu modelo atual e como a inserção de qualidade de serviço pode tornar essa rede mais eficiente. A abordagem de serviços diferenciados sobre redes IP e a viabilidade de seu emprego em nível de aplicação, particularmente em servidores Web, serão destacados. Em seguida, será apresentado o modelo para um Servidor Web com

Diferenciação de Serviços, o SWDS. Será descrita a sua arquitetura e também serão discutidos alguns algoritmos de controle de admissão e diferenciação de serviços.

No Capítulo 4, serão apresentadas algumas das técnicas de Avaliação de Desempenho encontradas na literatura e mostrada a metodologia adotada neste trabalho, uma vez que é necessária a utilização de algumas dessas técnicas para avaliar o desempenho do modelo SWDS com a inserção de novas estratégias de atendimento.

No capítulo 5, serão apresentados os critérios utilizados para a geração da carga de trabalho sintética utilizada para alimentar o modelo do SWDS descrito no capítulo anterior.

No capítulo 6, serão apresentadas as estratégias propostas para melhorar o desempenho do SWDS quando são consideradas as características das requisições Web na tomada de decisão e os resultados obtidos.

Finalmente, no capítulo 7, serão apresentadas as conclusões e propostas de trabalhos futuros.

Capítulo

2

Infra-estrutura da Web

2.1 Considerações Iniciais

Neste capítulo, é dada uma visão geral da infra-estrutura da Internet e da *World Wide Web*, sendo discutidas suas características e apresentados seus principais protocolos. Dada a importância desse elemento na rede, também é realizada uma descrição das formas de organização e funcionamento dos servidores Web.

2.2 A Internet

A Internet pública é uma rede de computadores mundial, isto é, uma rede que conecta milhões de equipamentos de computação em todo o mundo. A maior parte desses equipamentos é formada por PCs tradicionais e pelos chamados servidores, que armazenam e transmitem informações, como páginas Web e mensagens por e-mail (Koruse & Ross, 2006). Cada vez mais, equipamentos de computação estão sendo conectados à Internet. Esses equipamentos são chamados de *hospedeiros* ou *sistemas finais.*

Os sistemas finais, assim como a maioria dos outros componentes da Internet, operam protocolos que controlam o envio e o recebimento de informações na Internet. O TCP

(T*ransmission Control Protocol*) e o IP (*Internet Protocol*) são os protocolos mais importantes da Internet. Eles são conhecidos coletivamente como TCP/IP (Kurose & Ross, 2006).

2.2.1 Protocolos TCP/IP

O protocolo TCP/IP foi elaborado antes dos PCs, antes da proliferação das *Ethernets* e de outras tecnologias de redes locais e, portanto, antes da Web. Esse protocolo surgiu da necessidade de um protocolo de rede que fornecesse amplo suporte para aplicações ainda a serem definidas e permitisse, ao mesmo tempo, a interoperação de hospedeiros arbitrários e protocolos de camada de enlace (Kurose & Ross, 2006). O TCP/IP tornou-se um padrão para a interligação de computadores, tanto em redes locais quanto em longa distância.

O TCP/IP usa a comutação de pacotes para realizar a comunicação entre *hosts* e pode ser empregado sobre diferentes protocolos das camadas de enlace e física, de forma transparente para as aplicações. Outra característica importante é que o TCP/IP faz a entrega de dados de forma mais confiável, utilizando para isso um mecanismo de *acknowledgements* entre a origem e o destino final de uma comunicação.

As especificações dos protocolos TCP/IP estão livremente disponíveis na Internet, por meio de RFCs (*Request for Comments*), promulgadas pela IETF (*Internet Engineering Task Force*) (Postel, 1981c).

2.2.2 A Arquitetura TCP/IP

A arquitetura TCP/IP visualiza a rede de computadores em quatro camadas: acesso à rede, Internet, transporte e aplicação. Esse modelo de organização é equivalente às camadas do modelo OSI (*Reference Model of Open Systems Interconnection*), que propõe uma organização de arquitetura em sete camadas: aplicação, apresentação, sessão, transporte, rede,

enlace de dados e física (Tanenbaum, 2002). Essa equivalência é ilustrada pela Figura 2.1:

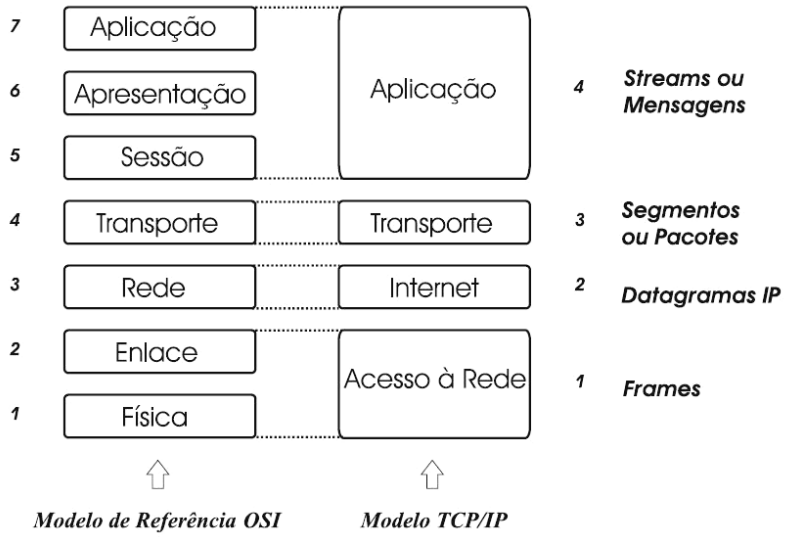

Figura 2.1: As Camadas da Arquitetura TCP/IP

Camada de Acesso à Rede

A Camada de Acesso à Rede, com o menor nível de abstração na arquitetura TCP/IP, usa padrões para conexão à rede física. Ela utiliza vários protocolos de acesso à rede conforme o tipo de rede disponível, dando flexibilidade aos protocolos TCP/IP para que funcionem em diversos meios físicos.

Essa camada também faz o encapsulamento dos datagramas IPs nos *frames* usados na rede, e converte os endereços IP para o formato da rede.

Camada de Internet

Nessa camada está estabelecido o protocolo IP (*Internet Protocol*) (Postel, 1981b), para transporte não-confiável de mensagens, e o protocolo ICMP (*Internet Control Message Protocol*) (Postel, 1981a), para controle da comunicação e informe de erros.

O protocolo IP provê um serviço de rede não orientado à conexão – não requerendo troca de informações de controle iniciais (*handshaking*) para que uma comunicação seja iniciada –

e providencia o roteamento da mensagem de um dispositivo para outro, em uma rede local ou uma de longa distância. As informações são enviadas dentro de unidades de transferência de dados chamadas datagramas. Cada datagrama contém informações necessárias para que possa ser roteado de forma independente. O roteamento dos datagramas entre os *hosts* e a fragmentação e remontagem dos datagramas ao passarem de uma camada para outra da pilha TCP/IP são funções do protocolo IP.

O IP também é responsável pelo endereçamento da rede. Cada *host* conectado à Internet possui um endereço distinto que é utilizado pelos equipamentos da rede para encontrar o caminho entre origem e destino.

Em suma, a camada de Internet permite a comunicação entre duas máquinas da rede.

Camada de Transporte

A Camada de Transporte permite a comunicação fim-a-fim entre dois *hosts* ou, mais especificamente, entre processos executados em *hosts* conectados à Internet. Essa camada introduz o conceito de porta, um nível de endereço adicional que identifica a aplicação na máquina. Um endereço completo nesse nível é estabelecido por um par (*host*, *port*).

Nessa camada, estão definidos os protocolos TCP (*Transmission Control Protocol*) (Postel, 1981c) e UDP (*User Datagram Protocol*) (Postel, 1980).

O protocolo TCP fornece um serviço confiável, garantindo que os dados cheguem sem erros ao seu destino. É um protocolo orientado à conexão, ou seja, uma conexão lógica fim-a-fim é estabelecida entre os dois *hosts* antes de a comunicação ser iniciada. O TCP também realiza controle de fluxo para suas aplicações e controle de congestionamento. O controle de fluxo procura eliminar a possibilidade de o remetente saturar o destinatário com o envio de dados. O controle de congestionamento tenta regular os remetentes quando a rede encontra-se congestionada.

O protocolo UDP é não orientado à conexão e oferece um serviço não-confiável. O UDP é indicado para a construção de aplicações cliente-servidor, principalmente, em redes locais (Coulouris et al., 2000). Mensagens DNS (*Domain Name Service*), SNMP (*Simple Network Management Protocol*) e algumas aplicações de transmissão de áudio e vídeo utilizam o protocolo UDP.

Camada de Aplicação

Essa camada, de maior nível de abstração da pilha TCP/IP, define o conjunto de serviços manipulados por usuários. Nela estão os processos que utilizam os serviços da camada de transporte para a entrega dos dados, podendo escolher entre os serviços oferecidos pelo protocolo TCP ou protocolo UDP. É nessa camada que são encontrados os serviços de TELNET, FTP, DNS, HTTP, SMTP, entre outros.

2.3 A Organização da Web

O funcionamento da *World Wide Web*, iniciado na década de 1990, levou a Internet para os lares e empresas de milhões de pessoas em todo o mundo. Ela serviu como plataforma para a habilitação e disponibilização de centenas de novas aplicações. Com isso, gerou um grande aumento no tráfego da Internet, superando serviços como FTP e e-mail. Com toda essa popularidade, muitos usuários pensam que a Web é a própria Internet.

A Web é considerada um sistema de hipertexto em escala global e uma grande plataforma cliente-servidor. Por meio de *browsers* ou navegadores, os usuários acessam as informações que são armazenadas em servidores Web ou HTTP espalhados por todos os continentes (Orfali et al., 1999).

A linguagem HTML e o protocolo HTTP são dois padrões sobre os quais a Web funciona, garantindo a sua portabilidade. A HTML (W3C, 1999a; W3C, 2000) é o idioma universal

falado na Web. Suas *tags* descrevem a estrutura do documento, fornecem informações sobre sua formatação e estabelecem os *links* com outros documentos ou outros recursos da Web. O protocolo HTTP é utilizado para a comunicação entre os *browsers* e os servidores Web e funciona sobre o TCP.

Uma característica importante na organização da Web é seu sistema de nomenclatura baseado em URLs (*Uniform Resource Locators*). Uma URL pode identificar páginas HTML, um recurso ou um objeto qualquer existente na Internet. Sua estrutura é composta por:

- *Protocolo:* informa qual é protocolo utilizado para o transporte dos dados;
- *Nome do servidor:* um endereço IP ou nome de um *host* válido;
- *Número da porta*: onde um processo aguarda mensagens, no servidor;
- *Localização do recurso: path* ou caminho até o recurso.

2.3.1 Interações Cliente-Servidor na Web

A Web é considerada como uma grande plataforma cliente-servidor. Um servidor recebe as requisições e, sendo consideradas válidas, interpreta e retorna os objetos requisitados ao cliente. Os objetos podem estar armazenados no servidor (Páginas Estáticas) ou podem ser gerados dinamicamente por um programa ou *script* invocado no servidor (Páginas Dinâmicas).

Páginas Estáticas

Páginas estáticas são páginas HTML fisicamente armazenadas nos servidores Web. Dessa forma, os clientes, por meio de *browsers*, acessam um servidor Web, que por sua vez devolve as páginas HTML para o usuário.

Segundo Casalicchio & Colajanni (2001), as Páginas Estáticas podem ser classificadas como estáticas leves e *disk-bound* (páginas estáticas geradas a partir de consultas ao banco de dados). Esse tipo de classificação leva em consideração o consumo de recursos do sistema causado pelas requisições HTTP. Em geral, é considerado o consumo de CPU e o número de operações de *I/O*.

Páginas Dinâmicas

O protocolo CGI (*Common Gateway Interface*), criado em 1995, trouxe uma maior interatividade para a Web, permitindo iniciar uma aplicação do lado servidor, a partir de um *browser* (Yeager & McGrath, 1996). No entanto, a independência de plataforma ainda é garantida, pois toda a comunicação entre o *browser* e o servidor Web continua ocorrendo no formato HTML.

O CGI recebe o pedido de execução de aplicação via formulários HTML, em que os parâmetros são digitados no *browser* pelo usuário, e, então, faz a transferência para o programa apropriado, localizado no lado servidor. Esse programa envia uma Página Dinâmica como resposta. O usuário não tem a percepção de que a página foi gerada dinamicamente, a partir de um processo iniciado sob o comando do servidor Web e apenas a recebe como se fosse uma página estática.

Segundo Casalicchio & Colajanni (2001), as requisições dinâmicas podem ser classificadas como dinâmicas leves, *CPU-bound* e *disk/CPU-bound*.

2.4 Servidores Web

O servidor Web é um elemento de grande importância na rede, sendo responsável pelo atendimento aos usuários e deve ser projetado de maneira a atender o maior número de requisições possíveis, uma vez que, quando sobrecarregado, pode se tornar o gargalo do

sistema.

Um servidor Web está sempre em um laço infinito, aguardando por requisições dos clientes. Os servidores atuais tratam as requisições que recebem segundo uma disciplina *FCFS (First Come, First Served)*, ou seja, é mantida uma única fila de espera em que cada requisição aguarda o momento de ser atendida, de acordo com sua ordem de chegada.

Embora diferentes esquemas de controle de concorrência possam ser implementados no servidor, visando a agilizar o atendimento das requisições, este ocorre em geral de maneira uniforme, sem considerar as particularidades e urgência de cada tipo de requisição (Teixeira, 2004). A seguir, algumas formas de organização dos servidores Web são apresentadas brevemente:

- Servidor Iterativo: trata as requisições pela ordem de chegada, sem nenhum tipo de concorrência;

- Processo por Requisição: existe um processo principal que fica aguardando as requisições. Ao chegar uma requisição, o processo principal cria uma cópia de si mesmo e passa a esse "filho" a responsabilidade de tratar a requisição;

- *Pool* de Processos: ao iniciar sua execução, o servidor cria um número mínimo de processos que ficam aguardando até que sejam convocados pelo processo *dispatcher* para atenderem as requisições dos clientes;

- *Threads* por Requisição: para cada requisição que chega é criada uma nova *thread* para tratá-la;

- *Pool* de *threads:* várias *threads* são criadas quando o servidor Web é iniciado. Elas ficam aguardando pela chegada de requisições.

2.5 Protocolo HTTP

O HTTP (*Hypertext Transfer Protocol*), o protocolo de camada de aplicação da Web, está implementado em dois programas: um programa cliente e um programa servidor. O programa cliente e o programa servidor "conversam" pela troca de mensagens HTTP. Isso permite que o cliente acesse os recursos armazenados no servidor Web. Todo o tráfego de informações entre os *browsers* e servidores Web é feito por mensagens HTTP. Os dados são representados conforme o padrão MIME (*Multi-Purpose Internet Mail Extensions*) (Borenstein, 1993).

O HTTP é um protocolo do tipo *stateless*, isto é, o servidor não guarda nenhuma informação em relação ao estado dos clientes e, se uma falha na execução da tarefa acontecer, o cliente deverá fornecer novamente todas as informações necessárias para que a transação possa ser realizada. Uma transação HTTP se desenrola segundo um mecanismo do tipo requisição/resposta (*request/reply*).

2.5.1 Mensagens de Requisição e Resposta

As mensagens de requisição e resposta são definidas pelo protocolo HTTP em um formato padrão. Uma requisição é formada tipicamente por:

- *Request line*: linha que informa a ação a ser executada no servidor em que se encontra o método invocado (HTTP), a localização do objeto no servidor e a versão do protocolo HTTP utilizada;
- *Request header*: uma ou mais linhas de cabeçalho contendo informações do cliente para informar ao servidor, por exemplo, os tipos de dados que ele é capaz de aceitar;
- Corpo da mensagem: opcional. É utilizado quando dados adicionais do cliente devem ser enviados ao servidor.

O protocolo HTTP determina que uma mensagem de resposta deve conter:

- *Response header*: nesse cabeçalho é encontrada a versão do protocolo e o código de status com o seu significado;
- *Request header*: composto de vários campos nos quais estão contidas informações das características do servidor e do objeto retornado ao cliente;
- Corpo da mensagem: está contido o objeto retornado ao cliente, em geral um documento HTML precedido por uma linha em branco.

Na Figura 2.2, é mostrado um exemplo de Requisição e Resposta, em que um cliente solicita (GET) o arquivo /doc/file.html do servidor www.nome.com, usando o protocolo HTTP/1.1. O campo *Accept*, no cabeçalho, informa que o cliente é capaz de receber textos em formato HTML (text/html) e imagens em formato JPEG (image/jpeg). O tipo de *browser* do cliente é mostrado pelo campo *User-Agent*.

A resposta do servidor diz que a requisição foi bem sucedida (indicada pelo código 200). O tipo do servidor Web é informado pelo campo *Server*, o campo *Content-Type* mostra que o objeto retornado é um documento HTML, e o campo *Content-Lenght* indica o número de bytes do objeto que está sendo enviado. Após a linha em branco, encontra-se o documento solicitado:

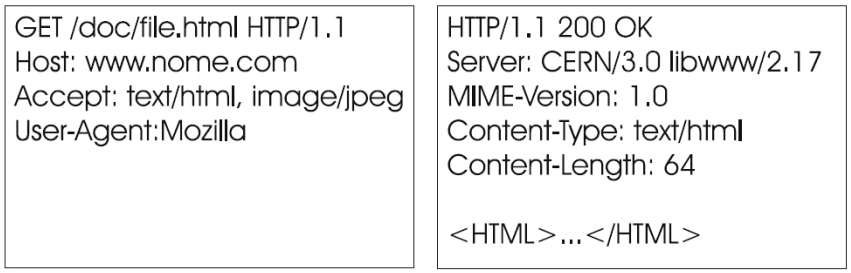

Figura 2.2: Exemplo de Requisição e Resposta HTTP

O cliente pode enviar comandos ao servidor invocando um conjunto de métodos definidos pelo protocolo HTTP. A versão 1.0 do protocolo HTTP é descrita na RFC 1945 (Berners-Lee et al., 1996), na qual estão definidos os métodos GET, HEAD e POST. No protocolo HTTP 1.1, descrito na RFC 2616 (Fielding et al., 1999), são acrescentados os métodos OPTIONS, PUT, DELETE, TRACE e CONNECT. Uma breve descrição desses métodos pode ser vista na Tabela 2.1:

Método	*Finalidade*
GET	Faz a requisição do recurso especificado pela URL
POST	Envia ao servidor informações do cliente, geralmente, digitadas em formulários HTML
HEAD	Utilizado para obter informações de um recurso sem retorná-lo ao cliente. Testa a validade de *links*, acessibilidade e a data da última atualização
OPTIONS	Usado para obter opções de comunicação disponíveis. Permite ao cliente determinar os requisitos associados ao recurso requisitado
PUT	Cria ou modifica um recurso no servidor
DELETE	Faz a solicitação para apagar um recurso no servidor, identificado na URL
TRACE	Envia mensagem de teste ao servidor
CONNECT	Reservado para servidores *proxy*

Tabela 2.1: Métodos definidos pelo protocolo HTTP

Para informar o resultado da execução realizada, o servidor retorna em sua resposta HTTP um código de status. Esse código pode pertencer a uma das classes apresentadas na Tabela 2.2:

Classe	*Descrição*
1xx	Finalidade Informativa
2xx	Sucesso
3xx	Redirecionamento
4xx	Erro do cliente
5xx	Erro do servidor

Tabela 2.2: Classes de código de status das respostas HTTP

2.5.2 HTTP 1.0 e 1.1

O protocolo HTTP 1.0 foi introduzido juntamente com a Web, em 1990. Nessa versão inicial, ele nada mais era do que um modo simples de recuperar dados por meio da Internet. Com o crescimento da Web, surgiram novos requisitos, e algumas de suas fraquezas foram aparecendo.

A primeira delas é que o HTTP 1.0 exige o estabelecimento de uma nova conexão TCP para cada objeto solicitado. No início, quando os documentos da Web eram constituídos basicamente de texto, isso não chegava a ser um problema. Porém, atualmente, uma simples página HTML pode conter dezenas de pequenas imagens. Isso tende a causar uma grande sobrecarga no tráfego da Internet, bem como nos servidores.

O protocolo HTTP 1.1, padronizado em 1999 pelo W3C (*World Wide Web Consortium*) (W3C, 1999b), usa como padrão o esquema de conexões persistentes, que permite que uma mesma conexão TCP seja usada por várias transações HTTP, o que é bem mais eficiente. O HTTP 1.1 também permite fazer o *pipelining* de requisições. Nesse caso, várias requisições são enviadas em seqüência, sem aguardar pelas respostas. Isso é bastante útil, por exemplo, para recuperar as imagens de uma página em ambientes que possuam uma alta latência para o estabelecimento de conexões TCP.

2.5.3 Sessões HTTP

O crescimento da fidelidade dos clientes em relação aos serviços de banco e compras de produtos na Internet pode aumentar muito a demanda aos servidores Web. Como conseqüência, além de haver um alto tráfego na rede, devem existir garantias de atendimento a essa demanda, o que torna difícil, no atual modelo da Internet (Wei et al., 2003), o oferecimento de um bom nível de qualidade a esses serviços.

Tipicamente, um acesso ao servidor Web ocorre na forma de uma sessão, que é constituída

de várias requisições individuais de um mesmo cliente (Arlitt, 2000). Por exemplo: um pedido de um cliente a um *site* comercial pode envolver uma requisição para selecionar o produto, outra para o fornecimento das informações necessárias, uma para estabelecer um acordo de pagamento e uma última requisição para o recebimento de uma confirmação.

Assim, tanto para o cliente quanto para o *site* de vendas, é necessário que seja garantido o atendimento de todas as requisições pertencentes a uma transação comercial, pois o sucesso da transação é obtido somente quando o atendimento de uma seqüência de requisições for concluído.

2.6 Considerações Finais

Neste capítulo, foi abordada a infra-estrutura da Internet. Foi apresentada a arquitetura TCP/IP, sendo mostradas as características e funções de cada uma de suas camadas. Foi apresentado também um panorama geral da Web e discutida sua organização atual como plataforma de comunicação de clientes e servidores. Em seguida, foi realizado um breve resumo a respeito do funcionamento dos servidores Web e suas formas de organização. Por fim, foi estudado o protocolo HTTP.

No próximo capítulo, será mostrada a necessidade de introduzir qualidade de serviço na Web. Essa necessidade surge dados os problemas e limitações do modelo atual de serviços existente.

Capítulo

3

Qualidade de Serviço e Servidores

3.1 Considerações Iniciais

Neste capítulo, são discutidos os problemas atuais da Internet e como a inserção de qualidade de serviço pode tornar essa rede mais eficiente. A abordagem de serviços diferenciados sobre redes IP e a viabilidade de seu emprego em nível de aplicação, particularmente em servidores Web, são destacados. Em seguida, é apresentado o modelo para um Servidor Web com Diferenciação de Serviços, o SWDS. É descrita a sua arquitetura e também são discutidos alguns algoritmos de controle de admissão e diferenciação de serviços.

3.2 Limitações da Internet Atual

O tráfego da Internet tem crescido enormemente nos últimos anos, principalmente depois do surgimento da Web. Esse crescimento não se deve apenas ao aumento do número de usuários e aplicações, mas também se dá em decorrência do aparecimento de novos tipos de aplicações, principalmente multimídia e de tempo real, que exigem da Internet uma resposta para a qual ela não foi projetada (Teixeira, 2004).

A Internet é capaz de fornecer um serviço de melhor esforço (*best-effort*). A rede tenta transportar os dados no menor tempo possível e sem erros, mas não são dadas às aplicações

garantias de que isso ocorrerá. Esse modelo de melhor esforço é decorrente da própria concepção do protocolo IP, no qual a Internet é baseada, que foi projetado para ser simples, eficiente e flexível. Essa política funciona bem quando o tráfego se mantém inferior à capacidade da rede, porém, em fases de congestionamento, poderão ocorrer atrasos na entrega dos pacotes e perda de dados na rede (Vasiliou & Lutfiyya, 2000).

Desde a criação das redes IPs, procura-se deixar a complexidade nas bordas da rede (*hosts*), mantendo os elementos internos (roteadores) mais simples. Os roteadores fazem a verificação dos endereços IPs dos datagramas em uma tabela de rotas, para determinar qual o próximo "salto" (*hop*) a ser dado (Stardust, 1999b). Essa solução tem sido suficiente para a transmissão de dados convencionais – e-mail, transferência de arquivos e navegação na Web. Entretanto, ela se revela inadequada para os novos tipos de aplicações encontradas na rede, tais como: aplicações multimídia, telefonia sobre IP, vídeo-conferência, transmissão de rádio e TV (Teixeira, 2004). Essas aplicações possuem certos requisitos que a Internet não está preparada para atender. As aplicações multimídia, por exemplo, necessitam de grande largura de banda e suportam pequenas perdas de pacotes. Já para a telefonia sobre IP, são exigidos requisitos de temporização. Além disso, a Web tem se tornado uma plataforma de negócios, exigindo alta confiabilidade e disponibilidade do meio de transmissão, cujos usuários estão dispostos a pagar mais por um serviço mais previsível e de melhor qualidade (Xiao & Ni; 1999).

À primeira vista, o problema da Internet poderia parecer de largura de banda, ou seja, supondo-se que fosse possível acrescentar capacidade de transmissão indiscriminadamente à rede, teoricamente seria possível acolher todo o tráfego existente, satisfazendo a todos. Porém, além de economicamente inviável, não é a solução adequada para alguns casos, pois, para algumas aplicações, o ponto crítico não está na quantidade de largura de banda disponível, mas sim na latência de transmissão (Teixeira, 2004).

Sendo assim, faz-se necessário realizar um gerenciamento da largura de banda disponível, diferenciando o tráfego que passa pelos elementos internos da rede, e, dessa forma, fornecer classes ou níveis de serviço diferenciados aos usuários e aplicações (Xiao & Ni; 1999).

3.3 Conceitos Básicos de QoS

A idéia de aplicar Qualidade de Serviço em redes IP surgiu da necessidade de minimizar os problemas existentes nesse tipo de rede. Fornecer a um elemento da rede a garantia de que seus requisitos de tráfego e serviço serão satisfeitos é a principal função da Qualidade de Serviço (QoS) (Stardust, 1999b). Em outras palavras, é a capacidade da rede de prover serviço de encaminhamento de dados de forma consistente e previsível.

A QoS é obtida por meio da cooperação de todos os níveis da rede, desde o topo até a base, bem como de todos os elementos da rede fim-a-fim. Qualquer garantia de QoS será tão forte quanto o mais frágil elemento do caminho da transmissão de dados na rede (Stardust, 1999b).

Os mecanismos de QoS, conforme a demanda das aplicações, administram a largura de banda existente na rede. Habilitar QoS para as aplicações de alta prioridade não deve implicar na anulação das aplicações menos prioritárias.

A QoS pode ser descrita de forma *absoluta* ou de forma *relativa*. Uma especificação de QoS em termos absolutos fornece métricas a serem cumpridas, relacionadas, em geral, ao atraso ou à perda de pacotes. Já a QoS relativa faz diferenciação de serviços, garantindo que uma aplicação em uma classe de mais alta prioridade nunca receba um serviço pior que o de qualquer classe inferior (Zhao et al., 1999).

Duas características importantes dentro do tema de QoS são identificadas: *garantia de desempenho* e *diferenciação de serviços* (Zhao et al., 1999). A primeira característica está relacionada ao gerenciamento de largura de banda, perda de pacotes, atraso e *jitter* (variação do atraso). A segunda está relacionada com o fornecimento de níveis de QoS distintos para

diferentes aplicações.

Em Koruse & Ross (2006), são estabelecidos alguns princípios para o fornecimento de garantias de qualidade de serviço:

- Princípio 1: a classificação de pacotes permite que um roteador faça distinção de pacotes pertencentes a diferentes fluxos de tráfego;
- Princípio 2: é desejável fornecer um certo grau de isolamento entre os fluxos de tráfego, de modo que um fluxo não seja adversamente afetado por outro mal comportado;
- Princípio 3: ao fornecer isolamento de fluxos, é desejável que se usem os recursos da maneira mais eficiente;
- Princípio 4: é necessário um processo de aceitação de chamada pelo qual os fluxos declarem suas exigências de QoS e, em seguida, sejam admitidos ou não na rede.

Outro componente importante para a determinação do modelo de QoS a ser fornecido aos usuários diz respeito ao tipo de tráfego que as aplicações geram e qual o comportamento esperado da rede para que elas funcionem corretamente. Com relação ao tipo de tráfego, as aplicações podem ser classificadas em Braden et al. (1994):

- Aplicações de tempo real (não elásticas): exigentes no que diz respeito ao tempo de transporte dos seus dados (características rígidas de temporização). Essa classificação pode ainda ser quebrada em duas categorias:
    - Aplicações tolerantes: são aquelas que toleram variações no atraso (*jitter*) causadas pela rede;
    - Aplicações intolerantes: as variações no atraso são inaceitáveis;

- Aplicações elásticas (não tempo real ou adaptáveis): para esse tipo de aplicação, a recepção correta dos dados é mais importante do que a sua apresentação em uma taxa constante. Exemplos de aplicações elásticas são: correio eletrônico, transferência de arquivos, consultas interativas a informações (como na Web) e aplicações cliente/servidor tradicionais.

3.4 Arquitetura para QoS

Na Internet, destacam-se dois tipos básicos de arquiteturas para QoS (Stardust, 1999b; Zhao et al., 1999):

- Baseada na Reserva de Recursos: os recursos da rede são atribuídos às aplicações segundo suas demandas de QoS e submetidos à política de gerenciamento de largura de banda. Essa é a abordagem empregada pela arquitetura de Serviços Integrados (*IntServ*);

- Baseada na Priorização: o tráfego na rede é classificado de acordo com suas características de demanda e recebe os recursos segundo a política de gerenciamento vigente. As classes de tráfego mais exigentes recebem tratamento preferencial, abordagem adotada pela arquitetura de Serviços Diferenciados (*DiffServ*).

3.4.1 Serviços Integrados

A idéia principal da arquitetura *IntServ* é a de reserva de recursos. As aplicações precisam encontrar um caminho até o receptor que satisfaça as demandas de QoS e realizar a reserva dos recursos necessários ao longo do mesmo, antes do início da transmissão dos dados. Esse sistema exige dos roteadores a manutenção de informações de estado para cada fluxo de

dados, levando complexidade para o núcleo da rede. Apesar de fornecer a maior QoS possível em uma rede baseada nos protocolos TCP/IP, pois realiza a reserva de recursos fim-a-fim, representa uma ruptura com a abordagem tradicionalmente empregada nas redes IP. Seu objetivo é fornecer um serviço mais próximo possível da abstração de circuitos virtuais, em uma rede comutada por pacotes, como a Internet.

São definidas duas classes de serviço na arquitetura *IntServ*, além do tradicional modelo de melhor esforço encontrado nas redes IP: *Serviço Garantido* e *Serviço de Carga Controlada*:

- Serviço Garantido: é garantida a disponibilidade de largura de banda e é fornecido um limite superior para o atraso de comunicação. As aplicações que possuem requisitos estritos de tempo real podem utilizar esse serviço;

- Carga Controlada: esse serviço oferece uma única função. As aplicações que fazem reserva de QoS usando os serviços de carga controlada recebem um serviço equivalente ao serviço fornecido pelo modelo de melhor esforço em condições de sobrecarga leve. Não é garantido um atraso máximo, apenas um limiar probabilístico, assim como não é assegurado que não haverá perda de pacotes. Os serviços de carga controlada são projetados para aplicativos que podem tolerar uma quantia razoável de perda e retardo de pacotes, tal como aplicações de áudio e videoconferência.

Protocolo RSVP

Para a arquitetura *IntServ*, deve ser realizada a reserva de recursos ao longo do caminho entre o transmissor e o receptor antes de a transmissão de dados ser iniciada. Em princípio, a reserva de recursos pode ser executada por qualquer protocolo que seja compatível com o modelo, mas na prática o protocolo RSVP (*Resource Reservation Protocol*) (Braden et al., 1997) é o padrão.

O protocolo RSVP é considerado como uma das soluções mais complexas em suporte ao fornecimento de QoS (Stardust, 1999a). É um protocolo de controle e sinalização que atua na camada de rede.

Algumas características importantes do protocolo RSVP são:

- O RSVP faz reservas tanto para transmissões *unicast* como para transmissões *multicast*;
- O RSVP faz reservas somente para fluxos unidirecionais. Para conseguir reservas bidirecionais, deve solicitar duas reservas unidirecionais distintas;
- No RSVP, é o receptor dos dados o responsável por iniciar e manter as reservas para os fluxos;
- O estado das reservas no RSVP segue a abordagem *soft-state*, ou seja, expira após um certo período de tempo, sendo necessário renová-lo periodicamente;
- O RSVP não é um protocolo de roteamento. Ele utiliza as rotas escolhidas pelo protocolo de roteamento adotado;
- O RSVP permite fazer o gerenciamento de fluxo em uma granulosidade bem fina, conseguindo-se altos níveis de QoS na Internet;
- Para conseguir QoS, é necessário que todos os roteadores ao longo do caminho dêem suporte ao RSVP, mantendo informações de estado e escalonando pacotes para cada fluxo de dados.

É importante destacar que o RSVP leva uma complexidade significativa para o núcleo da Internet, uma vez que exige que os roteadores gerenciem informações relativas aos fluxos de

dados que passam por eles. Isso leva a um rompimento com o modelo tradicional de serviços da Internet, que sempre procurou deixar a complexidade nas bordas da rede. Portanto, a abordagem de serviços integrados é mais bem empregada em um ambiente de rede local, fornecendo uma QoS para aplicações de um domínio.

3.4.2 Serviços Diferenciados

A abordagem *DiffServ* baseia-se na idéia de agregação de fluxos em umas poucas classes de serviço (Magalhães & Cardozo, 1999). Realiza a marcação de pacotes nos pontos de ingresso na rede de forma distinta, o que permite que os mesmos sejam tratados internamente de maneira diferenciada, segundo suas classes de serviço. Esse método, ao contrário do modelo *IntServ*, que exige complexidade por todo percurso da entrega de dados, mantém a complexidade nas bordas da rede, conservando um dos princípios básicos do projeto da Internet.

Baseado em um esquema de prioridades relativas, o modelo *Diffserv* garante um melhor tratamento ao tráfego gerado pelas aplicações com maior prioridade (Xiao & Ni, 1999). Para isso, é feita uma classificação de pacotes, para que eles sejam tratados de maneira diferenciada no interior da rede.

Os pacotes são classificados modificando-se a definição do layout do octeto *Type-of-Service* do cabeçalho do protocolo IPv4 (ou o campo *Traffic Class* do IPv6). Utilizando um campo do próprio datagrama IP para a marcação de pacotes, não há necessidade de um protocolo para o *DiffServ*, sendo somente preciso que o roteador examine essa informação para distinguir os pacotes entre um certo número de classes de serviço pré-definidas, dando a eles o tratamento conveniente.

Atualmente são definidas na arquitetura de serviços diferenciados duas classes de serviços principais, *Encaminhamento Garantido* e *Encaminhamento Expresso*, que são abordadas a

seguir (Kilkki, 1999):

- Encaminhamento Garantido: a classe de serviço Encaminhamento Garantido (*Assured Forwarding - AF*) oferece um serviço mais confiável do que o de melhor esforço. É garantido um tratamento preferencial ao tráfego, mas sem oferecer limites superiores para o atraso e *jitter*. O tráfego é dividido em classes com diferentes níveis de precedência de descarte. Dessa forma, o nível de garantia de encaminhamento de um pacote depende da quantidade de recursos alocados para a classe, a carga atual da classe e, em caso de congestionamento, o nível de descarte do pacote;

- Encaminhamento Expresso: o serviço Encaminhamento Expresso (*Expedited Forwarding - EF*) ou *Premium Service* define garantias mais rígidas de QoS para aplicações muito sensíveis a variações de características temporais da rede. Ele pode ser utilizado para implementar um serviço com pouco atraso, pouca variação do atraso (*jitter*) e largura de banda garantida. Para os usuários, esse serviço parece com uma linha privativa virtual. Com o serviço de Encaminhamento Expresso, os pacotes da classe *Premium* são colocados em uma fila de maior prioridade e sempre são os primeiros a serem encaminhados. Além disso, sempre é evitado que os pacotes com esse contrato sejam descartados.

Service Level Agreements

Um componente central dos Serviços Diferenciados é o *Service Level Agreement* (*SLA*) - acordo de nível de serviço. O SLA é um contrato de serviço entre um cliente e um provedor de serviços que especifica os detalhes da classificação de tráfego e o serviço de encaminhamento que um cliente deve receber.

O provedor de serviços precisa garantir que o tráfego de um cliente, com o qual ele tem um SLA, obtenha a QoS contratada. Assim, a administração da rede do provedor de serviços

precisa definir os planos de ação dos serviços apropriados e medir o desempenho da rede para garantir o desempenho de tráfego combinado.

As políticas de QoS podem ser aplicadas baseadas em números de portas, data e hora, endereços de origem e destino, ou em outras informações que estejam contidas no tráfego (Stardust, 1999b). Além disso, podem ser especificados procedimentos de tarifação e cobrança, serviços de criptografia e autenticação, procedimentos de renegociação dos parâmetros do SLA, entre outros (Vasiliou & Lutfiyya, 2000).

3.5 Serviços Diferenciados em Nível de Aplicação

Como já citado anteriormente, qualquer esforço para o fornecimento de QoS na Web não poderá ter sucesso se apenas mecanismos em nível de rede e sistema operacional forem utilizados, pois, em última instância, são os servidores Web os responsáveis pelo atendimento das solicitações dos usuários. Dessa forma, para que haja fornecimento de QoS, é necessária a cooperação de todos os elementos do sistema.

Infelizmente, a grande maioria dos servidores atuais ainda trata todas as solicitações igualmente, conforme uma disciplina FCFS (*First-Come First-Served*). Dessa forma, não é possível haver uma provisão de QoS na Web, uma vez que, para isso, é fundamental que esses servidores sejam providos de políticas de atendimento que consigam realizar a diferenciação de serviços.

Esse é um importante assunto a ser tratado, pois a Web tem sido fortemente utilizada como um meio para a condução de negócios, cujo elemento central é o servidor Web.

Em Teixeira (2004), é proposta uma arquitetura para um servidor Web que o torne capaz de fornecer serviços diferenciados a seus usuários e aplicações, de acordo com suas características de demanda. A Figura 3.1 descreve a arquitetura proposta, que é composta pelos seguintes módulos: um Classificador, um módulo de Controle de Admissão e um *cluster* de processos ou servidores Web:

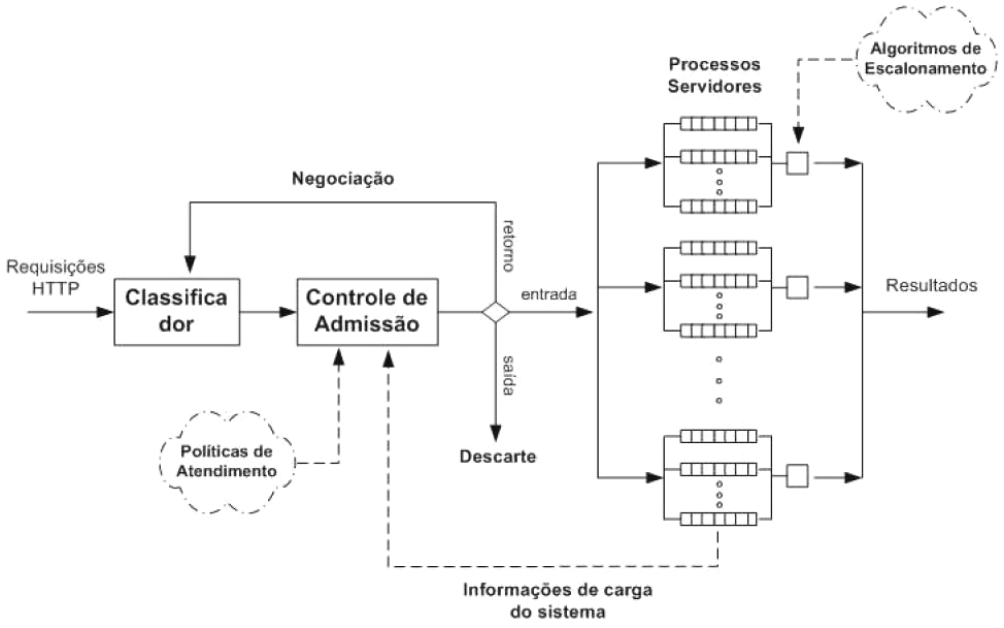

Figura 3.1: Servidor Web com Diferenciação de Serviços (SWDS)

O Classificador é o elemento responsável por receber as requisições que chegam ao sistema e subdividi-las em classes de serviço, segundo critérios preestabelecidos. Após essa fase, a nova requisição entra no sistema em uma determinada categoria e recebe um tratamento condizente com a classe à qual pertence.

O Controle de Admissão recebe as requisições já classificadas e gerencia a sua aceitação pelo servidor, levando em conta as políticas de atendimento vigentes e as informações da carga de trabalho do sistema. Caso o servidor esteja sobrecarregado, uma requisição poderá ser rejeitada (descarte) ou poderá ter suas exigências de qualidade de serviço relaxadas (negociação), a fim de que possa ser aceita em outra classe de prioridade (Estrela et al., 2006).

Após ser admitida no sistema, a requisição é atribuída a um dos nós do *cluster* de servidores Web, sendo atendida conforme o algoritmo de escalonamento ou diferenciação de serviços vigente. Após a conclusão do processamento, os resultados são retornados ao cliente que originou a solicitação.

Para o modelo SWDS, não é suposta nenhuma plataforma de *hardware* ou sistema

operacional específicos. Os nós do *cluster* são considerados como servidores Web convencionais, compostos de CPU, disco e interface de rede.

Apesar de ser formado por servidores Web, o *cluster* do modelo pode ser abstraído para processos ou até mesmo CPUs em um computador paralelo, pois não há exigências de que a arquitetura seja necessariamente formada por máquinas dispostas em um sistema distribuído.

Uma característica importante do modelo SWDS é a sua flexibilidade, pois admite vários tipos de mecanismos de controle de admissão e diferenciação de serviços.

3.5.1 Controle de Admissão

Em função das características da carga de trabalho da Web, o Controle de Admissão é um mecanismo essencial em servidores. A carga que atinge os servidores pode, de modo imprevisível, sobrecarregar o sistema e, dessa forma, comprometer toda a diferenciação de serviços realizada para atingir os níveis de QoS pretendidos. Portanto, o principal objetivo do módulo de controle de admissão do servidor SWDS é fazer com que a carga do sistema seja sempre mantida dentro de níveis aceitáveis. Para isso, são consideradas as informações da carga do sistema, além das políticas de atendimento.

Mecanismos de Controle de Admissão

Foram implementados, por Teixeira (2004), três mecanismos de controle de admissão no SWDS: Tamanho das Filas, Tempo de Resposta e Utilização do Sistema.

O mecanismo que considera o tamanho das filas impõe um tamanho máximo para as filas dos servidores do *cluster*, recusando novas requisições quando esse limite for atingido. A principal função desse mecanismo é manter a carga do sistema em níveis aceitáveis, mas ele não oferece garantias de QoS aos clientes.

O mecanismo baseado no tempo de resposta preocupa-se em manter o tempo de resposta das requisições de alta prioridade abaixo de um certo limiar. Seu funcionamento consiste em

aceitar as requisições de alta prioridade sempre que possível e somente admitir aquelas de baixa prioridade quando a carga estiver abaixo do limiar pré-determinado. Com esse mecanismo, evita-se que o tempo de resposta apresente picos devido aos aumentos na carga de trabalho, conseguindo-se uma maior estabilidade no sistema.

O mecanismo que leva em consideração a utilização do sistema emprega uma média exponencialmente ponderada da utilização do *cluster* para o controle de admissão, a qual pode ser ajustada conforme valores atuais ou o histórico da carga do sistema. Esse mecanismo consegue, de forma rigorosa, manter a utilização do servidor SWDS dentro do limiar especificado, o que permite oferecer um tratamento mais justo a todas as classes de requisições.

São encontrados na literatura outros trabalhos que consideram a presença do Controle de Admissão fundamental para que os servidores ofereçam qualidade de serviço, como os trabalhos de Cardellini et al. (2001), Lee et al. (2002) e Andreolini et al. (2004).

As políticas de controle de admissão podem ainda considerar outros parâmetros como base para a tomada de decisões, como as características das requisições Web, por exemplo.

3.5.2 Mecanismos de Diferenciação de Serviços

Com o objetivo de realizar o balanceamento de carga e prover diferenciação de serviços no Servidor Web com Diferenciação de Serviços (SWDS), foram propostos em Teixeira (2004) e Traldi et al. (2006) alguns algoritmos de diferenciação de serviços. Esses algoritmos especificam a forma de compartilhamento do *cluster* de servidores Web entre as classes de serviço. Eles determinam qual categoria de clientes terá preferência no atendimento, realizando a diferenciação de serviços propriamente dita.

Em Teixeira (2004) são propostos algoritmos de diferenciação de serviços baseados em *Reserva de Recursos* e algoritmos baseados em *Prioridades*, além dos algoritmos que objetivam apenas o balanceamento de carga.

Os algoritmos baseados na *Reserva de Recursos* fazem a subdivisão do *cluster* de servidores Web do SWDS em partições. É feita então uma associação dessas partições às classes de serviço. Com isso, uma parcela da capacidade de processamento do servidor é reservada para cada classe de requisições. Nos algoritmos baseados em *Prioridades*, a diferenciação é realizada atribuindo-se prioridades às requisições presentes nas filas do *cluster*.

Como algoritmo baseado em *Reserva de Recursos*, foi apresentado em Teixeira (2004) o RSV (Algoritmo de Reserva de Recursos). O algoritmo RSV subdivide os nós do *cluster* do SWDS em partições e associa cada partição a uma classe ou um conjunto de classes de serviço, de maneira estática. Esse algoritmo apresenta bons resultados quando a carga a que o sistema é submetido é previamente conhecida, uma vez que o particionamento dos recursos é realizado de maneira não dinâmica. Entretanto, para ambientes onde a carga varia, como no caso da Web, não é aconselhada a sua utilização.

Como algoritmos baseados em *Prioridades*, são propostos em Teixeira (2004) um Mecanismo de Prioridades Rigoroso e um Mecanismo de Prioridades Adaptativo.

O Mecanismo de Prioridades Rigoroso exige que o atendimento das requisições siga uma disciplina de prioridades rígida, ou seja, as requisições de prioridade inferior somente serão atendidas quando não houver nenhuma requisição de prioridade superior aguardando na fila. Esse método garante que as requisições de maior prioridade sempre recebam atendimento preferencial. Entretanto, dependendo da carga a que o sistema é submetido, pode ocorrer negação de serviço às requisições de menor prioridade.

A fim de evitar o monopólio do sistema por parte das requisições de alta prioridade, foi proposto o Mecanismo de Prioridades Adaptativo. Com o mecanismo adaptativo, pode-se atribuir maior ou menor importância às classes de maior prioridade por meio da utilização de um parâmetro k, denominado *look-ahead*, que determina o número máximo de posições da

fila de espera que serão percorridas a partir do início, à procura de requisições de uma determinada prioridade. Caso não encontre nenhuma requisição do tipo especificado, o algoritmo será repetido para o nível de prioridade imediatamente inferior. Quanto maior o valor de k, melhor será o tratamento dado às requisições de maior prioridade. Uma característica indesejável desse método é que, dependendo do valor de k, uma requisição da classe de menor prioridade pode esperar por muito tempo na fila para ser atendida, mesmo que o tempo de processamento necessário para o seu atendimento seja muito pequeno se comparado ao tempo necessário para o atendimento das requisições de maior prioridade. Isso leva a um aumento no tempo médio de espera na fila.

Em Traldi et al. (2006) é apresentado o algoritmo de Reserva Adaptativa de Recursos (RSVAdap). Esse algoritmo foi desenvolvido com o intuito de adaptar o funcionamento do algoritmo RSV para ambientes em que a carga é variável. O RSVAdap aloca os nós do *cluster* nas classes de usuários (requisições) sob demanda, segundo a carga de trabalho vigente. Para a alocação dos recursos, o algoritmo se baseia no número de requisições de cada classe presente no sistema no momento da análise e no nível de diferenciação pretendido. Definido o particionamento do *cluster*, o algoritmo RSVAdap, que atua no módulo escalonador do modelo SWDS, distribui as requisições pertencentes a uma determinada classe aos nós reservados a ela, utilizando o conceito de *Shortest Queue First (SQF).* Uma característica importante do algoritmo RSVAdap é que, havendo requisições de uma determinada classe no sistema, independentemente de seu nível de prioridade, o algoritmo realiza a reserva de pelo menos um nó do *cluster* a essa classe, impedindo que seja negado atendimento a essas requisições. Isso pode levar, em algumas situações, ao desperdício de recursos do *cluster* e também à diferenciação de serviços inadequada.

Em Traldi et al. (2006) foi realizada a adaptação para o nível de aplicação do algoritmo *Weighted Fair Queuing (WFQ)* projetado para o nível de rede. Esse algoritmo é baseado em

Prioridades e utiliza a idéia de dividir a capacidade de processamento dos nós do *cluster* entre as classes de requisições de acordo com os pesos que são atribuídos às classes. Como parâmetro para o escalonamento, o algoritmo também considera o tempo de processamento necessário para atender uma requisição, priorizando requisições menos custosas. Por não realizar reserva de recursos, esse mecanismo evita a subutilização do sistema.

Em Traldi (2005) é apresentado o algoritmo WFQ Adaptativo. Esse algoritmo baseia-se no funcionamento do algoritmo WFQ, entretanto também apresenta bons resultados para ambientes em que a carga é variável.

3.6 Considerações Finais

Neste capítulo, foram apresentadas as limitações da Internet atual, que motivam a inserção de QoS nessa rede. Foram apresentados também os conceitos básicos de QoS, bem como as arquiteturas utilizadas para a provisão de qualidade de serviço.

Em seguida, foi discutida a importância de ter diferenciação de serviços também em nível de aplicação e apresentada a arquitetura do Servidor Web com Diferenciação de Serviços, o SWDS. Foram também mostradas algumas políticas de controle de admissão e políticas de diferenciação de serviços. Essas políticas são fundamentais para que o SWDS consiga fornecer qualidade de serviço no atendimento às requisições Web.

Os resultados fornecidos pelas políticas descritas motivam a investigação de novas estratégias de controle de serviços. No entanto, essas políticas não levam em consideração as características das requisições Web na tomada de decisões, baseando-se apenas na classe da tarefa, que é definida pela sua prioridade.

Para avaliar o desempenho do modelo de Servidor Web com Diferenciação de Serviços (SWDS) quando são consideradas as características das requisições Web nas políticas de atendimento, é necessária a utilização de alguma técnica de Avaliação de Desempenho.

No próximo capítulo, serão apresentadas algumas das técnicas de Avaliação de Desempenho encontradas na literatura e mostrada a metodologia adotada neste trabalho.

Capítulo

4

Avaliação de Desempenho

4.1 Considerações Iniciais

Na computação, há sempre a preocupação em obter informações relevantes a respeito dos sistemas. Por meio dessas informações, obtidas no processo de avaliação de desempenho, pode-se verificar se o sistema avaliado está realizando bem as tarefas para as quais foi designado.

Assim, para analisar os mecanismos de diferenciação desenvolvidos para o SWDS, foram utilizadas técnicas de avaliação de desempenho.

A seguir, são apresentadas algumas dessas técnicas e mostrada a metodologia adotada neste trabalho.

4.2 Técnicas de Avaliação de Desempenho

Para a avaliação de desempenho de um sistema, pode-se utilizar uma das técnicas que se encontram distribuídas em dois grandes grupos: Técnicas de Aferição e Técnicas de Modelagem (Santana, 1990a; Santana, 1990b). As técnicas de Aferição mais utilizadas são: construção de Protótipos, utilização de *Benchmarcks* e Coleta de Dados. Quanto à

modelagem, as técnicas de Redes de Petri, *Statecharts* e Rede de Filas são as mais utilizadas. A hierarquia das Técnicas de Avaliação de Desempenho é mostrada na Figura 4.1:

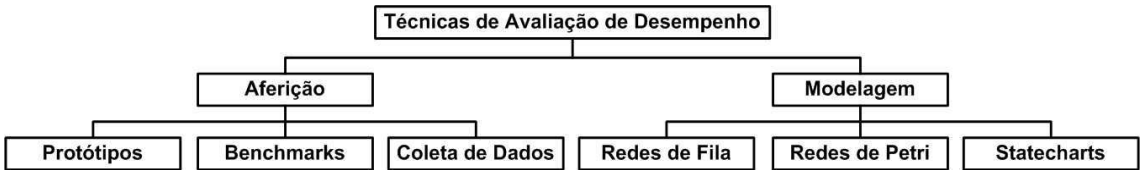

Figura 4.1: Hierarquia das Técnicas de Avaliação de Desempenho

4.2.1 Técnicas de Aferição

Técnicas de Aferição são aplicadas principalmente na avaliação de sistemas já existentes ou em fase final de desenvolvimento (Santana et al., 1997).

No caso da aferição de um sistema, são utilizados monitores de hardware ou de software (Jain, 1991; Hofmann et al., 1994). É necessário que haja uma grande preocupação em não alterar o funcionamento natural do sistema durante a experimentação prática.

4.2.2 Técnicas de Modelagem

Um modelo é uma descrição do sistema, constituindo uma *abstração* do sistema, em que são incorporadas as suas características fundamentais vistas sob uma ótica específica.

Uma das tarefas mais difíceis no processo de modelagem é decidir quais elementos do sistema devem ser incluídos no modelo. Assim, ao modelar um sistema, deve-se procurar abstrair quais de seus elementos são importantes na resolução de uma dada questão particular e quais relações entre esses elementos são pertinentes à resolução do modelo. Dessa forma, podem-se ter várias descrições (modelos) de um sistema, cada uma mais adequada à resolução de um problema particular (Soares, 1990) - isto constitui as várias visões que se tem do sistema.

Freqüentemente os modelos são classificados de acordo com os métodos utilizados na

obtenção dos resultados numéricos (medidas de desempenho). Assim, *modelos analíticos* são definidos como modelos cuja estrutura é formada por uma série de equações matemáticas, por meio dos quais o comportamento do sistema pode ser obtido pela atribuição dos valores aos parâmetros do modelo e pela solução das equações. Já os *modelos para simulação* podem ser definidos como aqueles representados por uma estrutura matemática/lógica, que pode ser exercitada de forma a imitar o comportamento do sistema por meio de programas computacionais, por exemplo. Com o experimento (simulação), várias observações são realizadas para dar subsídio às conclusões sobre o sistema (Soares, 1990). Pode-se ainda enxergar a natureza analítica ou de simulação em função da técnica empregada para solucionar o modelo. Isto é, tendo-se um modelo, se for viável obter a sua solução a partir de equações que o representem, então o modelo será resolvido analiticamente e se referi-rá a ele como um modelo analítico. Se, por outro lado, a solução do modelo for obtida por meio de um programa de simulação, tratar-se-á de uma solução por simulação e o modelo será dito, muitas vezes, um modelo de simulação (Santana, 1990a).

Existem várias técnicas de representação de sistemas adequadas à modelagem de sistemas computacionais. Entre elas, destacam-se: Redes de Petri, *Statecharts* e Rede de Filas. A seguir, a técnica de modelagem por Rede de Filas é detalhada.

Rede de Filas

Rede de Filas é uma técnica baseada na Teoria de Filas criada para modelar sistemas nos quais existe a ocorrência de filas (Francês, 1998).

Uma rede de filas básica consiste de entidades chamadas *centros de serviços* e um conjunto de entidades chamadas de *usuários*, que recebem serviço e circulam pelo sistema (Soares, 1990).

Um centro de serviço consiste de um ou mais servidores, correspondentes a recursos no sistema modelado, e uma área de espera (uma fila) de capacidade finita ou infinita para usuários que estão requisitando serviços. A CPU de um computador é um exemplo de recurso (Soares, 1990).

Para que possa existir distinção entre os vários usuários que circulam em um sistema, utiliza-se o conceito de Classes de Usuários. Assim, quando o modelo possui mais de uma classe de usuários, pode-se estabelecer prioridades entre elas.

Medidas de Desempenho

No processo de avaliação de desempenho de um sistema, algumas medidas são fundamentais. Essas medidas são chamadas de medidas de desempenho.

Para a avaliação de um sistema, deve-se determinar um conjunto de medidas de desempenho a serem colhidas e, por meio dessas medidas, verificar se o sistema está atendendo aos requisitos de desempenho estipulados previamente.

A seguir são apresentadas algumas medidas de desempenho, considerando-se um sistema em que há formação de filas, que é o caso dos sistemas computacionais:

- A **utilização de um recurso** é a fração do tempo em que o recurso esteve ocupado. Ela pode ser obtida por meio da razão entre o tempo que o recurso esteve ocupado durante um intervalo e o tempo total de duração desse intervalo;

- A **vazão** (*throughput*) é uma medida da taxa de conclusão dos trabalhos. Ela é dada pelo número de completudes por unidade de tempo;

- O **tamanho (comprimento) médio da fila** é o número médio de usuários em

espera e em serviço;

- O **tempo médio no sistema (centro de serviço)** é o tempo médio que os usuários gastam esperando na fila e em serviço.

O conhecimento pleno e prévio do sistema é fundamental para a escolha adequada da técnica a ser utilizada, pois o uso de uma técnica inadequada pode levar ao fracasso da avaliação. O estado de desenvolvimento do sistema a ser avaliado também deve ser levado em consideração.

4.3 Avaliação de Desempenho de Servidores Web

O uso da técnica de modelagem torna-se bastante atrativo quando o objeto de estudo são sistemas complexos ou ainda inexistentes, pois um modelo é uma abstração de uma situação real, em que se procuram incorporar suas características mais relevantes. No caso da Internet, devido ao seu tamanho e constante evolução, a avaliação de desempenho baseada em modelos se impõe como a alternativa mais adequada (Teixeira, 2004).

Para avaliar o desempenho de um sistema, é de suma importância caracterizar bem o tipo de carga a que ele normalmente é submetido, bem como o nível de detalhamento empregado na descrição da carga. Os recursos do sistema considerados na caracterização da carga devem ser aqueles cuja utilização tenha o maior impacto no desempenho do sistema (Ferrari et al., 1983). No caso de um servidor Web, a CPU e o sistema de I/O são os principais recursos a serem analisados. Finalmente, precisam ser determinados os parâmetros que melhor caracterizam cada componente da carga (taxa de chegada, número de clientes, tempo de CPU, de I/O).

Algumas características especiais devem ser consideradas quando se pretende avaliar o desempenho de um servidor Web:

- A população de usuários de um servidor Web é potencialmente ilimitada e as requisições podem vir de qualquer lugar do mundo;
- Um servidor Web pode ser submetido por longos períodos a cargas relativamente baixas e repentinamente ser alvo de cargas superiores à sua capacidade;
- A variabilidade no tamanho dos objetos armazenados em um servidor também deve ser considerada, pois a diversidade dos mesmos é muito grande. O servidor pode armazenar objetos que variam desde arquivos texto e HTML a arquivos multimídia;
- Atualmente os servidores também vêm sendo utilizados como servidores de aplicação, dando suporte à geração dinâmica de páginas.

A avaliação de desempenho é de especial importância quando se pretende avaliar o comportamento de um sistema considerando-se diferentes políticas de funcionamento, como por exemplo, os escalonadores em servidores Web e a inserção de características de QoS.

4.4 Avaliação de Desempenho do SWDS

A técnica escolhida para a avaliação de desempenho do Servidor Web com Diferenciação de Serviços (SWDS) foi a modelagem por meio de Rede de Filas. Foi adotado um modelo simplificado do servidor SWDS, conforme apresentado na Figura 4.2.

Esse modelo consiste de um Classificador, Controle de Admissão, um módulo de Escalonamento e um *cluster* de servidores Web. O módulo escalonador é responsável por fazer a distribuição das requisições entre os nós do *cluster*, assumindo as funções de um despachante. Cada nó do cluster é modelado com um recurso que representa o consumo de CPU e outro que representa o consumo de I/O, conforme Figura 4.3.

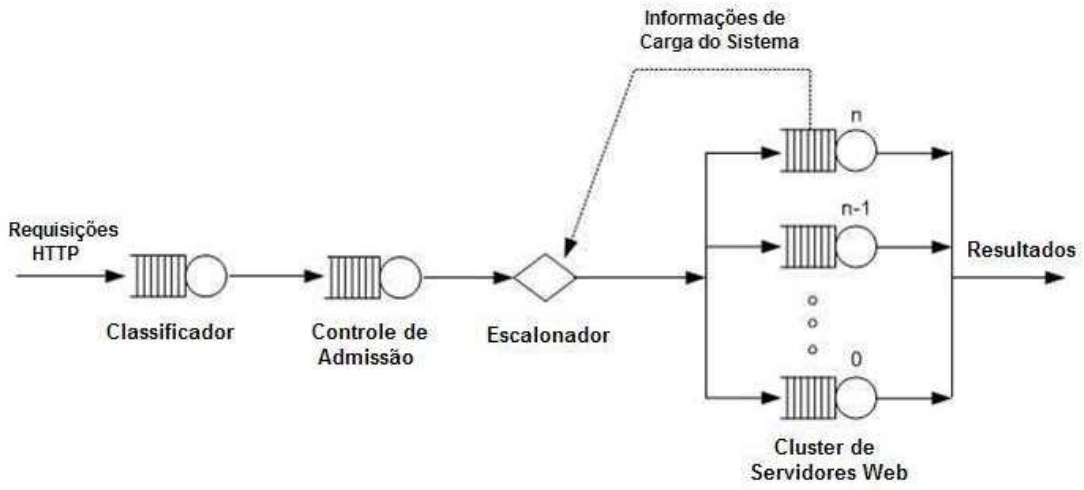

Figura 4.2: Modelo simplificado do Servidor Web com Diferenciação de Serviços (SWDS)

Classificador, Controle de Admissão, escalonador e os nós do *cluster* foram modelados como centros de serviço. Não foram modelados detalhes da rede externa, assumindo-se haver largura de banda suficiente na rede, uma vez que o foco principal é a análise do desempenho do SWDS e da sua capacidade de fornecer serviços diferenciados.

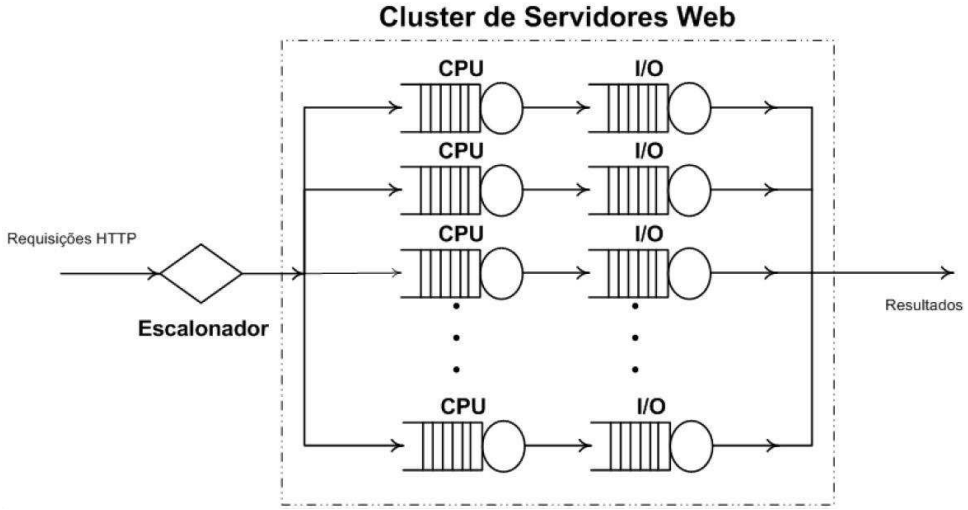

Figura 4.3: Modelo do *Cluster* de Servidores Web

4.5 Considerações Finais

Neste capítulo, foram mostradas algumas das técnicas de Avaliação de Desempenho

encontradas na literatura. Como técnicas de Aferição, foram discutidas a construção de Protótipos, a utilização de *Benchmarcks* e a Coleta de Dados. Quanto às técnicas de Modelagem, foram apresentadas as Redes de Petri, *Statecharts* e Rede de Filas.

Em seguida, discutiu-se sobre as medidas que são fundamentais para o processo de Avaliação de Desempenho, as medidas de desempenho.

Dando continuidade, foram exibidas algumas características dos servidores Web, que são sistemas cuja Avaliação de Desempenho baseada em modelos é considerada bastante adequada.

Finalmente, foi apresentado o modelo do Servidor Web com Diferenciação de Serviços utilizado para avaliar as estratégias de melhoria de desempenho propostas neste trabalho.

Para avaliar o desempenho de um sistema, é de suma importância caracterizar bem o tipo de carga a que ele normalmente é submetido. Por isso, no próximo capítulo serão apresentados os critérios utilizados para a geração de uma carga de trabalho sintética para um site de comércio eletrônico. Optou-se pelo comércio eletrônico porque esses serviços são uns dos mais impactados negativamente quando há um comportamento ruim do servidor em situações de sobrecarga.

No capítulo a seguir, também serão discutidas as características das requisições Web típicas do *e-commerce*, uma vez que se tem por objetivo melhorar o desempenho do modelo SWDS quando são consideradas tais características. As características consideradas devem ser as que causam os maiores impactos no desempenho do sistema. No caso de um servidor Web, os consumos de CPU e de sistema de I/O são as principais a serem analisadas.

Capítulo

5

Carga de Trabalho

5.1 Considerações Iniciais

Neste capítulo, é descrito o processo realizado para geração de uma carga de trabalho sintética para um site de comércio eletrônico. Para a realização de uma avaliação de desempenho sem a participação de uma empresa do mercado, é necessário reproduzir um cenário que retrate a realidade atual de sites de comércio eletrônico e gerar uma carga de trabalho sintética que retrate o mais próximo possível o atual comportamento dos consumidores de sites de comércio eletrônico.

Com o propósito de melhorar o desempenho do SWDS quando consideradas as características das requisições Web na tomada de decisão, para a geração da carga de trabalho, foram consideradas informações dos diversos tipos de requisições típicas de *e-commerce*.

O modelo de carga de trabalho foi definido com base em trabalhos relacionados.

5.2 Comércio Eletrônico

Segundo E-Bit (2007), os números e as taxas de crescimento das vendas *online* nos últimos

anos comprovam que o comércio eletrônico brasileiro está consolidado. Em 2007, esse segmento cresceu 43% em relação ao ano anterior, atingindo a cifra de R$ 6,3 bilhões faturados em vendas de produtos pela Internet. A expectativa para o primeiro semestre de 2008 é de que as compras cresçam aproximadamente 45% em relação ao primeiro semestre de 2007.

Com relação ao número de pessoas adeptas às compras no mundo virtual, as expectativas também são boas. 9,5 milhões de internautas já passaram pela experiência de comprar algum produto pela rede. Em 2008, espera-se que esse número suba para 10,5 milhões de e-consumidores.

Esses dados evidenciam que a demanda de tráfego para aplicações Web do gênero tem aumentado, uma vez que a utilização da Internet para comprar e vender produtos se torna um hábito cada vez mais comum. Esse aumento na demanda justifica a utilização de uma estrutura com diversos servidores Web para atender, com desempenho aceitável pelos consumidores, as requisições.

A carga de trabalho no ambiente *e-commerce* é caracterizada pela presença de sessões. As sessões são definidas por uma seqüência de requisições de diferentes tipos, feitas por um mesmo usuário durante uma visita ao site (Barbato, 2008). Neste trabalho, essas interações serão classificadas em duas categorias gerais:

- Categoria Navegar – envolvem navegação e pesquisa, mas nenhuma atividade de pedido de produtos. As interações típicas que abrangem essa categoria são: Página Principal, Listar Categorias, Visualizar Detalhes de produto e Buscar produto;
- Categoria Pedir – envolvem somente atividades de pedido de produtos e incluem as seguintes interações: *Login*, Local de Entrega, Forma de Pagamento e Confirmação de pedido.

Interações da Categoria Pedir têm maior probabilidade de gerar renda. Em situações de sobrecarga, servidores Web para sites de comércio eletrônico podem deixar de atender requisições que geram renda por conta do aumento excessivo na quantidade de requisições da Categoria Navegar. Essas situações merecem atenção, pois um comportamento adequado da estrutura da loja virtual em momentos de sobrecarga pode gerar resultados representativos ao negócio.

5.3 Modelo da Carga de Trabalho

Um dos problemas encontrados na simulação de sistemas relacionados à Web consiste na geração de uma carga de trabalho que reproduza um cenário o mais próximo da realidade atual.

Tendo em vista a dificuldade de encontrar um arquivo de registros com acessos reais ao site de uma loja varejista *online*, decidiu-se gerar uma carga de trabalho sintética para a experimentação do modelo.

Em uma típica loja virtual de varejo, muitos usuários navegam e realizam buscas pelas páginas do site a procura de um produto específico (interações do tipo Navegar). Se encontrarem o produto desejado e tiverem intenção de comprá-lo, poderão iniciar um processo de transação para a realização do pedido (interações do tipo Pedir). Existe ainda a possibilidade de o usuário abandonar o site durante qualquer uma das interações (Sabo, 2006).

Uma interação da Categoria Pedir é constituída de um primeiro passo: a página de *Login*. Desse ponto em diante, o usuário é orientado a seguir algumas etapas em seqüência (escolha do local de entrega da mercadoria e seleção da forma de pagamento) até culminar em uma requisição do tipo Confirmação, que finaliza o pedido.

Essas transições entre as diferentes categorias de interações (Navegar e Pedir) dos usuários pelo site podem ser representadas por uma máquina de estados finitos, em que as

probabilidades são estacionárias e as transições de estados sem memória. Dessa forma, em Menascé et al. (1999) são apresentados dois perfis de usuários: o perfil dos usuários considerados Compradores Ocasionais e o perfil dos usuários considerados Compradores Freqüentes.

A Figura 5.1 ilustra o comportamento dos usuários Compradores Ocasionais. Os estados correspondem às páginas que caracterizam as interações e os arcos representam as transições:

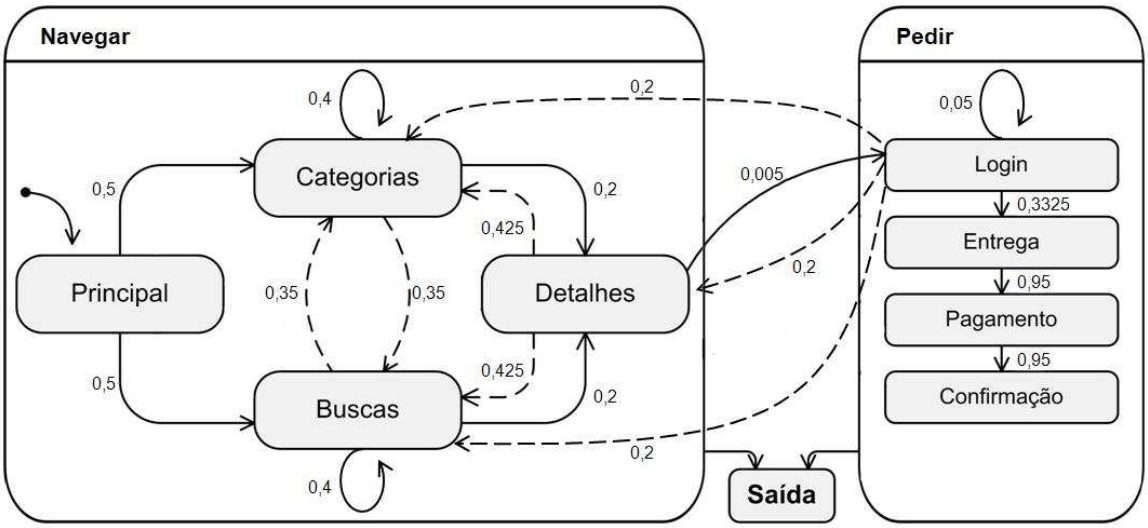

Figura 5.1: Compradores Ocasionais (Menascé et al., 1999)

Os Compradores Ocasionais são aqueles que usam os sites de comércio eletrônico para pesquisar por novos produtos, detalhes dos itens a venda, lançamentos e novidades.

A Figura 5.2 apresenta o comportamento dos usuários Compradores Freqüentes. Esses usuários são aqueles que possuem o hábito de efetivamente concluir o processo de compra.

Fica clara a diferença de comportamento entre os dois perfis, principalmente pela probabilidade de transição do estado Detalhes para o estado *Login*. Enquanto que para os Compradores Ocasionais essa probabilidade é de 0,5%, para os Compradores Freqüentes essa probabilidade é de 30%, ou seja, sessenta vezes maior.

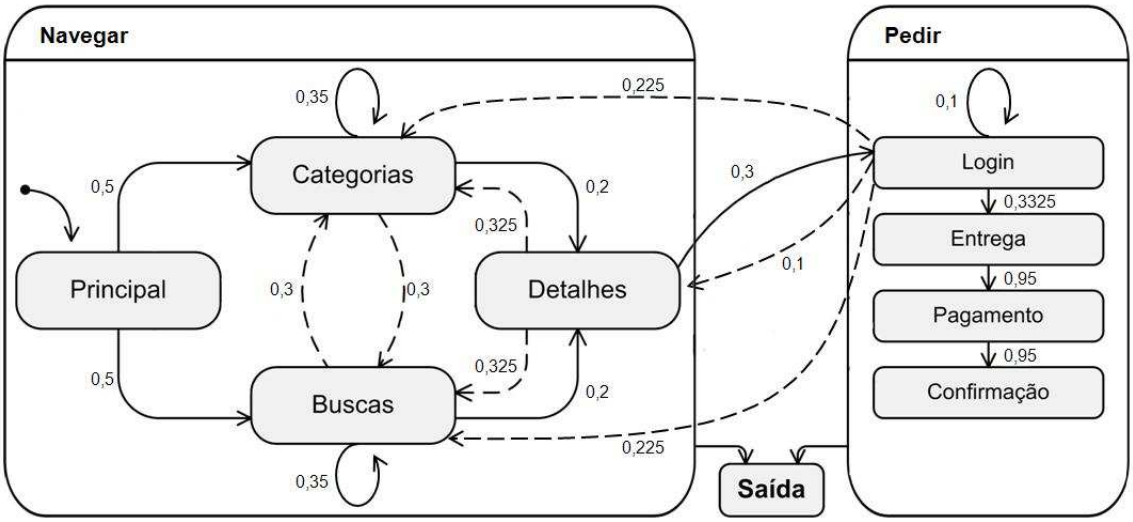

Figura 5.2: Compradores Freqüentes (Menascé et al., 1999)

Conforme Menascé et al. (1999), adotando-se 90% dos usuários que iniciam uma sessão como sendo do tipo Compradores Ocasionais e 10% como Compradores Freqüentes, tem-se a máquina de estados descrita na Figura 5.3:

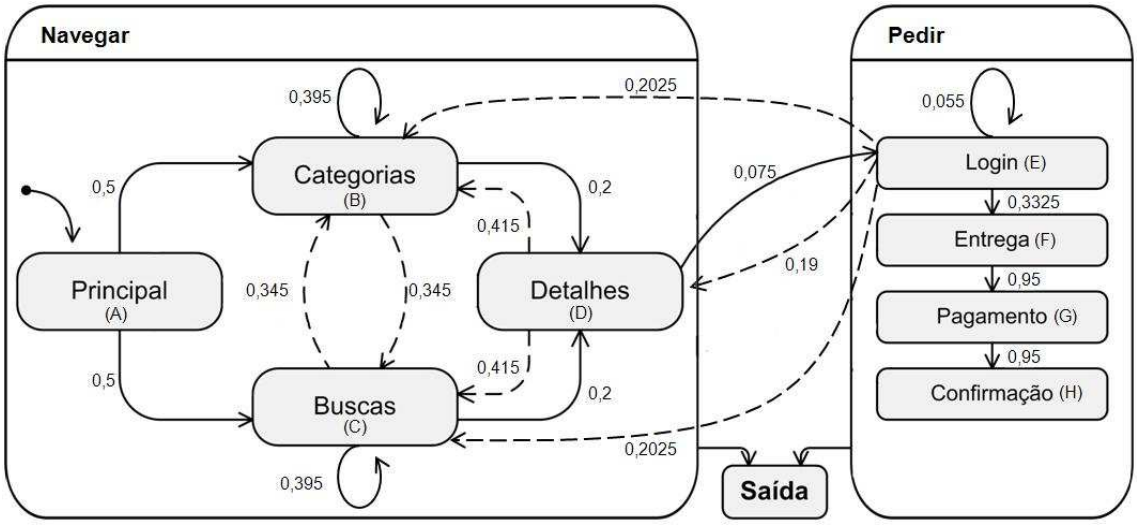

Figura 5.3: Máquina de Estados (Menascé et al., 1999)

Sabe-se que a probabilidade máxima de transição entre um estado e outro é 1. Na Figura 5.3, observa-se que a soma das probabilidades de transição de um estado para outros não atinge a totalidade. Por exemplo: a soma das probabilidades de transição do estado Categorias

para os estados Buscas (0,345), Detalhes (0,2) e/ou de retorno para o próprio estado Categorias (0,395) é de 0,94. Isso ocorre porque há ainda a probabilidade de o usuário deixar o site, que é de 0,06. Por uma questão de estética, isso não está representado na máquina de estados.

Tendo-se as probabilidades de transição de um estado para outro, é possível calcular a carga gerada ao sistema para cada requisição do tipo Principal atendida, ou seja, para cada nova sessão iniciada.

A seguir, é apresentado o sistema de equações que representa tal carga:

$$
\begin{array}{rrrrrrrrrr}
A & & & & & & & & = & 1 \\
-0{,}5A & +0{,}605B & -0{,}345C & -0{,}415\,D & -0{,}203E & & & & = & 0 \\
-0{,}5A & -0{,}345B & +0{,}605C & -0{,}415D & -0{,}2023E & & & & = & 0 \\
 & -0{,}2B & -0{,}2C & +D & -0{,}19E & & & & = & 0 \\
 & & & -0{,}075D & +0{,}945E & & & & = & 0 \\
 & & & & -0{,}333E & +F & & & = & 0 \\
 & & & & & -0{,}95F & +G & & = & 0 \\
 & & & & & & -0{,}95G & +H & = & 0
\end{array}
$$

Resolvendo-se o sistema de equações, obtém-se a carga média gerada ao sistema para cada nova sessão iniciada. Os resultados são apresentados na Tabela 5.1:

	Carga	**Carga Relativa**
A = Principal	1	6,44%
B = Categorias	5,887	37,90%
C = Buscas	5,887	37,90%
D = Detalhes	2,391	15,39%
E = *Login*	0,190	1,22%
F = Entrega	0,063	0,41%
G = Pagamento	0,060	0,39%
H = Confirmação	0,057	0,37%

Tabela 5.1: Carga média gerada ao sistema por cada sessão

Com o modelo de distribuição de tipos de requisições apresentado neste trabalho, é possível gerar uma carga de trabalho sintética que represente o comportamento atual de usuários e clientes de uma loja virtual varejista. Esse modelo de carga de trabalho foi utilizado

para a experimentação do modelo de simulação descrito na próxima seção.

5.4 Cenário Adotado

Com o modelo de carga de trabalho definido, decidiu-se por realizar experimentos reais e coletar tempos médios de resposta para cada categoria de página que recebe requisições Web. Com esses experimentos, pode-se obter o comportamento de um sistema Web real e encontrar valores para parametrizar as simulações de políticas de atendimento de requisições.

Para a obtenção de valores médios de tempos de resposta para cada tipo de requisição que compõe o modelo de carga de trabalho definido, seria necessário analisar registros de servidores Web reais que disponibilizassem esse tipo de informação. Tendo em vista a dificuldade de encontrar arquivos de registros do gênero, optou-se por reproduzir um cenário real o mais próximo da realidade atual, gerar as informações necessárias e realizar uma análise posterior.

Para desenvolver uma aplicação do gênero que implementasse as funcionalidades de uma loja virtual de varejo, seria necessário implementar um modelo de dados não trivial para armazenar informações de produtos em uma base de dados, dominar uma linguagem de programação para Web com o uso de sessão, *cache* , entre outras funcionalidades, e preencher uma base de dados com informações coerentes e equivalentes aos de um catálogo de uma loja virtual real. Assim, concluiu-se que o tempo necessário para o desenvolvimento de tal aplicação seria inviável e optou-se pela busca de alternativas.

Em Sabo (2006), é apresentado um estudo que utilizou como ferramentas a osCommerce (Leon et al., 2008), o banco de dados MySQL (MySQL, 2008) e o software Apache (servidor Web) para encontrar os valores médios de tempos de resposta para cada tipo de requisição que compõe o modelo de carga de trabalho. Como tal estudo já foi validado e apresentado à comunidade científica, decidiu-se pela utilização dessas informações neste trabalho.

A osCommerce é uma solução de código aberto e com licença GNU/GPL (*General Public License*) que implementa um modelo de loja varejista que vende produtos na Internet. Essa solução de código aberto oferece funções de comércio eletrônico que permitem aos consumidores navegar por categorias de produtos, procurar por informações sobre produtos existentes, ver detalhes de produtos, realizar pedidos e escolher o local de entrega e forma de pagamento. As interações relacionadas ao pedido de produtos são criptografadas por meio de conexões SSL (*Secure Socket Layer*). Os clientes precisam se registrar no site antes de terem permissão para comprar. O site mantém um catálogo de itens que podem ser pesquisados por um cliente. Cada item possui uma descrição textual e uma imagem miniatura associada com tamanho inferior a 5 Kbytes.

A Figura 5.4 ilustra o cenário que foi utilizado em Sabo (2006) para a realização dos experimentos:

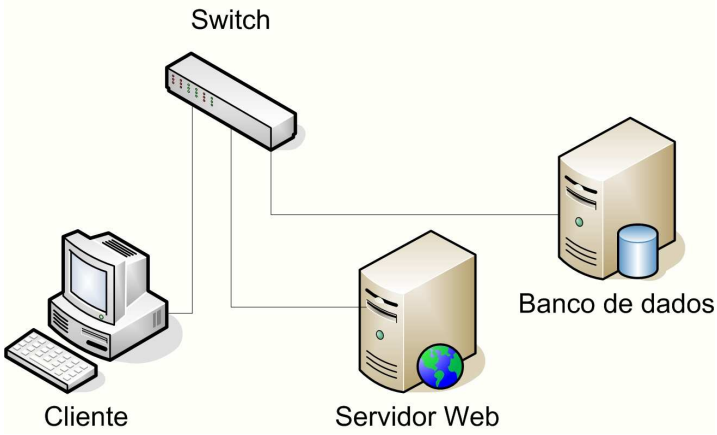

Figura 5.4: Cenário utilizado para a realização de experimentos

Um computador (identificado como Cliente) foi utilizado para gerar requisições de clientes Web, o outro foi configurado apenas como um Servidor Web – para registrar os tempos de atendimento) – e um terceiro foi configurado como um servidor de aplicações, responsável pelas operações de acesso ao Banco de Dados. Todos os computadores foram interligados por uma rede padrão Gigabit Ethernet e possuem processadores AMD Athlon XP com 3.2 GHz de

freqüência, 1 GByte de memória RAM e disco rígido de 7200 rotações por minuto, com capacidade de armazenamento de 80 GBytes.

Para que o Servidor Web Apache registrasse algumas informações importantes para uma análise melhor de componentes específicos do sistema, como tempos médios de consumo de CPU e I/O, foram realizadas, em Sabo (2006), algumas modificações em seu código fonte. Servidores Web, em geral, registram em arquivos texto (*logs*) eventos de sucesso e erros associados às requisições de clientes, sendo que cada registro traz apenas informações de data e hora de atendimento, nome do arquivo, método pelo qual realizou a solicitação, quantidades de bytes enviados e tempo total de atendimento das requisições. Assim, por meio de instrumentação do código fonte, novas funcionalidades foram disponibilizadas no software servidor Web Apache. Uma nova função que coleta o tempo atual de consumo de CPU de uma requisição Web em um processo do servidor em execução foi adicionada, com a utilização da chamada *clock()* do sistema operacional Linux. Na estrutura de uma requisição do Apache existe um campo denominado "*request_time*", designado para armazenar o tempo inicial de atendimento de uma requisição. Dessa forma, utilizando a mesma analogia, foi adicionado um novo campo denominado "*request_time_cpu*" para armazenar o tempo inicial de CPU da requisição.

Uma vez extraído do servidor Web o tempo total de atendimento de uma requisição e o tempo total de consumo de CPU, pode-se calcular o tempo consumido em entrada e saída (I/O). Esse tempo é obtido por meio da equação 5.1, em que: r representa a requisição, $T_{r,total}$ o tempo total de atendimento de uma requisição r, e $T_{r,cpu}$ o tempo total de consumo de CPU (Sabo, 2006):

$$T_{r,io} = T_{r,total} - T_{r,cpu} \quad (5.1)$$

O tempo de entrada e saída inclui o tempo de acesso a disco local e acesso a banco de dados. No caso, como o banco de dados foi instalado em outro computador, o tempo de

entrada e saída inclui também os atrasos de rede.

Os resultados apresentados em Sabo (2006) e que são considerados neste trabalho são mostrados nas Tabelas 5.2 e 5.3. São valores médios alcançados por meio da geração de quantidades diferentes de requisições, com intervalo de 1 segundo entre cada uma.

Na Tabela 5.2 são exibidos os tempos médios das requisições quando considerado o recurso de *cache* no cliente:

	Valores COM CACHE no cliente				
	Tempos de Resposta			**Porcentagens**	
	Total	**CPU**	**I/O**	**CPU**	**I/O**
Principal	0,088253	0,070430	0,017823	79,8046%	20,1954%
Detalhes	0,086097	0,063750	0,022347	74,0442%	25,9558%
Buscas	0,115783	0,094333	0,021450	81,4741%	18,5259%
Categorias	0,096115	0,076333	0,019782	79,4186%	20,5814%
Login	0,046445	0,041331	0,005114	88,9887%	11,0113%
Entrega	0,056348	0,048927	0,007421	86,8292%	13,1708%
Pagamento	0,105471	0,083768	0,021704	79,4220%	20,5780%
Confirmação	0,112506	0,063000	0,049506	55,9971%	44,0029%

Tabela 5.2: Valores obtidos COM o recurso de *cache* no cliente.

A Tabela 5.3 é composta pelos tempos médios das requisições quando desconsiderado o recurso de *cache* no cliente.

Páginas Web dinâmicas, em relação a páginas Web estáticas, têm como diferencial um maior consumo de recursos de CPU e o fato de não terem seus conteúdos armazenados em memória *cache*. Operações de acesso à base de dados são caracterizadas por exigirem maior demanda de dispositivos de I/O.

Como se pode observar nas Tabelas 5.2 e 5.3, o tempo de consumo de CPU das requisições é maior que o tempo de consumo de dispositivos de I/O. Isso se deve não somente pelo fato de a base de dados estar presente em um servidor de aplicações (separado do servidor Web), mas também pela presença do recurso de *cache* do sistema operacional no *host* servidor Web.

	Valores SEM CACHE no cliente				
	Tempos de Resposta			Porcentagens	
	Total	**CPU**	**I/O**	**CPU**	**I/O**
Principal	0,091791	0,072928	0,018863	79,4504%	20,5496%
Detalhes	0,087716	0,073754	0,013962	84,0830%	15,9170%
Buscas	0,121153	0,096474	0,024679	79,6301%	20,3699%
Categorias	0,098332	0,082080	0,016252	83,4725%	16,5275%
Login	0,076757	0,041000	0,035757	53,4153%	46,5847%
Entrega	0,125837	0,083333	0,042504	66,2232%	33,7768%
Pagamento	0,140452	0,072333	0,068118	51,5005%	48,4995%
Confirmação	0,144596	0,058667	0,085929	40,5729%	59,4271%

Tabela 5.3: Valores obtidos SEM o recurso de *cache* no cliente.

Arquivos estáticos, acessados com maior freqüência no servidor Web – imagens, folhas de estilos etc. –, tendem a permanecer na memória *cache* gerenciada pelo sistema operacional que executa o servidor. Dessa forma, ao contrário do que se espera, são resgatados da memória e não de um dispositivo de armazenamento. A ferramenta utilizada para medir o tempo de consumo de CPU classifica acessos à memória como operações de CPU.

Caso o cliente Web utilize *cache* local (opcional e configurável no navegador), objetos que compõem as páginas Web podem ou não ser requisitados para o servidor, o que gera economia de recursos. Como as páginas atendidas pelo servidor são dinâmicas e compostas por pequenos arquivos estáticos embutidos – com tamanho inferior a 5 Kbytes –, essa economia se torna desprezível.

Em requisições da Categoria Navegar, o ganho de desempenho COM ou SEM o uso do recurso de *cache* no cliente se torna desprezível. No entanto, quando se trata de interações do tipo Pedir, não usar o recurso de *cache* no cliente gera um aumento no tempo de I/O da requisição. Isso se deve ao fato de as requisições da categoria Pedir serem criptografadas por meio de conexões SSL (*Secure Socket Layer*). Sem o recurso de *cache* no cliente, a chave de

segurança da respectiva sessão do cliente deverá ser reenviada ao cliente a cada requisição. Dessa forma, o servidor executará novamente o procedimento de *handshaking* – troca de certificados, negociação de compressão e codificação, configuração de identificador de sessão. Esse procedimento é realizado durante o atendimento de uma requisição e pode aumentar em escalas muito maiores o tempo de I/O, conforme a largura de banda da rede que interliga o cliente e o servidor (Goldberg et al., 1998).

5.5 Considerações Finais

Neste capítulo foi proposto um modelo de carga de trabalho Web para sites de comércio eletrônico, especificamente de varejo.

Para a definição da porcentagem das requisições que acessam o servidor Web, foi utilizado o padrão de comportamento apresentado em Menascé et al. (1999). Após montar e resolver o sistema de equações que representa a máquina de estados, chegou-se aos valores apresentados na Tabela 5.1. Nota-se que, para cada sessão atendida pelo sistema, a maioria das requisições é do tipo Categorias e Buscas, pois são os tipos de solicitações que permitem ao usuário a exploração do site em busca do produto desejado. Outro ponto de destaque é o fato de que existe uma probabilidade muito baixa de cada sessão efetivamente gerar uma venda.

Em seguida, foram mostrados e discutidos os tempos médios de consumo de CPU e I/O das requisições Web apresentados em Sabo (2006). Como tal estudo já foi validado e apresentado à comunidade científica, decidiu-se pela utilização dessas informações neste trabalho. Os valores foram apresentados nas Tabelas 5.2 (COM o recurso de *cache* no cliente) e 5.3 (SEM o recurso de *cache* no cliente).

Tendo-se a máquina de estados que descreve o comportamento dos usuários no sistema e o tempo médio de consumo de CPU e I/O de cada uma, é possível gerar a carga de trabalho necessária para alimentar o modelo proposto neste trabalho.

No próximo capítulo, serão apresentadas as estratégias propostas para melhorar o desempenho do SWDS quando são consideradas as características das requisições Web típicas de um site de comércio eletrônico e os resultados dos experimentos.

Capítulo

6

Resultados Obtidos

6.1 Considerações Iniciais

Com o intuito de aprimorar o funcionamento do servidor Web, foram estudadas diversas estratégias de atendimento às requisições. Inicialmente foram analisados os impactos de alterar apenas o modo de funcionamento das filas dos servidores. Em seguida, foram observadas também as vantagens e desvantagens de utilizar mecanismos de controle de admissão ao sistema. Finalmente, foram feitas combinações dessas duas abordagens.

A seguir, são apresentados a metodologia utilizada para a realização dos experimentos e os resultados obtidos com as diversas análises realizadas.

6.2 Metodologia para Experimentos

A simulação foi a técnica escolhida para solucionar o modelo de Rede de Filas do SWDS apresentado na Figura 4.2. Optou-se por essa abordagem por ser bastante flexível e permitir a verificação de diferentes configurações prontamente.

Todas as características necessárias para a geração da carga de trabalho foram consideradas do estudo apresentado no Capítulo 5. As porcentagens de acesso das requisições são definidas

de acordo com a Tabela 5.1. Os tempos de atendimento de CPU e I/O são estabelecidos por distribuições exponenciais com valores médios baseados nas Tabelas 5.2 ou 5.3. A taxa de acerto em *cache* para cada requisição é apresentada na Tabela 6.1:

Acertos em Cache (Cliente)	
Principal	30%
Detalhes	30%
Buscas	30%
Categorias	30%
Login	-
Entrega	100%
Pagamento	100%
Confirmação	100%

Tabela 6.1: Taxas de acerto em *cache* no cliente

A taxa de acerto em *cache* para requisições da Categoria Navegar é fixada em 30%, segundo SPECweb2005 (2005). Já as requisições da Categoria Pedir possuem taxas de acerto diferenciadas. Isso se deve ao fato de as requisições dessa categoria serem criptografadas por meio de conexões SSL (*Secure Socket Layer*) e apresentarem algumas particularidades em relação ao uso do recurso de memória *cache* no cliente.

Neste trabalho, considera-se que o procedimento de *handshaking* – troca de certificados, negociação de compressão e codificação, configuração de identificador de sessão – é executado apenas em requisições do tipo *Login* e que a chave gerada permanece armazenada no *cache* local do cliente. Isso significa que, para todas as requisições do tipo *Login*, serão utilizados os tempo de consumo de CPU e I/O da Tabela 5.3, uma vez que não existe probabilidade de o cliente possuir uma chave de sessão válida que o permita utilizar objetos embutidos armazenados em *cache* local.

Tendo em vista que as requisições dos tipos Entrega, Pagamento e Confirmação não repetirão o procedimento de *handshaking*, atribuem-se a elas os tempos de consumo de CPU e

I/O da Tabela 5.2. Mesmo que, por motivos adversos, após a etapa de *Login*, o conteúdo da memória *cache* do cliente possa ser apagado, assume-se uma taxa de acerto de 100% nessas requisições.

Para dar origem às requisições, foi utilizado o *log* de acesso a servidores do site da Copa do Mundo de 1998 (World Cup, 1998). As simulações são controladas pelo tempo, tendo sido definido em vinte minutos, sendo quatro minutos para o *warm-up*. O *log* traz informações de oito servidores, mas para este trabalho são consideradas as informações de apenas dois deles, sendo suficiente para gerar uma situação de sobrecarga.

Todas as requisições geradas a partir do *log*, quando admitidas pelo sistema, são consideradas requisições da categoria Principal. Sendo atendidas, as respostas são enviadas ao consumidor e, após o tempo de pensar do consumidor (*thinking time*), darão origem a requisições do tipo Categorias ou do tipo Buscas, de acordo com as probabilidades apresentadas na máquina de estados da Figura 5.3. Por sua vez, as requisições do tipo Categorias darão origem a requisições do tipo Buscas, Detalhes ou a outras requisições do tipo Categorias. Essa lógica é utilizada para a geração dos demais tipos de requisições. O tempo de pensar do consumidor foi representado segundo uma distribuição uniforme [4, 8].

A capacidade de processamento do Classificador foi definida como 8000 requisições/seg. A capacidade do Controle de Admissão foi definida como 4000 requisições/seg (Traldi et al., 2006).

Para todos os experimentos, o modelo do SWDS foi configurado como um *cluster* de dez nós (servidores Web). Essa configuração apresentou-se adequada, pois permitiu a manutenção da situação de sobrecarga necessária às análises, mantendo a característica de servidor distribuído.

Todos os resultados apresentados correspondem a uma média de cinco simulações. São apresentados também os intervalos de confiança de 95%.

Para a implementação e solução do modelo por simulação, foi utilizada a biblioteca SimpackJava (Fishwick, 2004), desenvolvida a partir da biblioteca SMPL (MacDougall, 1987).

6.3 Resultados dos Experimentos

Os resultados obtidos para as diversas estratégias de atendimento implementadas no modelo SWDS são apresentados a seguir.

6.3.1 Algoritmo *Round-Robin* (RR)

O primeiro experimento realizado utilizou o algoritmo de escalonamento *Round-Robin* para distribuir as requisições entre os nós do cluster do SWDS. Esse algoritmo atribui as requisições aos servidores do *cluster* segundo um esquema de rodízio.

Não foi implementado nenhum mecanismo de controle de admissão no modelo. Nas filas dos nós servidores, foi mantida a tradicional disciplina de atendimento *FCFS (First-Come First-Served),* que atende as requisições de acordo com a ordem de chegada, independentemente da sua classe de prioridade.

Esse experimento foi realizado com o intuito de analisar o comportamento do SWDS em situações de sobrecarga, quando nenhum mecanismo de diferenciação de serviços é utilizado.

Pode-se observar, conforme dados apresentados na Tabela 6.2, que todas as requisições foram admitidas pelo sistema e receberam o mesmo tipo de atendimento, sem diferenciação. A utilização dos recursos de CPU do sistema atinge 100%, uma vez que as requisições são do tipo *CPU-Bound*, conforme Tabelas 5.2 e 5.3. Com a sobrecarga, o tempo médio de resposta às requisições ficou muito elevado, com média de 305,90 s. Além disso, apenas 57,03% das requisições admitidas pelo sistema foram concluídas.

Para auxiliar no critério de avaliação das políticas de atendimento propostas nesta dissertação, foram elaborados dois novos índices: eficiência e rejeição.

- Índice de Eficiência: é obtido pela razão entre a quantidade de requisições concluídas da classe *Confirmação* (C) e a quantidade total de requisições feitas ao sistema (U). Ele mostra, do total de requisições feitas ao servidor, qual é o percentual de requisições concluídas de efetivação de venda.

- Índice de Rejeição: é obtido pela razão entre a parcela de requisições não admitidas pelo sistema (U - A) e a quantidade total de requisições (U). Esse indicador mostra a proporção de requisições rejeitadas pelo servidor.

Esses indicadores são apresentados na parte inferior das tabelas.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
224,85	**0,41**	**128,23**	**0,03**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	97.063,20	319,66	97.063,20	319,66	100,00
Categorias	52.874,60	190,07	52.874,60	190,07	100,00
Buscas	53.045,80	206,00	53.045,80	206,00	100,00
Detalhes	12.124,20	74,19	12.124,20	74,19	100,00
Login	547,20	16,78	547,20	16,78	100,00
Entrega	108,40	15,10	108,40	15,10	100,00
Pagamento	58,80	12,05	58,80	12,05	100,00
Confirmação	29,20	7,21	29,20	7,21	100,00
Total	**215.851,40**	**394,30**	**215.851,40**	**394,30**	**100,00**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	55.326,00	83,65	57,00	305,95	0,42
Categorias	30.238,00	83,28	57,19	305,73	1,22
Buscas	30.145,00	85,99	56,83	306,07	1,28
Detalhes	6.951,40	63,23	57,33	305,38	2,03
Login	326,00	14,16	59,58	308,36	7,53
Entrega	61,80	11,66	57,01	311,64	17,72
Pagamento	31,20	6,11	53,06	288,55	19,39
Confirmação	19,20	4,06	65,75	308,79	25,16
Total	**123.098,60**	**29,48**	**57,03**	**305,90**	**0,51**

Eficiência (C / U)	0,009%	Rejeição (U - A) / U	0,00%

Tabela 6.2: Algoritmo *Round-Robin*

Para esse experimento, o Índice de Eficiência é muito baixo (0,009%), indicando que, apesar do sistema estar sobrecarregado, conclui poucas vendas.

O Índice de Rejeição manteve-se zerado, pois não foi implantado nenhum mecanismo de controle de admissão. Assim, o SWDS foi mantido em situação de sobrecarga, fazendo com que o tempo de resposta às requisições fosse crescendo ao longo da simulação, conforme gráfico da Figura 6.1.

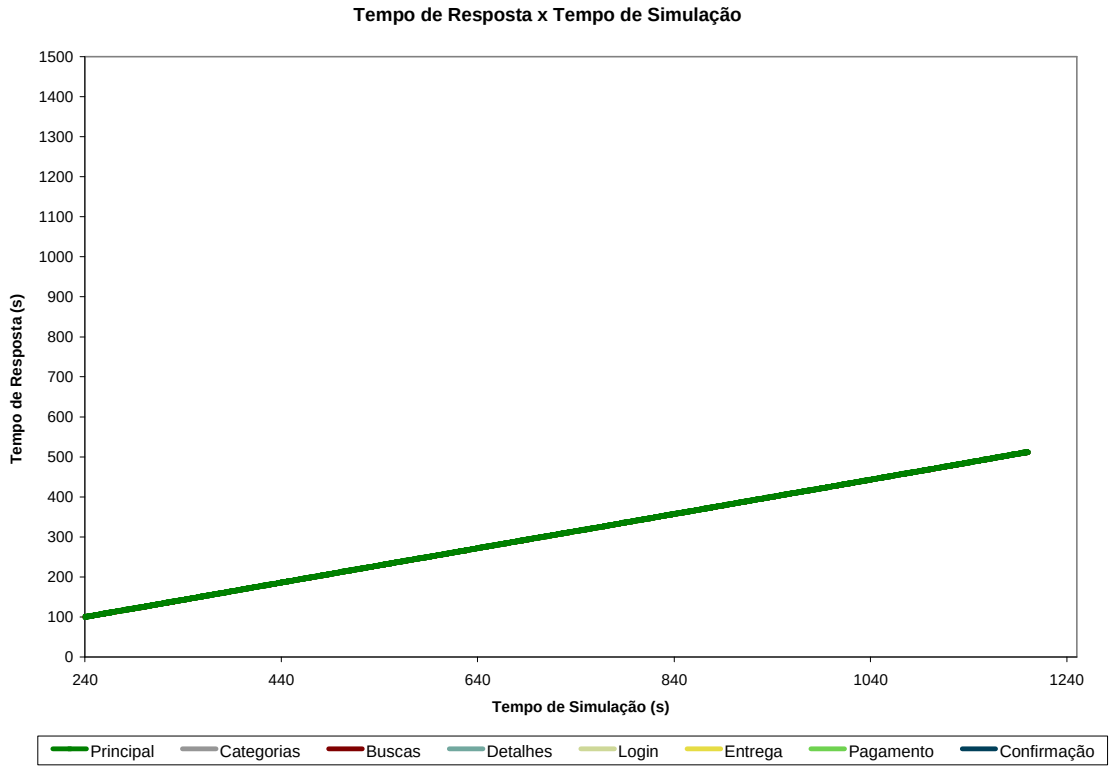

Figura 6.1: Algoritmo *Round-Robin*

O baixo Índice de Eficiência obtido nesse experimento e o elevado tempo médio de resposta às requisições evidenciam a necessidade de outros mecanismos de atendimento.

6.3.2 Algoritmo *Weighted Fair Queuing* (WFQ)

Com o objetivo de melhorar a utilização dos recursos do servidor SWDS, de acordo com os interesses geralmente envolvidos em um negócio do tipo *e-commerce*, foi inserido no modelo o algoritmo de diferenciação de serviços *Weighted Fair Queuing (WFQ)* (Traldi et al.,

2006). Esse algoritmo é baseado em *Prioridades* e utiliza a idéia de dividir a capacidade de processamento dos nós do *cluster* entre as classes de requisições de acordo com os pesos que são atribuídos às classes. Como parâmetro para o escalonamento, o algoritmo também considera o tempo de processamento necessário para atender a uma requisição, priorizando requisições menos custosas (Traldi et al., 2006). De uma maneira simplista, se forem atribuídos pesos 2 e 1 às classes A e B, respectivamente, esperar-se-á que a cada três requisições (de mesmo custo) atendidas pelo servidor, duas sejam da classe A e uma seja da classe B.

Uma vez inserido o WFQ no modelo SWDS, a próxima etapa para definir os parâmetros para a realização do experimento foi determinar os pesos a serem utilizados pelo algoritmo, de acordo com os interesses do negócio, para as diversas classes de requisições existentes: Principal, Categorias, Buscas, Detalhes, *Login*, Entrega, Pagamento e Confirmação.

Para a determinação dos pesos a serem utilizados, foi proposto um modelo de Otimização Linear que busca maximizar o número de requisições atendidas pelo servidor em um intervalo de tempo. O modelo proposto é apresentado a seguir:

Maximizar: $A + B + C + D + E + F + G + H$ (6.1)

Sujeito às restrições:

0,072A	+0,071B	+0,096C	+0,080D	+0,041E	+0,049F	+0,084G	+0,063H	<=	1
-0,5A	+0,605B	-0,345C	-0,415D	-0,203E				<=	0
-0,5A	-0,345B	+0,605C	-0,415D	-0,203E				<=	0
	-0,2B	-0,2C	+D	-0,190E				<=	0
			-0,075D	+0,945E				<=	0
				-0,333E	+F			<=	0
					-0,95F	+G		<=	0
						-0,95G	+H	<=	0

Maximizar a equação 6.1 significa maximizar a quantidade de requisições atendidas pelo servidor. No entanto, devem ser satisfeitas todas as restrições apresentadas no modelo.

A primeira restrição é responsável por otimizar o número de requisições que o servidor atende em um determinado intervalo de tempo, uma vez que os coeficientes dessa equação de restrição são os tempos médios de CPU das requisições de cada tipo, levando-se em conta as taxas de acerto em *cache* apresentadas na Tabela 6.1. Utilizou-se o tempo de CPU por se tratar de requisições *CPU-Bound*. Esses coeficientes são apresentados na Tabela 6.3:

A = Principal	0,072179 s
B = Categorias	0,070753 s
C = Buscas	0,095832 s
D = Detalhes	0,080356 s
E = *Login*	0,041000 s
F = Entrega	0,048927 s
G = Pagamento	0,083768 s
H = Confirmação	0,063000 s

Tabela 6.3: Tempos médios de CPU - considerando acertos em *cache*

As demais equações de restrição garantem a consistência da carga de trabalho com relação à máquina de estados da Figura 5.3, uma vez que, mudando-se a proporção de requisições de um determinado tipo, altera-se a proporção dos demais tipos também.

Com o auxílio da ferramenta Solver (Excel, 2003), foi solucionado o modelo de Otimização Linear pelo método simplex (Dantzig, 1951). O resultado é apresentado na Tabela 6.4:

Tipo de Requisição	Peso	Peso Relativo
A = Principal	7,654	54,75%
B = Categorias	6,326	45,25%
C = Buscas	0,000	0,00%
D = Detalhes	0,000	0,00%
E = *Login*	0,000	0,00%
F = Entrega	0,000	0,00%
G = Pagamento	0,000	0,00%
H = Confirmação	0,000	0,00%

Tabela 6.4: Pesos obtidos para o WFQ (maximizando atendimentos)

Os pesos apresentados na Tabela 6.4 foram utilizados no algoritmo WFQ inserido no

SWDS. Como o custo das requisições já foi considerado no modelo de otimização, o algoritmo WFQ foi configurado para desprezar essa informação no critério de diferenciação.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
236,84	**0,05**	**139,77**	**0,01**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	96.794,80	39,83	96.794,80	39,83	100,00
Categorias	60.640,20	94,13	60.640,20	94,13	100,00
Buscas	57.861,60	220,30	57.861,60	220,30	100,00
Detalhes	12.060,20	143,22	12.060,20	143,22	100,00
Login	4,20	4,34	4,20	4,34	100,00
Entrega	1,40	2,42	1,40	2,42	100,00
Pagamento	0,40	1,11	0,40	1,11	100,00
Confirmação	-	-	-	-	-
Total	**227.362,80**	**45,03**	**227.362,80**	**45,03**	**100,00**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	73.978,60	82,72	76,43	166,44	8,12
Categorias	60.167,80	88,55	99,22	40,12	18,17
Buscas	18,40	5,59	0,03	455,46	80,33
Detalhes	11,00	1,24	0,09	747,22	130,76
Login	2,40	2,86	57,14	660,47	210,18
Entrega	0,60	1,11	42,86	709,12	130,15
Pagamento	-	-	-	-	-
Confirmação	-	-	-	-	-
Total	**134.178,80**	**6,29**	**59,02**	**109,89**	**9,49**

Eficiência (C / U)	0,000%	**Rejeição (U - A) / U**	0,00%

Tabela 6.5: Algoritmo WFQ (maximizando atendimentos)

Conforme resultados apresentados na Tabela 6.5, foram atendidas mais requisições pelo servidor SWDS, um aumento de 9% em relação ao algoritmo RR. O *throughput* evoluiu de 128,23 req/s para 139,77 req/s. Isso ocorreu porque foram atendidas poucas requisições com maior custo de processamento ao sistema – Buscas, Detalhes e Pagamentos. Assim, mais requisições puderam ser atendidas.

No entanto, é evidente que para um negócio de *e-commerce* o objetivo principal não é atender o maior número de requisições web, mas sim gerar o maior número de vendas. Dessa forma, a função objetivo do modelo de otimização linear foi ajustada para maximizar o número de conclusões de processos de venda. O novo modelo é apresentado a seguir:

Maximizar: 0 A + 0 B + 0 C + 0 D + 0 E + 0 F + 0 G + H (6.2)

Sujeito às restrições:

0,072A	+0,071B	+0,096C	+0,080D	+0,041E	+0,049F	+0,084G	+0,063H	<=	1
-0,5A	+0,605B	-0,345C	-0,415D	-0,203E				<=	0
-0,5A	-0,345B	+0,605C	-0,415D	-0,203E				<=	0
	-0,2B	-0,2C	+D	-0,190E				<=	0
			-0,075D	+0,945E				<=	0
				-0,333E	+F			<=	0
					-0,95F	+G		<=	0
						-0,95G	+H	<=	0

A equação a ser maximizada foi determinada levando-se em consideração que o processo de venda é concluído com as requisições do tipo Confirmação. Assim, maximizar a equação 6.2 significa maximizar a probabilidade de gerar vendas pelo servidor.

O modelo foi solucionado, e o novo resultado é apresentado na Tabela 6.6:

Tipo de Requisição	Peso	Peso Relativo
A = Principal	0,791	6,44%
B = Categorias	4,656	37,90%
C = Buscas	4,656	37,90%
D = Detalhes	1,891	15,39%
E = *Login*	0,150	1,22%
F = Entrega	0,050	0,41%
G = Pagamento	0,047	0,39%
H = Confirmação	0,045	0,37%

Tabela 6.6: Pesos obtidos para o WFQ (maximizando vendas)

A solução do modelo de otimização indica que, para maximizar o número de vendas pelo sistema, deve ser priorizado o atendimento a todas as sessões cujo atendimento tenha sido iniciado ou, então, ao se admitir uma nova sessão no sistema, deve-se garantir o atendimento às requisições subseqüentes, pertencentes à mesma sessão. Isso fica claro ao comparar as Tabela 5.1 e 6.6.

Os pesos apresentados na Tabela 6.6 foram utilizados no algoritmo WFQ e os resultados obtidos são apresentados na Tabela 6.7.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
216,20	**0,19**	**122,81**	**0,06**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	97.147,60	124,75	97.147,60	124,75	100,00
Categorias	44.798,00	183,38	44.798,00	183,38	100,00
Buscas	44.916,00	201,51	44.916,00	201,51	100,00
Detalhes	18.103,40	90,10	18.103,40	90,10	100,00
Login	1.400,60	19,05	1.400,60	19,05	100,00
Entrega	452,80	6,77	452,80	6,77	100,00
Pagamento	392,40	9,56	392,40	9,56	100,00
Confirmação	345,40	15,72	345,40	15,72	100,00
Total	**207.556,20**	**185,63**	**207.556,20**	**185,63**	**100,00**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	8.242,20	71,29	8,48	598,98	11,12
Categorias	44.591,40	191,10	99,54	40,54	11,81
Buscas	44.672,60	171,05	99,46	42,22	19,25
Detalhes	17.942,80	74,87	99,11	61,05	11,38
Login	1.338,60	12,74	95,57	78,52	10,84
Entrega	416,40	11,70	91,96	97,61	30,78
Pagamento	364,20	17,89	92,81	75,88	27,10
Confirmação	329,00	13,05	95,25	55,64	21,47
Total	**117.897,20**	**60,36**	**56,80**	**84,11**	**8,27**

Eficiência (C / U)	0,159%	Rejeição (U - A) / U	0,00%

Tabela 6.7: Algoritmo WFQ (maximizando vendas)

Comparando-se os resultados aos obtidos com o algoritmo RR, alguns pontos destacam-se:

- O número de vendas concluídas melhorou 1614%, passando de 19,2 para 329 pedidos. Isso elevou o Índice de Eficiência de 0,009% para 0,159%;
- O tempo médio de resposta às requisições passou de 305,9s para 84,11s. Uma queda de 72,5%. No entanto, as requisições do tipo Principal são extremamente penalizadas, uma vez que o WFQ atende menos sessões, mas garante o atendimento das iniciadas até a conclusão da venda;
- O *throughput* diminuiu de 128,23 req/s para 122,81 req/s. Uma queda de 4,2%. Isso ocorreu dada a nova distribuição de requisições atendidas pelo sistema. São

atendidas mais requisições com maior custo de processamento, como dos tipos Buscas e Detalhes.

O Índice de Rejeição manteve-se zerado, pois não foi considerado nenhum mecanismo de controle de admissão, ou seja, todas as requisições são admitidas pelo sistema.

Como o WFQ inicia o atendimento a novas sessões apenas quando pode garantir o atendimento a todos os seus pedidos subseqüentes, as requisições da categoria Principal são penalizadas. Isso faz com que o tempo de resposta a essas requisições cresça consideravelmente ao longo da simulação. Para os demais tipos de requisições, pode-se considerar que o tempo cresce de maneira equivalente. Isso pode ser observado no gráfico da Figura 6.2:

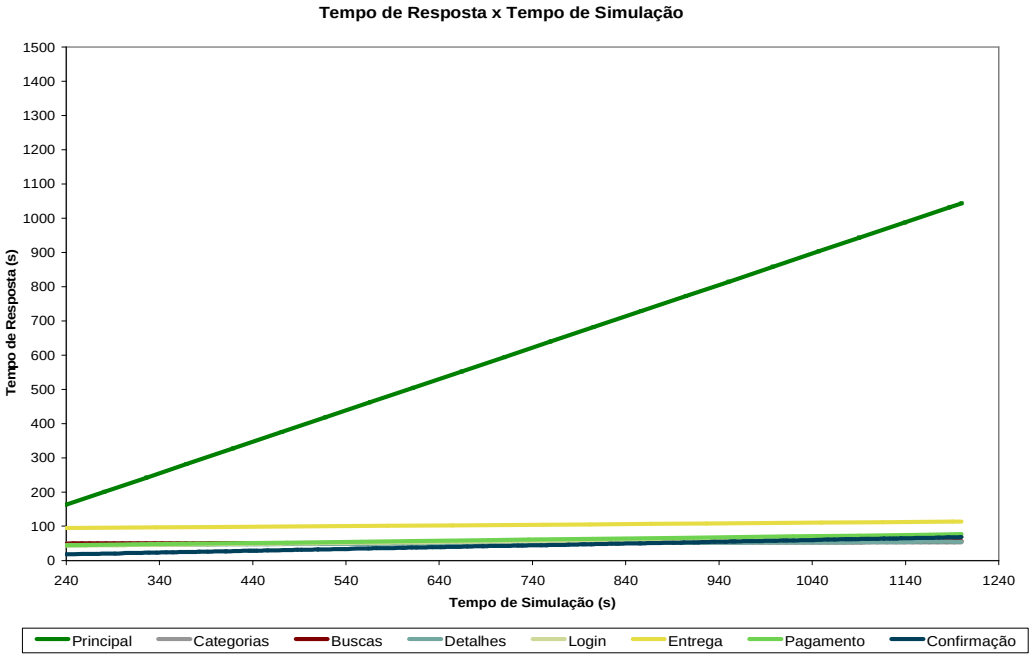

Figura 6.2: Algoritmo WFQ

Apesar de todas as melhorias obtidas com a inserção do algoritmo WFQ no modelo, outros mecanismos de diferenciação de serviços ainda são necessários para o SWDS. Essa necessidade fica clara quando é analisado o tempo médio de resposta às requisições que, mesmo com a redução, ainda permanece elevado. Assim, em seguida são apresentados os

resultados dos experimentos com a inserção de mecanismos de controle de admissão no modelo SWDS.

6.3.3 Algoritmo RR com Controle de Admissão por Tamanho de Fila

Com o objetivo de reduzir o tempo médio de resposta às requisições, foi inserido, no modelo SWDS, o mecanismo de Controle de Admissão (CA) por tamanho de fila. O módulo CA recebe a informação do tamanho médio das filas dos nós do *cluster* de servidores Web a cada requisição que chega ao servidor. Se o tamanho médio exceder o valor definido pelo administrador do sistema, a requisição será rejeitada pelo SWDS. Para os experimentos a seguir, o tamanho máximo médio das filas foi definido em 1024 requisições.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
225,41	**0,19**	**128,62**	**0,04**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	97.049,60	174,50	58.602,00	87,18	60,38
Categorias	53.467,00	56,55	29.042,40	108,74	54,32
Buscas	53.502,40	88,41	29.122,80	66,75	54,43
Detalhes	11.710,80	76,63	6.341,80	82,18	54,15
Login	505,00	30,33	279,20	29,92	55,29
Entrega	89,40	21,41	47,20	13,78	52,80
Pagamento	44,80	13,64	24,20	9,35	54,02
Confirmação	22,40	7,68	12,00	6,02	53,57
Total	**216.391,40**	**182,82**	**123.471,60**	**40,57**	**57,06**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	58.621,60	131,13	100,03	79,78	0,02
Categorias	29.086,40	117,81	100,15	79,77	0,02
Buscas	29.063,80	91,11	99,80	79,81	0,03
Detalhes	6.339,60	55,93	99,97	79,78	0,03
Login	275,80	33,74	98,78	79,78	0,07
Entrega	47,80	14,94	101,27	79,66	0,14
Pagamento	23,80	9,01	98,35	79,74	0,14
Confirmação	12,00	3,93	100,00	79,72	0,29
Total	**123.470,80**	**41,06**	**100,00**	**79,79**	**0,02**

Eficiência (C / U)	0,006%	Rejeição (U - A) / U	42,94%

Tabela 6.8: Algoritmo RR com CA por tamanho de fila

Conforme informações da Tabela 6.8, observa-se que o tempo médio de resposta foi

reduzido com sucesso, caindo de 305,9s (algoritmo RR sem o uso do Controle de Admissão) para 79,79s. De acordo com o gráfico da Figura 6.3, esse tempo médio de resposta se mantém constante ao longo da simulação, uma característica bastante interessante do uso do CA.

No entanto, o Índice de Rejeição foi de 42,94%, indicando que uma parcela representativa das requisições não foi admitida pelo sistema. Isso é necessário para que a situação de sobrecarga não leve a um aumento progressivo no tempo médio de resposta, inviabilizando o uso do sistema.

O Índice de Eficiência manteve-se muito baixo, em 0,006%, dado que não foi utilizado nenhum tipo de controle que focasse na conclusão dos processos de venda. O CA descarta as requisições Web, independentemente do tipo, quando o tamanho médio das filas dos nós do *cluster* de servidores Web atinge o máximo definido, não priorizando requisições com maiores probabilidades de gerar venda. A política de atendimento das filas também não oferece nenhum tipo de privilégio às requisições, pois segue a disciplina *FCFS*.

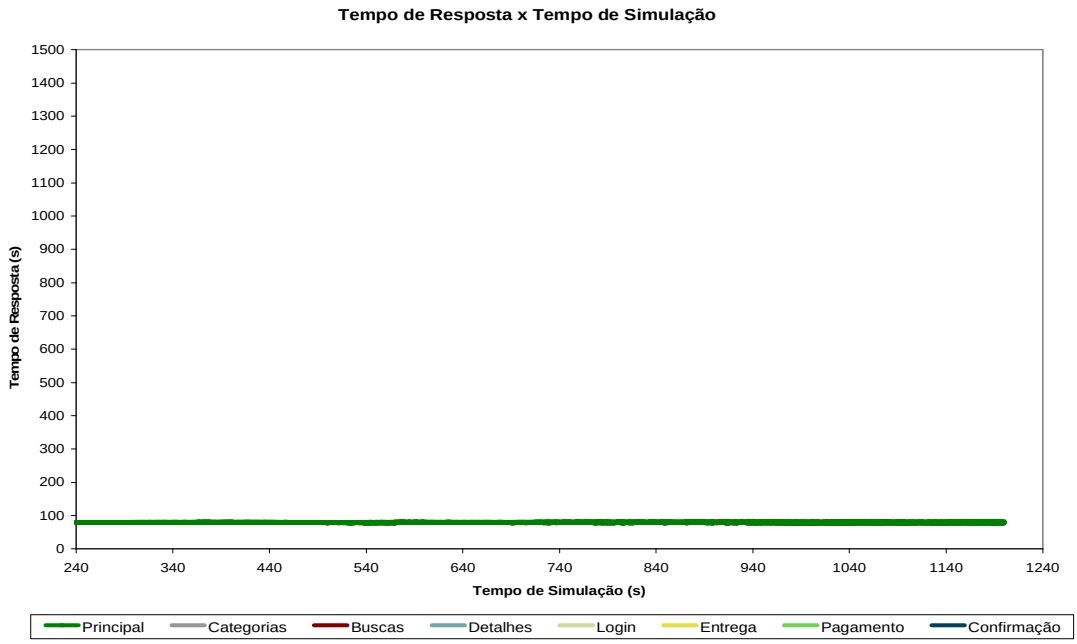

Figura 6.3: Algoritmo RR com CA por tamanho de fila

Assim, visando melhorar o Índice de Eficiência do SWDS, foi inserido no módulo CA um

mecanismo de descarte seletivo de requisições. Quando as filas dos servidores Web atingem 95% do tamanho máximo permitido, é dado início a um processo de descarte que leva em consideração o tipo da requisição que chega ao sistema.

Para a definição do critério de descarte, foram calculadas as probabilidades que cada tipo de requisição tem de gerar uma venda. Esses dados foram obtidos dadas as probabilidades de transição de um estado para outro da máquina de estados da Figura 5.3. A seguir é apresentado o sistema de equações que representa essas probabilidades:

$$
\begin{array}{lrrrrrrrrr}
A & -0{,}5B & -0{,}5C & & & & & & = & 0 \\
 & +0{,}605B & -0{,}345C & -0{,}2D & & & & & = & 0 \\
 & -0{,}345B & +0{,}605C & -0{,}2D & & & & & = & 0 \\
 & -0{,}415B & -0{,}415C & +D & -0{,}075E & & & & = & 0 \\
 & -0{,}203B & -0{,}203C & -0{,}19D & +0{,}945E & -0{,}333F & & & = & 0 \\
 & & & & & +F & -0{,}95G & & = & 0 \\
 & & & & & & +G & -0{,}95H & = & 0 \\
 & & & & & & & +H & = & 1
\end{array}
$$

Resolvendo-se o sistema de equações, obtêm-se os valores apresentados na Tabela 6.9.

A = Principal	5,7%
B = Categorias	5,7%
C = Buscas	5,7%
D = Detalhes	7,4%
E = *Login*	35,7%
F = Entrega	90,3%
G = Pagamento	95%
H = Confirmação	100%

Tabela 6.9: Probabilidade de cada estado gerar venda

Analisando a tabela, observou-se que, como não há desistência por parte dos compradores entre a movimentação do estado Principal aos estados Categorias ou Buscas, a probabilidade desses estados gerarem venda é a mesma. No entanto, para a definição do critério de descarte, foi considerado que os clientes nos estados Categorias ou Buscas estão um passo a frente dos clientes que estão no estado Principal no processo de compra. Dessa forma, ao atingir 95% do tamanho das filas dos servidores, o sistema inicia o processo de descarte de requisições do tipo Principal apenas. Se esse procedimento não for suficiente para conter a sobrecarga do

sistema, ao atingir 100% do tamanho limite para as filas, o sistema descarta as requisições sem levar em consideração o tipo.

Conforme informações da Tabela 6.10, a inserção do mecanismo de admissão seletiva no SWDS mostrou-se promissor. O tempo médio de resposta e o Índice de Rejeição variaram muito pouco em relação ao mecanismo de CA por tamanho de fila comum. Entretanto, houve uma melhora expressiva do Índice de Eficiência, que evoluiu de 0,006% para 0,158%. Uma melhora de 0,152 pp. Por outro lado, houve uma sensível queda no *throughput*, gerando uma diminuição de 4,6% na quantidade de requisições atendidas pelo SWDS. Isso ocorreu dada a nova distribuição de requisições atendidas pelo sistema. São atendidas mais requisições com maior custo de processamento, como dos tipos Buscas e Detalhes.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
215,82	**0,36**	**122,72**	**0,05**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	96.855,20	361,02	7.481,60	46,06	7,72
Categorias	44.686,00	188,24	44.686,00	188,24	100,00
Buscas	44.720,40	113,24	44.720,40	113,24	100,00
Detalhes	18.262,80	65,06	18.262,80	65,06	100,00
Login	1.430,80	74,58	1.430,80	74,58	100,00
Entrega	458,40	25,29	458,40	25,29	100,00
Pagamento	411,00	19,69	411,00	19,69	100,00
Confirmação	362,80	22,03	362,80	22,03	100,00
Total	**207.187,40**	**343,12**	**117.813,80**	**51,99**	**56,86**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	7.292,20	40,53	97,47	79,39	0,03
Categorias	44.869,20	234,97	100,41	79,45	0,03
Buscas	44.943,40	127,92	100,50	79,48	0,03
Detalhes	18.188,60	33,42	99,59	79,46	0,03
Login	1.369,60	64,55	95,72	79,38	0,03
Entrega	434,20	22,46	94,72	79,35	0,09
Pagamento	384,40	18,84	93,53	79,41	0,03
Confirmação	326,60	19,66	90,02	79,40	0,06
Total	**117.808,20**	**45,73**	**100,00**	**79,46**	**0,03**

Eficiência (C / U)	0,158%	**Rejeição (U - A) / U**	43,14%

Tabela 6.10: Algoritmo RR com CA seletivo

Para conter a sobrecarga, o sistema precisou descartar apenas requisições do tipo Principal. Foram admitidas 7,72% das sessões que solicitaram atendimento ao servidor. No entanto, não foi explorado todo o potencial de venda das sessões aceitas, pois não são atendidas 100% das requisições subseqüentes (que possuem maiores probabilidades de gerar venda).

Apesar de funcionarem de maneiras diferentes, há uma grande semelhança entre os resultados obtidos com a inserção do mecanismo de admissão seletiva e os resultados obtidos com a inserção do algoritmo WFQ. Isso ocorre dado que os dois métodos possuem o mesmo objetivo: garantir que todas as sessões iniciadas sejam atendidas até o final pelo sistema. O WFQ busca seu objetivo alterando o funcionamento das filas dos servidores, priorizando ou penalizando requisições, enquanto que o mecanismo de admissão seletiva opera no módulo CA, aceitando ou rejeitando as requisições que chegam ao sistema.

6.3.4 Algoritmo WFQ com Controle de Admissão por Tamanho de Fila

Com o intuito de aprimorar o atendimento do SWDS, foi alterada a política de atendimento utilizada nas filas (*FCFS),* que não leva em consideração nenhum tipo de diferenciação de serviços, com a inserção do algoritmo WFQ. Os critérios para o CA foram mantidos, inclusive o mecanismo de admissão seletiva e o tamanho das filas em 1024 posições.

Como o mecanismo de admissão seletiva já trabalha com o objetivo de aprimorar o desempenho do SWDS descartando novas sessões em caso de sobrecarga, a fim de garantir o atendimento completo das sessões já aceitas pelo sistema, a definição dos pesos a serem utilizados para o WFQ nesse experimento seguiu nova abordagem.

Foram atribuídos maiores pesos às requisições com maiores probabilidades de gerar venda, conforme dados da Tabela 6.9. Mais uma vez, considerou-se que os clientes nos estados Categorias ou Buscas estão um passo a frente dos clientes que estão no estado Principal, no processo de compra. Assim, os pesos foram definidos aumentando-se progressivamente em

5% a prioridade das requisições com maiores probabilidades de gerar venda. Os pesos são apresentados na Tabela 6.11.

	Probabilidades de venda	**Pesos**
A = Principal	5,7%	1,00
B = Categorias	5,7%	1,05
C = Buscas	5,7%	1,05
D = Detalhes	7,4%	1,10
E = *Login*	35,7%	1,15
F = Entrega	90,3%	1,20
G = Pagamento	95%	1,25
H = Confirmação	100%	1,30

Tabela 6.11: Pesos para o WFQ com CA seletivo

De acordo com as informações da Tabela 6.12, a inclusão do algoritmo WFQ trouxe benefícios ao sistema.

Os indicadores de *throughput*, rejeição e tempo médio de resposta permaneceram praticamente inalterados (sensível melhora) se comparados aos dados do experimento com a utilização do algoritmo RR. No entanto, houve uma importante evolução no Índice de Eficiência, que passou de 0,158% para 0,205%. Isso ocorreu devido ao avanço na quantidade de conclusões de vendas, onde o aumento foi de 29,9%.

Nessa configuração, o sistema prioriza as requisições com maiores probabilidades de gerar venda e, dessa forma, impulsiona as sessões que estão em estágios mais avançados. Em decorrência disso, o tempo médio de resposta das requisições com maiores probabilidades de gerar venda são menores.

O sistema precisou descartar apenas requisições do tipo Principal para conter a sobrecarga, conforme dados da tabela apresentada. Foram admitidas ao sistema 7,75% das sessões que solicitaram atendimento ao servidor e, praticamente 100% das requisições subseqüentes, foram concluídas. Essa é a vantagem desse mecanismo em relação ao mecanismo que utiliza o

algoritmo RR.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
215,98	**0,28**	**122,82**	**0,10**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	96.949,60	269,47	7.509,60	41,19	7,75
Categorias	44.662,20	111,59	44.662,20	111,59	100,00
Buscas	44.749,20	257,62	44.749,20	257,62	100,00
Detalhes	18.211,60	167,98	18.211,60	167,98	100,00
Login	1.422,20	39,81	1.422,20	39,81	100,00
Entrega	473,20	32,07	473,20	32,07	100,00
Pagamento	450,40	32,30	450,40	32,30	100,00
Confirmação	423,60	25,15	423,60	25,15	100,00
Total	**207.342,00**	**266,58**	**117.902,00**	**93,91**	**56,86**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	7.525,60	41,80	100,21	161,58	2,64
Categorias	44.654,40	110,48	99,98	77,99	0,79
Buscas	44.740,40	258,43	99,98	77,35	0,86
Detalhes	18.208,80	169,84	99,98	60,73	1,13
Login	1.424,80	38,21	100,18	43,62	5,66
Entrega	474,20	32,69	100,21	26,11	9,68
Pagamento	450,40	32,30	100,00	6,56	4,36
Confirmação	424,20	24,43	100,14	4,14	3,21
Total	**117.902,80**	**92,49**	**100,00**	**79,25**	**0,07**

Eficiência (C / U)	0,205%	**Rejeição (U - A) / U**	43,14%

Tabela 6.12: Algoritmo WFQ com CA seletivo

O gráfico da Figura 6.4 ilustra o funcionamento desse mecanismo apresentando a evolução dos tempos de resposta às requisições ao longo do tempo de simulação. Ao contrário do que ocorre no algoritmo RR, quando todos os tipos de requisições recebem o mesmo tratamento, nesse caso são privilegiadas as requisições em estágios mais avançados no processo de compra.

Com o objetivo de se reduzir o tempo médio de resposta às requisições, esse experimento foi repetido alterando-se, no Controle de Admissão, o tamanho médio das filas para 300 posições.

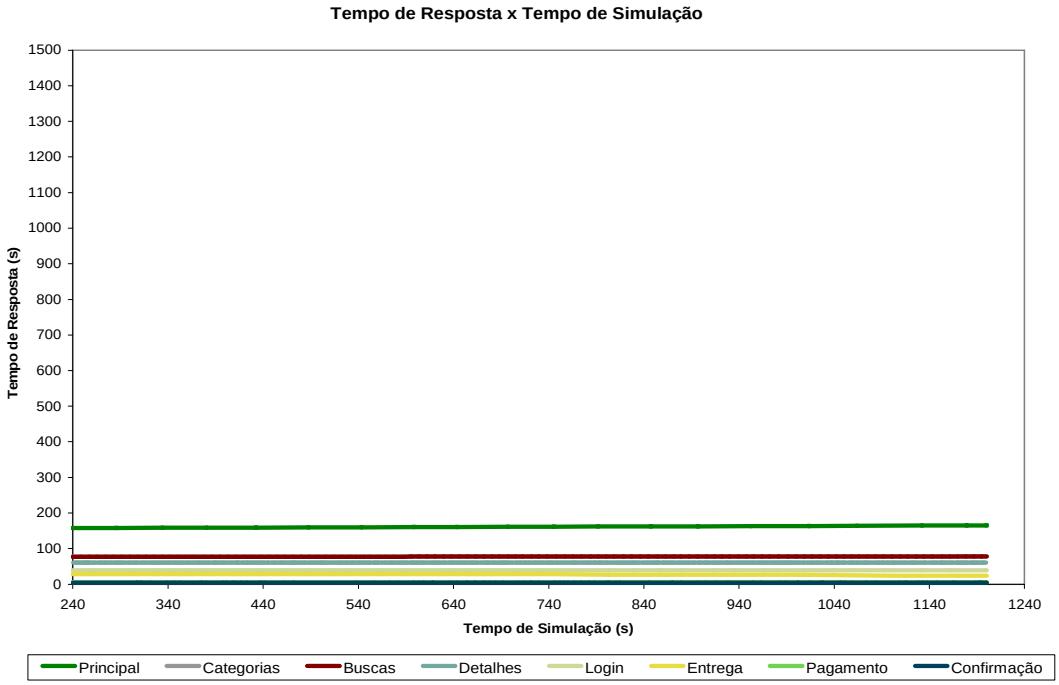

Figura 6.4: Algoritmo WFQ com CA por tamanho de fila e admissão seletiva

Conforme dados apresentados na Tabela 6.13, o tempo médio de resposta foi reduzido significativamente para 23,42s, enquanto que os demais indicadores permaneceram praticamente inalterados. O tempo médio de resposta das requisições com maiores probabilidades de gerar venda são menores, o que mostra também que o comportamento do sistema foi mantido.

6.4 Considerações Finais

Neste capítulo foram propostos mecanismos de diferenciação de serviços e controle de admissão que consideram as características das requisições web para melhorar o desempenho do SWDS. Esses mecanismos, por meio de modificações nas disciplinas de atendimento das filas dos servidores e critérios de prevenção de sobrecarga, buscam benefícios ao sistema, de acordo com os objetivos estabelecidos. Foi apresentado também um modelo de otimização linear que auxilia na parametrização do algoritmo de diferenciação de serviços WFQ.

Taxa Chegada (req/s)	IC (95%)	Throughput (req/s)	IC (95%)	Util. CPU (%)	IC (95%)
216,07	**0,26**	**122,88**	**0,11**	**100,00**	**0,00**

Requisições	Chegadas (n)	IC (95%)	Admissões (n)	IC (95%)	Admissões (%)
Principal	97.071,00	166,64	7.609,60	68,25	7,84
Categorias	44.681,40	129,04	44.681,40	129,04	100,00
Buscas	44.644,80	276,22	44.644,80	276,22	100,00
Detalhes	18.189,60	137,34	18.189,60	137,34	100,00
Login	1.462,60	70,04	1.462,60	70,04	100,00
Entrega	481,60	31,20	481,60	31,20	100,00
Pagamento	459,40	35,94	459,40	35,94	100,00
Confirmação	435,40	40,79	435,40	40,79	100,00
Total	**207.425,80**	**248,61**	**117.964,40**	**100,71**	**56,87**

Requisições	Términos (n)	IC (95%)	Términos (%)	Tempo Med Resp (s)	IC (95%)
Principal	7.618,60	66,83	100,12	34,83	2,78
Categorias	44.676,60	128,02	99,99	23,92	2,20
Buscas	44.640,00	278,26	99,99	23,62	2,58
Detalhes	18.187,00	137,65	99,99	18,86	10,66
Login	1.464,20	69,01	100,11	13,35	9,25
Entrega	484,20	33,03	100,54	12,54	8,78
Pagamento	460,20	38,92	100,17	11,18	8,57
Confirmação	435,80	41,06	100,09	4,56	5,25
Total	**117.966,60**	**106,74**	**100,00**	**23,42**	**0,02**

Eficiência (C / U)	0,210%	**Rejeição (U - A) / U**	43,13%

Tabela 6.13: Algoritmo WFQ com CA Seletivo (reduzindo fila)

Diversas simulações foram realizadas e os resultados obtidos mostram que a exploração das características das requisições web, além de ser fundamental para um bom entendimento do comportamento do servidor, possibilita uma melhoria significativa de performance do sistema.

No próximo capítulo é apresentada uma visão geral do trabalho desenvolvido. Em seguida, são descritas as conclusões, destacando-se os principais resultados e contribuições desta dissertação.

Capítulo

7

Conclusões

7.1 Visão Geral

O objetivo central da pesquisa desenvolvida foi avaliar a possibilidade de se melhorar o desempenho do modelo de Servidor Web com Diferenciação de Serviços (SWDS), quando são consideradas as características das requisições Web nas políticas de atendimento.

Durante o desenvolvimento do trabalho, optou-se por adotar o contexto do comércio eletrônico para a realização das pesquisas, uma vez que esse ambiente é um dos mais impactados negativamente quando há um comportamento ruim do servidor em situações de sobrecarga.

Assim, tendo escolhido o ambiente de comércio eletrônico, foi realizada uma investigação das características das requisições Web típicas do *e-commerce*, para que tais características pudessem ser usadas para se melhorar o desempenho do modelo SWDS.

Em seguida, foi proposto um modelo de carga de trabalho e um modelo de simulação para a realização dos experimentos. Com isso, foi possível avaliar os resultados obtidos com a inserção de diversos mecanismos de diferenciação de serviços no SWDS.

Foram propostos novos mecanismos de escalonamento de requisições e também novos mecanismos de controle de admissão. Diversas simulações foram realizadas e os resultados

obtidos mostram que a exploração das características das requisições web, além de ser fundamental para um bom entendimento do comportamento do servidor, possibilita a melhoria de desempenho do sistema.

7.2 Resultados e Contribuições

As principais contribuições e resultados deste trabalho são apresentados a seguir:

1. **Revisão Bibliográfica -** Foi apresentada uma visão geral da infra-estrutura da Internet e da *World Wide Web*, sendo discutidas suas características e protocolos. Além disso, foi realizada também a descrição das formas de organização e funcionamento dos servidores Web. Em seguida, foram discutidos os problemas atuais da Internet e as vantagens da inserção de qualidade de serviço na rede. A abordagem de serviços diferenciados sobre redes IP e a viabilidade de seu emprego em nível de aplicação, particularmente em servidores Web, foram destacados. Na seqüência, foi apresentado o modelo SWDS. Foi descrita a sua arquitetura e discutidos alguns algoritmos de controle de admissão e diferenciação de serviços. Para finalizar a revisão bibliográfica, foram apresentadas algumas técnicas de avaliação de desempenho.

2. **Modelo e Geração da Carga de Trabalho -** Com base em trabalhos relacionados, um modelo de carga de trabalho para *e-commerce* foi desenvolvido considerando-se os principais tipos de requisições que caracterizam uma loja virtual. Utilizando o modelo desenvolvido, gerou-se uma carga sintética para a realização dos experimentos.

3. **Alterações no Modelo SWDS -** Foi realizado um detalhamento do SWDS para que fossem consideradas no modelo as características que causam os maiores impactos no

desempenho do sistema. No caso de um servidor Web, os consumos de CPU e de operações de I/O. Assim, no modelo proposto cada nó do cluster servidor é modelado com um recurso que representa o consumo de CPU e outro que representa o consumo de I/O. Essa modificação permite um melhor entendimento do funcionamento do servidor, bem como a identificação de gargalos e pontos de melhoria.

4. **Novas Políticas de Escalonamento -** Foram propostas e inseridas no modelo SWDS novas políticas de escalonamento para a realização da diferenciação de serviços com base nas características das requisições Web. Essas políticas utilizam como base o algoritmo WFQ e são comparadas às políticas que não utilizam qualquer tipo de priorização no escalonamento das requisições. O algoritmo WFQ é capaz de dividir a capacidade de processamento dos nós do *cluster* entre os diversos tipos de requisições, de acordo com os pesos que a eles são atribuídos. Assim, essa característica foi explorada tanto para priorizar o atendimento às sessões já admitidas pelo sistema, quanto para priorizar o atendimento às requisições com maiores probabilidades de gerar venda. Para a definição dos pesos a serem utilizados pelo algoritmo WFQ, foi proposto um modelo de otimização linear que leva em consideração os objetivos do negócio.

5. **Novas Políticas de Controle de Admissão -** Foram propostas e inseridas no modelo SWDS novas políticas de controle de admissão para a realização da diferenciação de serviços com base nas características das requisições Web. Essas políticas utilizam como base o tamanho médio das filas dos servidores Web no critério de admissão ou rejeição da requisição. Foi proposto também um mecanismo de admissão seletivo, que considera a probabilidade de gerar venda dos diversos tipos de requisição no processo

decisório.

A seguir, é apresentada uma tabela que sintetiza os principais resultados obtidos com a inserção dos mecanismos de diferenciação de serviços no modelo SWDS.

Configuração	Chegadas (n)	Admissões (n)	Índice de Rejeição	Términos (n)	Vendas Concluídas (n)	Índice de Eficiência	Tempo Med Resposta (s)
RR	215.851,40	215.851,40	**0,00%**	123.098,60	19,20	0,009%	**305,90**
WFQ (max. atendimentos)	**227.362,80**	**227.362,80**	**0,00%**	**134.178,80**	**0,00**	**0,000%**	109,89
WFQ (max. vendas)	207.556,20	207.556,20	**0,00%**	117.897,20	329,00	0,159%	84,11
RR - CA Fila	216.391,40	123.471,60	42,94%	123.470,80	12,00	0,006%	79,79
RR - CA Seletivo	**207.187,40**	**117.813,80**	**43,14%**	**117.808,20**	326,60	0,158%	79,46
WFQ - CA Seletivo	207.342,00	117.902,00	**43,14%**	117.902,80	424,20	0,205%	79,25
WFQ - CA Seletivo (fila reduzida)	207.425,80	117.964,40	43,13%	117.966,60	**435,80**	**0,210%**	**23,42**

Tabela 7.1: Resultados obtidos

Na Tabela 7.1, foram destacados em cor verde os resultados tidos como melhores casos e em vermelho os piores casos para diversos critérios de comparação. A seguir, seguem algumas explicações para os resultados em realce:

⇒ **Número de chegadas e términos:** o caso com o maior número de chegadas de requisições no sistema foi obtido com a utilização do algoritmo WFQ, configurado para maximizar o número de requisições atendidas. Isso ocorreu justamente porque o servidor, nessa configuração, consegue atender o maior número de requisições, pois são

atendidas menos requisições de maior custo de processamento (Buscas, Detalhes e Pagamentos). Essa característica já era esperada, uma vez que as requisições atendidas motivam a chegada de novas requisições, de acordo com o modelo de geração de carga de trabalho definido. O caso que apresentou o menor número de chegadas foi quando se utilizou o algoritmo RR com controle de admissão seletivo. Pelo modelo de geração de carga de trabalho definido, esse caso coincide com o caso em que são atendidas menos requisições. Isso ocorre porque são atendidas mais requisições de maior custo de processamento, por influência do critério de admissão seletiva. Por conseqüência, são admitidas menos requisições, o que eleva o índice de rejeição.

⇒ **Quantidade de requisições admitidas e índice de rejeição:** o melhor caso, quando foi admitido o maior número de requisições, ocorreu fazendo-se uso do algoritmo WFQ configurado para maximizar o número de requisições atendidas. A explicação está no fato de não ter sido utilizado nenhum mecanismo de controle de admissão e também por influência do algoritmo WFQ, que eleva o número de requisições concluídas e, conseqüentemente, o número de chegadas. A não utilização de mecanismos de controle de admissão influencia no índice de rejeição, que se mantém zerado. Os piores índices de rejeição foram obtidos quando se utilizou o mecanismo de admissão seletiva, pois são atendidas menos requisições, uma vez que o servidor conclui mais requisições com maior custo de processamento.

⇒ **Número de vendas concluídas e índice de eficiência:** o pior caso foi verificado quando utilizado o algoritmo WFQ configurado para maximizar o número de requisições atendidas. Isso ocorre porque são aceitas muitas sessões no sistema e, por conseqüência, poucas são concluídas. O melhor caso ocorreu com o uso do algoritmo

WFQ e controle de admissão seletivo, pois são priorizadas as requisições com maior probabilidade de gerar venda e o término das sessões admitidas pelo sistema. É importante destacar que foram obtidas evoluções muito relevantes no índice de eficiência utilizando-se mecanismos que penalizam sensivelmente outras métricas.

⇒ **Tempo médio de resposta:** o pior caso verificou-se com o algoritmo RR, quando não foi utilizado nenhum mecanismo de controle de admissão. Nessa configuração, é admitido o maior número de requisições com maior custo ao sistema, o que gera filas maiores e, por conseguinte, maiores tempos de resposta. O melhor caso ocorreu quando foi utilizado o algoritmo WFQ com controle de admissão seletivo e redução do tamanho da fila. Essa redução é a responsável pela queda no tempo de resposta.

7.3 Trabalhos Futuros

A seguir, são apresentadas algumas sugestões de trabalhos futuros:

- Análise do custo de processamento do algoritmo WFQ em ambientes reais, para que possa ser feito um estudo da relação custo benefício de sua utilização na prática;

- Estudo e aprimoramento do modelo de otimização linear apresentado para auxiliar na parametrização do algoritmo WFQ, de acordo com os objetivos do negócio. Esse modelo pode ser testado para outros perfis de comportamento de usuários, a fim de explorar melhor os benefícios trazidos por ele e entender suas limitações;

- Inserção de outros algoritmos de diferenciação de serviços e mecanismos de controle de admissão que possam aprimorar o desempenho do modelo SWDS e que

levem em consideração, além das características das requisições web, outras informações relevantes do sistema;

- Elaboração de mecanismos que induzam o usuário a percorrer caminhos menos custosos ao servidor em um fluxo de compra no site de comércio eletrônico. Isso poderia, por exemplo, reduzir requisições do tipo Buscas e ampliar o número de requisições do tipo Categorias, que são, em geral, menos custosas.

Referências

Andreolini, M.; Casalicchio, E.; Colajanni, M.; Mambelli, M. (2004). A cluster-based web system providing differentiated and guaranteed services. Cluster Computing, v.7, n.1, p.7–19.

Arlitt, M. (2000). Characterizing web user sessions. SIGMETRICS Perform. Eval. Rev., v.28, n.2, p.50–63.

Barbato, A. K.; Santana, R. H. C.; Traldi, O. A.; Santana, M. J.; Teixeira, M. A. M. (2005). Avaliação de Políticas de Controle de Admissão para Servidores Web com Serviços Diferenciados. XIII Simpósio Internacional de Iniciação Científica da Universidade de São Paulo (SIICUSP), São Carlos, Brasil.

Barbato, A. K. (2008). Políticas para servidores Web baseados em sessões visando qualidade e diferenciação de serviços. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.

Berners-Lee, T.; Fielding, R.; Frystyk, H. (1996). Hypertext Transfer Protocol - HTTP/1.0. RFC 1945, IETF.

Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. (1998). An Anchitecture for Dierentiated Services. RFC 2475, IETF.

Borenstein, N. (1993). MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC 1521,IETF.

Braden, R.; Clark, D.; Shenker, S. (1994). Integrated Services in the Internet Architecture. RFC 1633, IETF.

Braden, R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S. (1997). Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification. RFC 2205, IETF.

Cardellini, V.; Casalicchio, E.; Colajanni, M.; Mambelli, M. (2001). Web switch support for differentiated services. SIGMETRICS Perform. Eval. Rev., v.29, n.2, p.14–19.

Casalicchio, E.; Colajanni, M. (2001). A client-aware dispatching algorithm for web clusters providing multiple services. In Proceedings of the 10th international Conference on World Wide Web, p. 535{544, New York, NY. ACM Press.

Comer, D. E. (2000). Internetworking with TCP/IP: Principles, Protocols and Architecture. Prentice Hall, 4. edição.

Coulouris, G.; Dollimore, J.; Kindberg, T. (2000). Distributed Systems: Concepts and Design. Addison-Wesley, 3. edição.

E-Bit (2007). Informações do Comércio Eletrônico. Web Shoppers, 17. edição.

Estrella, J. C.; Teixeira, M. A. M.; Santana, M. J.; Santana, R. H. C.; Bruschi, S. M. (2006). Negotiation mechanisms on application level: A new approach to improve quality od service in web servers. WCCIA The 2nd International Workshop on Colaborative Computing, Integration and Assurance, p. 255-259. IEEE Proceedings.

Ferrari, D.; Serazzi, G.; Zeigner, A. (1983). Measurement and Tuning of Computer Systems. Prentice Hall.

Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. (1999). Hypertext Transfer Protocol - HTTP/1.1. RFC 2616, IETF.

Fishwick, P. A. (2004). SimPackJ: Version 1.0. University of Florida. Disponível em http://www.cise.ufl.edu/~fishwick/simpackj

Francês, C. R. L. (1998). Stochastic Feature Charts: Uma Extensão Estocástica para os Statecharts. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.

G. B. Dantzig, Maximization of a linear function subject to linear inequalities, in T. C. Koopmans (ed.), Activity Analysis of Production and Allocation, John Wiley & Sons, New York, 1951, pp. 339–347.

Goldberg, A.; Buff, R.; Schmitt, A. (1998). Secure web server performance dramatically improved by caching ssl session keys. Workshop on Internet Server Performance.

Hofmann, R.; Klar, R.; Mohr, B.; Quick, A.; Single, M. (1994). Distributed Performance Monitoring: Methods, Tools and Applications, vol. 5, 6 ed. IEEE Transactions on Parallel Distributed Systems.

Internet Traffic Archive (1998). World Cup web site access logs. Disponível em <http://ita.ee.lbl.gov/html/contrib/WorldCup.html>. Acesso em 14 nov. 2008.

Jain, R. (1991). The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation and Modeling. Wiley - Interscience, New York, NY.

Kant, K.; Mohapatra, P. (2000). Scalable Internet servers: Issues and challenges. Proceedings of the Workshop on Performance and Architecture of Web Servers (PAWS). ACM SIGMETRICS.

Kilkki, K. (1999). Differentiated Services for the Internet. Macmillan Publishing.

Kurose, J. F.; Ross, K. W. (2006). Rede de Computadores e a Internet: uma abordagem top-down. Addison Wesley, São Paulo.

Lee, S. C. M.; Lui, J. C. S.; Yau, D. K. Y. (2002). Admission control and dynamic adaptation for a proportional-delay diffserv-enabled web server. Proceedings of the 2002 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems.

Leon, H. P.; Bissett, S.; Rollin, P.; Ressler, M. (2008). osCommerce - Open Source E-Commerce Solutions. Disponível em: <http://www.oscommerce.org>. Acesso em: 14 nov. 2008.

MacDougall, M. H. (1987). Simulating Computer Systems Techniques and Tools. The MIT Press.

Magalhães, M. F.; Cardozo, E. (1999). Qualidade de serviço na Internet. Relatório técnico, UNICAMP/FEEC/DCA, Campinas, SP.

Menascé, D. A.; Almeida, V. A.; Fonseca, R.; Mendes, M. A. (1999). A methodology for workload characterization of E-commerce sites. In Proceedings of the 1st ACM Conference on Electronic Commerce. EC '99. ACM, New York, NY.

Microsoft Office Excel (2003). Microsoft Office Professional Edition. Microsoft Corporation.

MySQL (2008). Sun Microsystems. Disponível em <http://www.mysql.com>. Acesso em 14 nov. 2008.

Orfali, R.; Harkey, D.; Edwards, J. (1999). Client/Server Survival Guide. John Wiley, 3. edição.

Postel, J. (1980). User Datagram Protocol. RFC 768, IETF.

Postel, J. (1981a). Internet Control Message Protocol. RFC 792, IETF.

Postel, J. (1981b). Internet Protocol. RFC 791, IETF.

Postel, J. (1981c). Transmission Control Protocol. RFC 793, IETF.

Sabo, C. P. (2006). Avaliação de desempenho com algoritmos de escalonamento em clusters de servidores Web. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.

Santana, M. J. (1990a). Advanced Filestore Architeture for a Multiple-Lan Distributed Computing System. Phd thesis, University of Southamptom.

Santana, R. H. C. (1990b). Performance Evaluation of Lan-Based File Server. Phd thesis,

University of Southamptom.

Santana, R. H. C.; Santana, M. J.; Francês, C. R. L.; ORLANDI, R. C. G. S. (1997). Tool and Methodologies for Performance Evaluation of Distributed Computing System - A Comparison Study. In: The proceedings of the 1997, Portland, Oregon, USA.

Soares, L. F. G. (1990). Modelagem e Simulação Discreta de Sistemas. São Paulo: VII Escola de Computação.

SPECweb2005 (2005). SPECweb2005 E-Commerce Workload Design Document. Disponível em: <http://www.spec.org/web2005>. Acesso em 14 nov. 2008.

Stardust (1999a). White Paper — QoS protocols & architectures. Disponível em http: //www.qosforum.com

Stardust (1999b). White Paper — The need for QoS. Disponível em http://www.qosforum.com

Tanenbaum, A. S. (2002). Computer Networks. Prentice Hall, 4. edição.

Teixeira, M. A. M. (2004). Suporte a Serviços Diferenciados em Servidores Web: Modelos e Algoritmos. Tese de doutorado, Instituto de Ciências Matemáticas e de Computação - 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. XXIII Simpósio Brasileiro de Redes de Computadores (SBRC), Fortaleza, CE.

Traldi, O. A. (2005). Servidor web com diferenciação de serviços e algoritmo weighted fair queuing adaptativo (wfqadap). Monografia de conclusão de curso, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo.

Traldi, O. A.; Barbato, A. K.; Santana, R. H. C.; Santana, M. J.; Teixeira, M. A. M. (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, New York, NY, USA. ACM Press.

Vasiliou, N.; Lutfiyya, H. (2000). Providing a differentiated quality of service in a World Wide Web server. ACM SIGMETRICS Performance Evaluation Review, v.28, n.2, p.22–28.

W3C (1999a). HTML 4.01 specification. Disponível em <http://www.w3c.org/TR/html4>. Acesso em 14 nov. 2008.

W3C (1999b). HTTP 1.1 specification. Disponível em <http://www.w3.org/Protocols/rfc2616/rfc2616.html>. Acesso em 14 nov. 2008.

W3C (2000). XHTML 1.0 The Extensible HyperText Markup Language. Disponível em <http://www.w3c.org/TR/html>. Acesso em 14 nov. 2008.

Wei, Y.; Lin, C.; Ren, F.; Dutkiewicz, E.; Raad, R. (2003). Session based differentiated quality of service admission control for web servers. ICCNMC ’03: Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing, p. 112. IEEE Computer Society.

Xiao, X.; Ni, L. M. (1999). Internet QoS: a big picture. IEEE Network.

Yeager, N. J.; McGrath, R. E. (1996). Web Server Technology: the Advanced Guide for World Wide Web Information Providers. Morgan Kaufmann.

Zhao, W.; Olshefski, D.; Schulzrinne, H. (1999). Internet quality of service: an overview. IEEE Network.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo