# Servidor Web com Diferenciação de Serviços: Fornecendo QoS para os Serviços da Internet

**Mário Meireles Teixeira**[1] **, Marcos José Santana**[2] **, Regina H. C. Santana**[2]

[1]Universidade Federal do Maranhão – Departamento de Informática
Av. dos Portugueses s/n, Campus do Bacanga
65085-580 São Luís, MA

[2]Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação
Caixa Postal 668 – 13560-970 São Carlos, SP

mario@deinf.ufma.br

{mjs,rcs}@icmc.usp.br

***Abstract.*** *Our work proposes an architecture for a web server capable of providing differentiated services to its clients according to their demand characteristics. The architecture is validated by means of a simulation model and real web server traces are used for workload generation. We have designed and implemented two priority-based service differentiating mechanisms in this architecture with very promising results. Our server also deals with admission control issues, in order to prevent system overload when user demand rises unexpectedly. Our results have been compared both with and without overload control, so as to assess its influence on overall server performance. Overload control has proved to be a fundamental feature to assure system stability as well as the quality of service contracted by the clients.*

***Resumo.*** *Este trabalho propõe uma arquitetura de servidor web capaz de fornecer serviços diferenciados a seus clientes de acordo com suas características de demanda. A arquitetura é validada por meio de simulação e* traces *de acesso a servidores web reais são utilizados na geração da carga de trabalho. Neste artigo, descreve-se a implementação de dois mecanismos de diferenciação de serviços baseados em prioridades, nesta arquitetura, ambos com resultados muito animadores. A arquitetura proposta também contempla aspectos de controle de admissão, procurando evitar a sobrecarga do sistema caso a demanda dos usuários cresça inesperadamente. Os resultados obtidos foram avaliados na presença e ausência do módulo de controle de admissão, a fim de determinar sua influência no desempenho global do sistema. Conclui-se que o controle de sobrecarga é de fundamental importância para garantir a estabilidade do sistema, assim como a qualidade de serviço contratada pelos clientes.*

## 1. Introdução

A Internet vem experimentando um crescimento intenso nos últimos anos e não se vêem sinais de que tal tendência vá se reverter em um futuro próximo. Na última década, impulsionado principalmente pelo surgimento da Web e pelo crescimento dos provedores comerciais, o tráfego na rede aumentou em algumas ordens de grandeza, numa tendência que se mantém até hoje. Entretanto, originalmente a Internet não foi projetada para o uso observado atualmente, nem para dar suporte à quantidade de carga que lhe é imposta.

O serviço oferecido atualmente na Internet baseia-se em um modelo de melhor esforço e uma das conseqüências de sua adoção é o fato de que todo o tráfego é tratado de maneira uniforme, sem nenhum tipo de diferenciação ou priorização, uma tendência que se reflete até mesmo no projeto de serviços críticos da Internet, como a Web, cujos servidores, em sua grande maioria, tratam as requisições dos clientes segundo uma disciplina em que o primeiro a chegar será o primeiro a ser atendido. Contudo, verifica-se que nem todos os tipos de tráfego e transações são equivalentes ou têm a mesma prioridade para os usuários [Dovrolis and Ramanathan, 1999] [Kant and Mohapatra, 2000]. Considerando-se a infra-estrutura de rede, existem algumas propostas nesse sentido, elaboradas sob a coordenação da IETF (*Internet Engineering Task Force*). Entretanto, em nível de aplicação os esforços ainda são limitados, não sendo encontrados na grande maioria dos servidores web em operação [Vasiliou and Lutfiyya, 2000]. Qualquer tentativa de fornecer uma qualidade de serviço (QoS) diferenciada na Web não terá sucesso se forem empregados apenas mecanismos em nível de rede, pois, em última instância, são os servidores web os responsáveis pelo atendimento das solicitações dos usuários.

Este trabalho propõe uma nova arquitetura para um servidor web capaz de fornecer serviços diferenciados a seus usuários e aplicações, de acordo com seu perfil de utilização. Um servidor web com esta característica pode melhor responder às exigências de qualidade das novas aplicações da Internet atual, propiciando um serviço com um comportamento mais estável e segundo requisitos de qualidade de serviço previamente contratados.

Considera-se, neste trabalho, a existência de duas classes de usuários e analisa-se a aplicação de dois mecanismos de diferenciação de serviços baseados em prioridades na arquitetura. Também é implementado um mecanismo de controle de admissão de requsições, o qual se baseia em uma média exponencialmente ponderada da utilização do sistema para a tomada das decisões de descarte. O modelo do Servidor Web com Diferenciação de Serviços (SWDS) é validado por meio de uma simulação dirigida por *traces*, sendo utilizados *logs* de acesso a servidores web reais como carga de trabalho. Os resultados obtidos demonstram a viabilidade da arquitetura proposta para a diferenciação de serviços em servidores web, bem como a eficiência dos algoritmos implementados.

O restante deste artigo está organizado da seguinte maneira: a Seção 2 propõe um modelo para um servidor web com diferenciação de serviços, destacando suas características mais relevantes. Na Seção 3, discute-se a validação do modelo por meio de simulação, envolvendo aspectos como geração de carga de trabalho e parametrização. As Seções 4 e 5 detalham os algoritmos de prioridades, rigoroso e adaptativo, implementados no modelo, analisando os resultados obtidos. A Seção 6 descreve o mecanismo de controle de admissão implementado e compara os resultados obtidos na presença e ausência do controle de sobrecarga. Finalmente, na Seção 7 destacam-se as principais conclusões deste trabalho.

## 2. Modelo para um Servidor Web com Diferenciação de Serviços

Os servidores web atuais tratam as requisições que recebem segundo uma disciplina FIFO (*First In, First Out*), ou seja, é mantida uma única fila de espera onde cada requisição aguarda o momento de ser atendida, segundo sua ordem de chegada. Embora diferentes esquemas de controle de concorrência possam ser implementados no servidor, visando agilizar o atendimento das requisições, o mesmo ocorre em geral de maneira uniforme, sem considerar as particularidades e a urgência de cada tipo de requisição.

Este trabalho propõe um novo modelo para um *Servidor Web com Diferenciação de Serviços*, o qual deverá ser capaz de oferecer diferentes níveis de serviço a seus clien-

tes, garantindo a sua qualidade. A Figura 1 descreve a arquitetura proposta, que é formada pelos seguintes módulos: um Classificador, um módulo de Controle de Admissão e um *cluster* de processos ou servidores web.

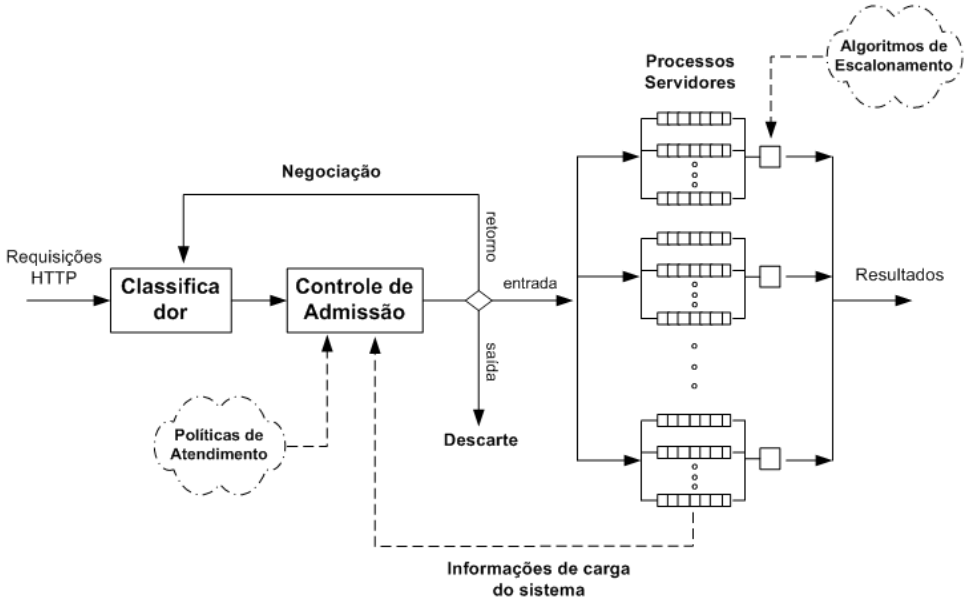



**Figura 1: Servidor Web com Diferenciação de Serviços (SWDS)**

O **Classificador** é o elemento responsável por receber as requisições que chegam ao sistema e subdividi-las em classes, segundo critérios pré-estabelecidos. O módulo de **Controle de Admissão** gerencia a aceitação de novas requisições pelo servidor, levando em conta as políticas de atendimento correntes e as informações da carga de trabalho do sistema. Caso o mesmo esteja muito sobrecarregado, uma requisição poderá ser rejeitada (descarte) ou ter suas exigências de qualidade de serviço relaxadas (negociação), a fim de que possa ser aceita em uma classe de prioridade inferior. 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 o processamento, os resultados são entregues aos clientes.

Para os propósitos deste trabalho, cada nó do *cluster* é visto como um servidor web convencional, composto de CPU, disco e interface de rede, não sendo considerada a presença de memória *cache* nos servidores. Cada nó possui múltiplas filas de prioridade, a fim de acomodar as diferentes classes de serviço. Neste artigo, considera-se que os nós formam um *cluster* de servidores, mas nada impede que se faça uma abstração destes para processos, tarefas ou até mesmo CPUs em um computador paralelo, pois o modelo não exige que o *cluster* seja necessariamente formado por máquinas dispostas em um sistema distribuído. Igualmente, não é pressuposta nenhuma plataforma de hardware ou sistema operacional em particular.

O modelo proposto para o servidor SWDS é bastante flexível e presta-se à implementação de diversos algoritmos de diferenciação de serviços. O modelo pode servir como base para a implementação de um programa servidor ou para a construção de infra-estruturas de servidores web distribuídos voltadas ao atendimento diferenciado de uma ampla gama de clientes.

## 3. Experimentação do Modelo

A fim de validar o modelo proposto, optou-se por uma simulação orientada a eventos, usando o pacote de simulação SIMPACK[1]. Nesta seção, discutem-se aspectos como geração da carga de trabalho e parametrização do modelo.

### 3.1. Geração de Carga

Estudos anteriores mostraram que a Web apresenta uma grande variabilidade nas características da sua carga de trabalho, sendo difícil definir um modelo de carga que se aplique a todas as situações. Foi observado, por exemplo, que a distribuição dos tamanhos de arquivo possui características de cauda pesada [Arlitt and Williamson, 1996] e também que o tráfego na rede se apresenta em rajadas, em diferentes escalas de tempo, tendo um comportamento auto-similar [Crovella and Bestavros, 1997]. Sendo assim, dada a dificuldade de produzir sinteticamente uma carga com essas características, e a exemplo de outros trabalhos da área, optou-se por utilizar neste estudo *traces* de acesso a servidores web reais para a geração de carga.

Durante a simulação, os registros do *log* são lidos seqüencialmente e usados como entrada para o modelo, onde são processados de acordo com a parametrização descrita na Seção 3.2. No caso de coincidência de *timestamps* dos registros, o que é comum, assume-se que as chegadas estão uniformemente distribuídas em um intervalo entre 0 e 1 segundo, no máximo. O valor superior deste intervalo é variado a fim de gerar diferentes taxas de chegada para as requisições HTTP.

Nos experimentos, foi usado o *log* correspondente a parte de um dia de acesso, o qual possui 7 milhões de registros e registra consultas a 27 servidores web diferentes. Uma caracterização detalhada desta carga de trabalho pode ser encontrada em [Arlitt and Jin, 1999].

### 3.2. Parametrização do Modelo

A fim de processar cada requisição HTTP, um servidor web típico deve realizar diversas tarefas: análise da URL fornecida, autenticação do usuário, leitura do arquivo no disco, sua transmissão para o cliente e a gravação do *log* [Hu et al., 1999] [Menascé and Almeida, 2003]. Para arquivos pequenos, o tempo de processamento é dominado pela CPU, devido às várias tarefas necessárias para atender uma requisição HTTP. Para arquivos grandes, a maior parte do tempo é consumida com processamento de E/S (disco e rede). No caso de requisições dinâmicas, a CPU é novamente bastante exigida.

Para fins de experimentação, cada nó do *cluster* foi modelado separadamente, com sua própria CPU, disco e interface de rede, sem presença de memória *cache*. O modelo do servidor SWDS também contempla os módulos de Classificação e Controle de Admissão, os quais influem no desempenho global do sistema. A parametrização do modelo, resumida na Tabela 1, tomou por base observações próprias, feitas a partir de *benchmarks* realizados na rede do LaSDPC[2], bem como indicações encontradas em outros trabalhos da área [Cardellini et al., 2001] [Casalicchio and Colajanni, 2000] [Chen and Mohapatra, 1999] [Hu et al., 1999] [Menascé and Almeida, 2003].

### 3.3. Metodologia de Teste

Como já foi dito, a fim de validar o modelo e os algoritmos aqui propostos, recorreu-se à simulação, utilizando-se como carga de trabalho os *logs* da web discutidos anteriormente. O programa de simulação implementa o modelo do SWDS apresentado na Figura 1, com quatro servidores web homogêneos formando um *cluster*, o qual obedece

[1]Disponível em `http://www.cise.ufl.edu/~fishwick/simpack/simpack.html`

[2]Laboratório de Sistemas Distribuídos e Programação Concorrente do ICMC-USP.

| Parâmetro | Valor |
|---|---|
| Capacidade do Classificador | 8000 req/seg |
| Capacidade do Controle de Admissão | 4000 req/seg |
| Taxa de transferência do disco | 37 MBps |
| Latência do disco | 8,5 ms |
| Capacidade da interface de rede | 80 Mbps |
| Tempo de serviço das requisições dinâmicas | 10 ms |

**Tabela 1: Parâmetros do modelo (SWDS)**

à parametrização descrita na Seção 3.2. Assume-se que todos os nós têm acesso aos mesmos documentos. Inicialmente, optou-se por não utilizar o controle de admissão das requisições HTTP em caso de sobrecarga do sistema, a fim de não interferir no desempenho dos algoritmos de diferenciação de serviços. Dessa forma, consideram-se as filas dos servidores web como ilimitadas.

Nos experimentos, as requisições que chegam ao sistema são divididas em duas classes, de alta e baixa prioridade, pelo módulo Classificador da arquitetura, sendo atribuídas aos nós segundo um esquema *round robin*. A disciplina de fila de cada nó é determinada pelos algoritmos descritos nas Seções 4 e 5 a seguir. Os resultados apresentados correspondem à média de dez simulações, com um intervalo de confiança de 95%.

## 4. Mecanismo de Prioridades Rigoroso

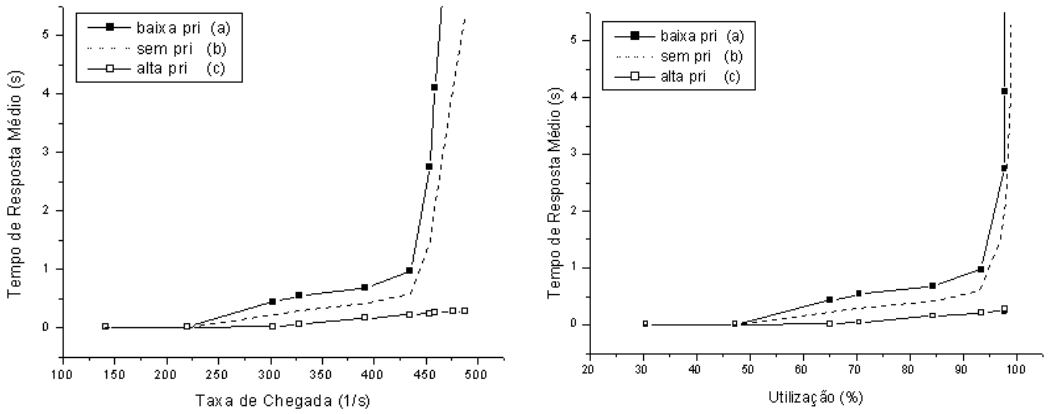
O Mecanismo de Prioridades Rigoroso (*Strict Priority Queueing*) aqui implementado exige que o atendimento das requisições obedeça estritamente a ordem das prioridades, ou seja, somente serão atendidas requisições de prioridade inferior se não houver nenhuma requisição de prioridade superior aguardando em fila [Allen, 1990]. Nos experimentos, as classes de serviço em que se dividem os clientes do sistema (portanto, as requisições HTTP) são mapeadas nas diferentes classes ou níveis de prioridade oferecidos pelo servidor, numeradas de 0 a $n$. Quanto maior o número da classe maior a sua prioridade no sistema. Os clientes que possuírem a mesma prioridade serão atendidos, dentro de sua classe, segundo uma disciplina FIFO. O atendimento é do tipo não-preemptivo e uma requisição atribuída a um processo servidor deve permanecer nele até a conclusão do serviço.

### 4.1. Resultados Experimentais

Inicialmente, considerou-se o caso em que 50% das requisições que chegam ao sistema pertencem à classe de alta prioridade. A Figura 2(a) mostra o comportamento do tempo de resposta médio para as duas classes de requisições e constata-se que o tempo de resposta das requisições de alta prioridade (curva c) permanece baixo mesmo a taxas de chegada elevadas. Quanto às tarefas de baixa prioridade (curva a), nota-se uma acentuada degradação no seu desempenho a partir de 435 requisições/s, conseqüência do tratamento preferencial dado à classe superior. A curva intermediária (b) representa o tempo de resposta quando não se utiliza a diferenciação de serviços, o caso de um servidor web convencional.

Também foi analisado o comportamento do tempo de resposta em relação à utilização do sistema (Figura 2(b)). O percentual de 50% de requisições na classe de alta prioridade é mantido durante todo o experimento e pode-se verificar que o sistema de fato fornece tratamento diferenciado às requisições, em diferentes níveis de utilização. As requisições de alta prioridade apresentam um tempo de resposta baixo mesmo quando

o sistema se aproxima da utilização completa, o que mostra que o uso de prioridades é um mecanismo eficiente para se atingir a diferenciação de serviços, pois consegue-se reservar uma parcela da capacidade de processamento do servidor para a classe de maior prioridade, independentemente da carga de trabalho presente no sistema.




(a) Taxa de Chegada (b) Utilização

**Figura 2: Tempo de resposta usando o mecanismo de prioridades rigoroso**

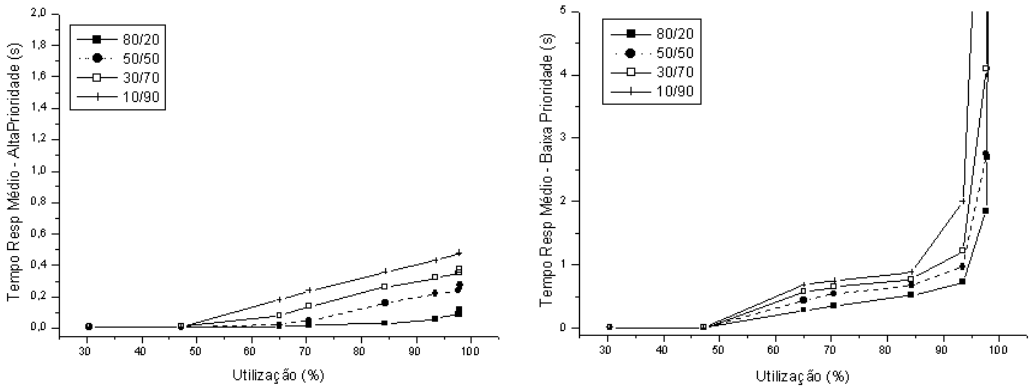
### 4.1.1. Variação dos Percentuais de Clientes nas Classes de Serviço

Em seguida, foram realizados novos experimentos com diferentes distribuições dos clientes nas classes de serviço, a fim de verificar sua influência no tempo de atendimento das requisições. A Figura 3(a) mostra o comportamento do tempo de resposta das requisições de alta prioridade, em relação à utilização do sistema, para os casos em que se têm 20, 50, 70 e 90% dos clientes nesta classe. Observa-se que, à medida que se aumenta o percentual de clientes na classe de alta prioridade, a inclinação da curva do tempo de resposta ocorre a níveis cada vez mais baixos de utilização, o que denota a saturação do sistema. Dessa forma, há um limite razoável para o número de clientes que podem ser considerados de alta prioridade, sob pena de não ocorrer a diferenciação de serviços.

Na Figura 3(b), vê-se a mesma situação acima sob o ponto de vista das requisições de baixa prioridade, para os casos em se têm 80, 50, 30 e 10% dos clientes na classe de menor prioridade. Novamente, a inclinação da curva do tempo de resposta ocorre a níveis de utilização progressivamente menores, à medida que mais clientes são alocados à classe de alta prioridade, os quais acabam por monopolizar os recursos do sistema. Nesta situação, as requisições de baixa prioridade têm pouca influência no desempenho global do servidor e recebem pouca ou nenhuma atenção do mesmo. Pode-se notar que, em todos os casos analisados, o tratamento recebido pelas requisições da classe de alta prioridade é nitidamente superior ao experimentado pelas requisições de baixa prioridade.

Resultados adicionais, discutidos em [Teixeira et al., 2004], mostram que, a altas taxas de utilização do sistema, o tratamento recebido pelas requisições de baixa prioridade apresenta uma degradação considerável, chegando ao ponto em que menos de 2% delas são atendidas. Tem-se, neste caso, uma situação de negação de serviço, o que pode fazer com que os clientes menos privilegiados parem de submeter requisições ao servidor.

Conclui-se, portanto, que é preciso estar atento ao comportamento de todas as classes de serviço, uma vez que não é, em geral, desejável que as classes de menor prioridade sejam por demais penalizadas. Caso contrário, poder-se-ia afastar clientes que, embora aparentemente de “menor” importância, ainda assim são potenciais geradores de




(a) Alta prioridade (b) Baixa prioridade

**Figura 3: Variação dos percentuais de clientes**

receita para o negócio representado por um dado site na Web. Além disso, se a maioria dos clientes que chegam a um site é de alta prioridade, na prática fica difícil produzir alguma forma de diferenciação de serviços entre eles, qualquer que seja a abordagem empregada.

## 5. Mecanismo de Prioridades Adaptativo

O mecanismo de prioridades adaptativo (PRIAdap), proposto pelos autores deste artigo em [Teixeira et al., 2004], procura fazer uma sintonia fina do emprego das prioridades, relaxando ou intensificando a sua utilização conforme o caso. Dessa forma, pode-se atribuir uma maior ou menor importância às requisições de alta prioridade, a fim de evitar que estas venham a monopolizar o uso dos recursos do sistema. O mecanismo adaptativo reconhece que as requisições de menor prioridade não podem esperar indefinidamente nas filas (como às vezes ocorre no mecanismo rigoroso), por isso introduz uma certa flexibilidade na escolha da próxima requisição a ser atendida.

Para fins de implementação do algoritmo, cada processo servidor possui uma fila de espera única, na qual as requisições são inseridas por ordem de chegada, independentemente de sua prioridade. É utilizado 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 e assim por diante. Em último caso, será escolhida a primeira requisição da fila. Quanto maior o valor de $k$, tanto melhor será o tratamento dispensado às requisições de maior prioridade e, para $k = 1$, as requisições serão atendidas segundo sua ordem de chegada, sem diferenciação.

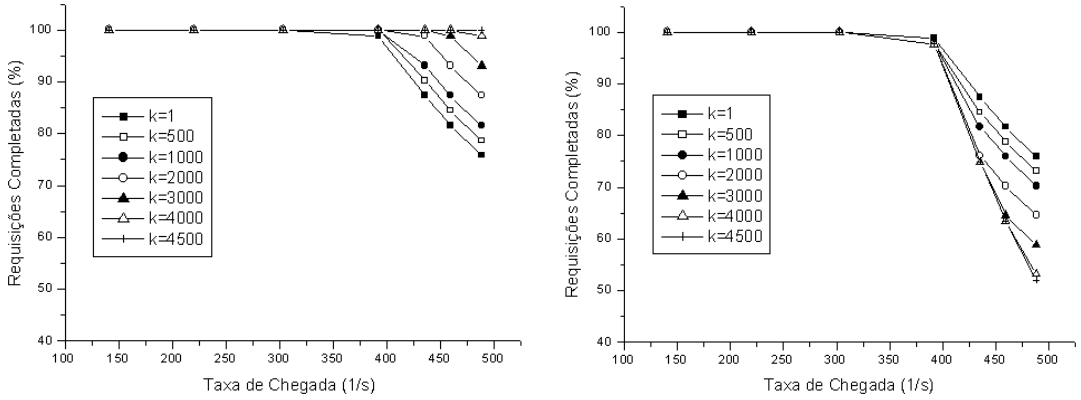
O uso do *look-ahead* como parâmetro de controle do algoritmo permite regular o nível de priorização do sistema ou, em outras palavras, determina quão rigoroso será o esquema de prioridades empregado. O algoritmo PRIAdap introduz características de adaptabilidade no comportamento do servidor SWDS, permitindo ajustá-lo para melhor atender às variações na carga de trabalho e na distribuição dos clientes pelas classes de serviço.

### 5.1. Resultados Experimentais

A validação do algoritmo PRIAdap foi feita segundo a metodologia da Seção 3.3, usando-se o modelo apresentado para o servidor SWDS (Figura 1). Neste caso, como exigido pelo próprio algoritmo, cada nó do *cluster* possui uma única fila de espera e não múltiplas filas e as requisições são atendidas conforme o algoritmo de prioridades adaptativo descrito.

Em um mesmo instante, todos os nós possuem sempre o mesmo valor para o *look-ahead*. Considera-se que 50% dos clientes pertencem à classe de alta prioridade. O objetivo dos experimentos é determinar se o uso de um mecanismo adaptativo na priorização das requisições HTTP pode garantir a diferenciação de serviços sem sacrificar demasiadamente as requisições de baixa prioridade.

Inicialmente, foram realizadas simulações buscando avaliar a influência do valor escolhido para o *look-ahead* no percentual de requisições completadas com sucesso. Os experimentos buscam determinar o percentual de requisições concluídas a diferentes taxas de chegada, sendo os valores do *look-ahead* ($k$) variados entre 500 e 4.500. Na Figura 4(a), pode-se notar que valores maiores de $k$ aumentam gradativamente o percentual de requisições de alta prioridade que conseguem ser completadas, ao mesmo tempo em que pioram sensivelmente o atendimento recebido pelas requisições de baixa prioridade (Figura 4(b)).




(a) Alta prioridade (b) Baixa prioridade

**Figura 4: Percentual de requisições completadas com diferentes valores do *look-ahead***

Como se pode observar, para um valor de $k = 1$, o atendimento recebido pelas requisições de ambas as classes é virtualmente o mesmo, tanto que, a taxas de chegada superiores a 400 requisições/s, algumas requisições de alta prioridade nem mesmo chegam a ser completadas. Por outro lado, quando o valor de $k$ supera o tamanho máximo observado para as filas do servidor ($k = 4.500$), o algoritmo PRIAdap passa a se comportar como o mecanismo de prioridades rigoroso comentado anteriormente. Neste caso, as requisições de baixa prioridade têm o seu pior desempenho, com menos de 50% delas sendo completadas. Finalmente, para $500 \leq k \leq 4.000$, constatam-se os outros níveis de diferenciação de serviços atingidos, conforme a proposta inicial do algoritmo PRIAdap, que é a de regular o nível de priorização empregado pela arquitetura.

O ajuste do parâmetro de *look-ahead* é crucial para o desempenho do servidor SWDS. Um valor de $k$ por demais elevado pode levar à negação de serviço para as requisições de menor prioridade, enquanto um valor muito baixo prejudica a diferenciação de serviços. O *look-ahead* pode ser ajustado manualmente pelo administrador do sistema, ou dinamicamente a partir das informações da carga de trabalho.

Para concluir, foi analisado o comportamento do tempo de resposta médio das requisições, mostrado na Figura 5. Observa-se que, para $k = 1$, as curvas se sobrepõem, pois o mesmo tratamento é dispensado a ambas as classes. Entretanto, para $k = 3000$, a diferenciação de serviços já se torna evidente e o atendimento recebido pelas requisições de alta prioridade é visivelmente melhor.

Os experimentos realizados permitem-nos concluir que a variação controlada

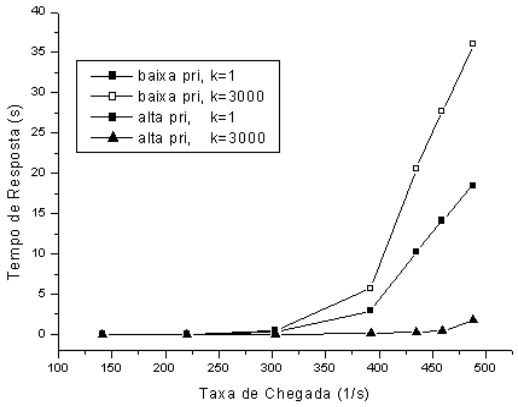



**Figura 5: Tempo de resposta usando o mecanismo de prioridades adaptativo**

do valor do *look-ahead* constitui-se em um mecanismo apropriado para fornecer diferenciação de serviços entre diferentes classes de requisições. O algoritmo PRIAdap evita os problemas de negação de serviço evidenciados no mecanismo de prioridades rigoroso e possibilita ao administrador do sistema um controle mais efetivo sobre o atendimento dispensado aos clientes.

O *look-ahead* pode ser ajustado segundo as características da carga de trabalho e as políticas vigentes na organização, podendo também ser atualizado automaticamente pelo sistema a partir de informações obtidas do módulo de Controle de Admissão. Este parâmetro seria, então, ajustado periodicamente em função de certos objetivos colocados pelo administrador como, por exemplo, o tempo de resposta máximo esperado para as requisições de alta prioridade ou o *throughput* médio que se busca atingir para as requisições de uma determinada classe de serviço. Em um cenário real, esses objetivos seriam, de fato, estabelecidos em Acordos de Níveis de Serviço (*Service Level Agreements* – SLAs) firmados com os clientes do sistema.

Outra vantagem do mecanismo adaptativo proposto é que ele é distribuído por natureza. Muitos dos esquemas encontrados atualmente [Cardellini et al., 2001] dependem de algum componente central (um despachante) para realizar a diferenciação de serviços. Nossa abordagem adaptativa, por sua vez, transfere a carga associada com a diferenciação de serviços para os nós do *cluster*, o que alivia a carga no despachante e melhora a escalabilidade e a confiabilidade do sistema.

## 6. Controle de Admissão

Os algoritmos de diferenciação de serviços descritos nas Seções 4 e 5 correspondem à funcionalidade de escalonamento do servidor SWDS e determinam a ordem de atendimento das requisições. Entretanto, verifica-se que o controle de sobrecarga é fundamental para garantir a qualidade dos serviços da Internet. Por vezes, tenta-se controlar a carga do sistema estabelecendo-se limiares fixos, especificados pelo administrador, tais como número de conexões abertas ou de *threads* ativas [Welsh and Culler, 2003]. Contudo, é difícil determinar os limites ideais, de forma estática [Iyer et al., 2000], em um ambiente de características tão variáveis como a Web. Por esta razão, este artigo propõe um mecanismo de controle de admissão sensível a mudanças na carga de trabalho oferecida ao sistema, o qual foi implementado no modelo do servidor SWDS. A validação do mecanismo é semelhante à abordagem descrita na Seção 3.3.

### 6.1. Configuração de Referência

Inicialmente, foi feito um experimento sem o emprego do controle de admissão, a fim de se ter uma referência com a qual comparar os resultados do mecanismo de controle. A faixa do *log* escolhida para a geração de carga foi entre 3,6 e 4,6 milhões de registros, usando-se todos os servidores do *log*. Nesta faixa, ocorre um súbito aumento da carga de trabalho, situação ideal para se avaliar a eficiência do controle de admissão. O algoritmo de diferenciação de serviços empregado foi o PRIAdap, com um valor de *look-ahead* igual a 300.

A Figura 6 mostra o comportamento do tempo de resposta no decorrer do experimento. Constata-se uma situação caótica, pois o mesmo sobe continuamente, acompanhando o aumento da carga de trabalho que chega ao servidor. A partir de cerca de 500 segundos de tempo de simulação, quando ocorre um súbito incremento na carga, as novas requisições tendem a esperar cada vez mais para ser atendidas. Embora a diferenciação de serviços seja respeitada, observa-se que o algoritmo PRIAdap sozinho não é capaz de garantir uma qualidade mínima de atendimento para nenhuma das duas classes. No pior caso, o tempo de resposta chega a quase 90 segundos, o que inviabiliza o atendimento aos clientes.

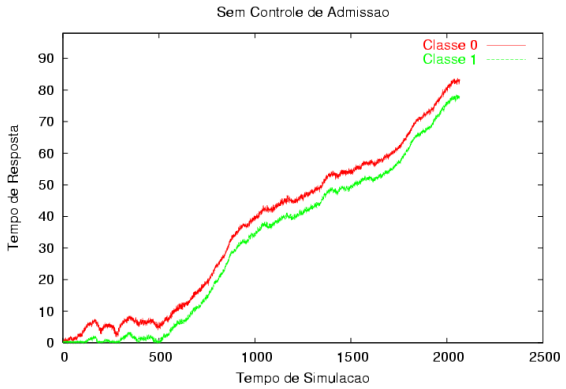



**Figura 6: Tempo de resposta sem utilização do Controle de Admissão**

### 6.2. Admissão segundo a Utilização do Sistema

O mecanismo de controle de admissão aqui proposto emprega a *utilização média do cluster* de servidores web como métrica, a qual é calculada através de uma média exponencialmente ponderada. A escolha da utilização como parâmetro de controle é bastante conveniente, pois ela é capaz de indicar, numa visão global, se um sistema está de fato sobrecarregado. O uso de uma média, em vez do valor real da utilização, tem o objetivo de minimizar o efeito de um congestionamento momentâneo no servidor e o peso escolhido determina a sensibilidade da média a mudanças na utilização do *cluster*. Abordagem semelhante a essa é encontrada no algoritmo RED (*Random Early Detection*), o qual descarta os pacotes das filas dos roteadores antes que ocorra uma situação de sobrecarga. Para tanto, emprega uma média exponencialmente ponderada do tamanho das filas, além de limiares probabilísticos [Stallings, 2002].

O algoritmo funciona da seguinte forma: a utilização atual do *cluster* é medida a cada nova requisição que chega ao servidor e este valor é combinado com dados históricos a fim de obter a média exponencialmente ponderada, $U_{med}$. Caso a mesma esteja acima de um limiar pré-estabelecido (`MAXUTIL`), então o servidor SWDS recusará quaisquer novas requisições, independentemente de sua classe, até que $U_{med}$ caia para níveis aceitáveis, isto é, abaixo do limiar.

Seja assim definido o cálculo da média exponencialmente ponderada:

$$U_{med} = (1 - p)U_{ant} + pU_{atual} \quad (1)$$

onde:

- $U_{ant}$ – valor anterior (histórico) da média;
- $U_{atual}$ – valor atual observado para a utilização;
- $p$ – peso, onde $0 \leq p \leq 1$.

O peso $p$ funciona como um coeficiente de sensibilidade a mudanças na carga do sistema, implicando em:

a) $p \rightarrow 1$: o mecanismo de controle torna-se bastante sensível a mudanças na utilização do sistema, reagindo prontamente. Indicado para situações em que o servidor esteja continuamente exposto a uma carga elevada;
b) $p \rightarrow 0$: o mecanismo de controle reage mais lentamente a mudanças na utilização do sistema, apresentando um comportamento mais estável. Indicado para situações em que a carga é geralmente baixa, apresentando algumas rajadas ocasionais.

Entre esses dois valores extremos do peso $p$, têm-se diferentes perfis de comportamento do módulo de Controle de Admissão, desde o mais complacente até o mais rigoroso. Na configuração (a), o servidor apresenta uma política de admissão mais restritiva, devido ao peso da utilização medida ($U_{atual}$) no cálculo da média. Neste caso, o tempo de resposta e as filas tendem a ser menores, pois as recusas de serviço precoces evitam a sobrecarga do sistema. Na configuração (b), a política de admissão leva em conta o histórico da carga do sistema ($U_{ant}$), minimizando o impacto da utilização atual na tomada de decisões. Neste caso, para cargas altas, o tempo de resposta e o número de requisições não completadas tendem a ser maiores, já que o servidor é menos severo na recusa das requisições.

**Resultados Experimentais**

Os experimentos foram repetidos com a mesma configuração da Seção 6.1, para efeito de comparação. Foi escolhida a faixa do *log* entre 3,6 e 4,6 milhões de registros, usando-se o algoritmo PRIAdap para a diferenciação de serviços, com o *look-ahead* ajustado para 300. O limiar de utilização `MAXUTIL` foi fixado em 90% e o peso $p$, em 0,9.

A Figura 7 mostra o comportamento do tempo de resposta no decorrer da simulação. O controle de admissão começa a operar desde o início, pois a carga oferecida ao sistema supera o limiar `MAXUTIL`. Observa-se que o tempo de resposta cai drasticamente em relação ao caso sem controle de admissão (Figura 6) e mantém-se assim durante toda a simulação, independentemente do aumento da carga.

A Tabela 2 mostra o tamanho máximo observado para as filas dos recursos do servidor SWDS: no caso sem controle de admissão, as filas de espera dos servidores web do *cluster* chegam a mais de 9.700 requisições, o que comprova a situação de extrema sobrecarga do sistema, tornando-lhe impossível fornecer um serviço com uma qualidade minimamente aceitável a seus clientes. Por outro lado, quando o controle de admissão é ativado, as filas de espera dos nós do *cluster* caem para cerca de 260 clientes, um valor perfeitamente gerenciável. A utilização permanece sempre abaixo do limiar `MAXUTIL`.

Na Tabela 3, tem-se uma comparação dos tempos médios de resposta observados: no caso sem controle de admissão, estes chegam a mais de 30 segundos para ambas as

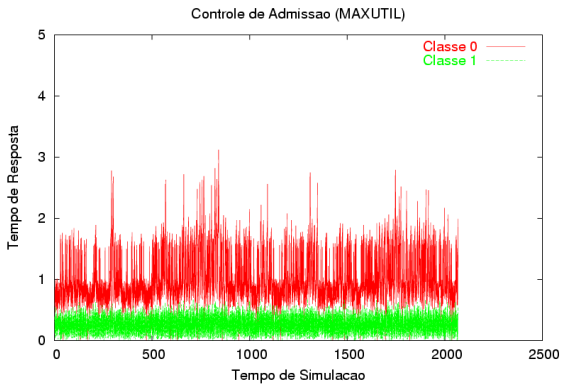



**Figura 7: Tempo de resposta com Controle de Admissão (`MAXUTIL=90%`)**

| | Classificador | Controle de Admissão | Servidores Web |
|---|---|---|---|
| **Sem Ctrl.Adm.** | 624 | — | 9760 |
| **Com Ctrl.Adm.** | 624 | 339 | 264 |

**Tabela 2: Comparação do tamanho máximo das filas de espera dos recursos**

classes, tornando quaisquer esforços de diferenciação de serviços inúteis. Porém, quando o controle de sobrecarga é ativado, o tempo de resposta das requisições de menor prioridade (classe 0) cai para cerca de 1 segundo e o das requisições de maior prioridade (classe 1), para apenas 0,27 segundo. Note-se que a diferenciação de serviços entre as classes é respeitada.

E ainda, ao se observar a razão entre o número de términos e de requisições admitidas no sistema (Tabela 3), vê-se que, quando não é usado o controle de admissão, cerca de 4% das requisições de ambas as classes não conseguem ser completadas, devido à sobrecarga do servidor. Por outro lado, quando o controle de admissão é ativado todas as requisições admitidas no sistema (cerca de 86% do total de chegadas) conseguem ser concluídas. O controle de admissão, portanto, evita aceitar no sistema um número excessivo de requisições, as quais apenas contribuiriam para o aumento de sua sobrecarga. Assim, tem-se uma melhora substancial no tempo de serviço das requisições que são admitidas para processamento.

| | Classe | Tempo de Resposta | Chegadas | Admissões | Términos |
|---|---|---|---|---|---|
| **Sem Ctrl.Adm.** | 0 | 36,24 | 503.703 | 503.763 | 483.813 |
| | 1 | 31,37 | 495.166 | 495.165 | 476.510 |
| **Com Ctrl.Adm.** | 0 | 1,04 | 503.801 | 436.369 | 436.369 |
| | 1 | 0,27 | 495.128 | 428.966 | 428.966 |

**Tabela 3: Comparação de medidas de desempenho entre as classes de serviço**

O algoritmo de controle de admissão implementado não discrimina entre as classes de serviço ao tomar as decisões de descarte, recusando as requisições indistintamente assim que o servidor atinge o limiar de utilização estipulado. Em trabalhos futuros, pretende-se explorar também a diferenciação de serviços no controle de admissão.

Conclui-se que o mecanismo de controle de admissão proposto consegue efetivamente controlar a sobrecarga do servidor SWDS, mantendo a utilização sempre abaixo do limiar desejado e propiciando um atendimento aceitável para todas as classes de serviço. A escolha da utilização do *cluster* como referência para o controle de admissão tem con-

seqüências imediatas sobre o tamanho das filas, portanto sobre o tempo de resposta médio das requisições, o que contribui para a melhoria de desempenho do sistema como um todo.

## 7. Conclusões

Este artigo propôs uma arquitetura para o fornecimento de serviços diferenciados em servidores web, denominada SWDS, que permite oferecer diferentes níveis de serviço a diferentes classes de usuários. O modelo apresentado é uma evolução em relação às arquiteturas de servidores web tradicionalmente usadas, que atendem as requisições dos clientes segundo uma disciplina FIFO, sem considerar as particularidades de cada grupo de aplicação ou usuários. A arquitetura foi validada por meio de simulação, sendo a carga de trabalho gerada a partir de *logs* de acesso a servidores web.

Este trabalho também descreve os resultados obtidos com dois algoritmos de diferenciação de serviços implementados na arquitetura do servidor SWDS, ambos baseados em prioridades. O mecanismo rigoroso revelou-se uma abordagem eficiente para o fornecimento de serviços diferenciados em servidores web, funcionando satisfatoriamente com diferentes níveis de carga de trabalho e configurações do servidor. Os experimentos mostraram, contudo, que este mecanismo opera melhor quando o sistema não está muito sobrecarregado e também que é importante atentar para que a distribuição dos clientes pelas classes de serviço seja adequada, a fim de evitar a monopolização dos recursos pelas requisições de maior prioridade.

Nosso mecanismo de prioridades adaptativo, uma solução inovadora, utiliza um parâmetro de *look-ahead* nas filas de espera do *cluster* que determina o nível de priorização empregado pelo sistema, conferindo uma maior ou menor importância às requisições de alta prioridade. Para valores elevados do *look-ahead*, este algoritmo comporta-se como o caso rigoroso e, para valores próximos de um, as requisições são atendidas segundo sua ordem de chegada, anulando-se a diferenciação de serviços. O mecanismo adaptativo evita os problemas observados no esquema rigoroso e permite que o administrador do sistema tenha um controle mais efetivo sobre o atendimento dispensado aos clientes. Com isso, introduz características de adaptabilidade no servidor SWDS, tornando-o capaz de melhor responder às variações na carga de trabalho e nos percentuais de clientes alocados às classes de serviço. Este mecanismo mostrou-se mais apropriado para um ambiente altamente dinâmico, como é o caso da Web.

Finalmente, foi implementado um mecanismo de controle de admissão no servidor SWDS, que emprega uma média exponencialmente ponderada da utilização do *cluster* para orientar as decisões de descarte. Este mecanismo consegue manter a carga de trabalho sempre abaixo do limiar especificado, evitando aceitar no sistema requisições que não poderão ser atendidas. A variação do peso da média ponderada permite, ainda, ajustar a sensibilidade do servidor SWDS a mudanças no perfil da carga de trabalho. Os experimentos realizados permitem concluir que a utilização de um Controle de Admissão é fundamental para o fornecimento de serviços diferenciados na Web com uma melhor qualidade, pois torna a diferenciação de serviços ainda mais eficiente, sendo possível fornecer um serviço com uma característica mais estável e, principalmente, mais justa. Assim, evita-se penalizar ou favorecer demasiadamente uma ou outra classe de clientes.

Como trabalhos futuros, pretende-se prosseguir com a implementação de novos algoritmos de diferenciação de serviços, dentre eles o algoritmo *Weighted Fair Queueing* (WFQ) e suas variantes, além de tornar o algoritmo PRIAdap aqui proposto auto-adaptativo. Objetiva-se também estudar o comportamento dos algoritmos em cenários com mais de duas classes de serviço. Outras áreas de pesquisa em andamento são a introdução da funcionalidade de Negociação no módulo de controle de admissão e a

realização de controle de admissão baseado em sessões HTTP (*Session Based Admission Control* – SBAC).

## Referências


Allen, A. O. (1990). *Probability, Statistics and Queueing Theory with Computer Science Applications*. Academic Press, 2 edition.

Arlitt, M. and Jin, T. (1999). Workload characterization of the 1998 World Cup web site. Technical Report HPL-1999-35, HP Laboratories.

Arlitt, M. F. and Williamson, C. L. (1996). Web server workload characterization: the search for invariants. In *Proceedings of ACM SIGMETRICS '96*, pages 126–37.

Cardellini, V., Casalicchio, E., Colajanni, M., and Mambelli, M. (2001). Web switch support for differentiated services. *ACM Performance Evaluation Review*, 29(2):14–19.

Casalicchio, E. and Colajanni, M. (2000). Scalable web cluster with static and dynamic contents. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER '00)*.

Chen, X. and Mohapatra, P. (1999). Providing differentiated services from an Internet server. In *Proceedings of the IEEE International Conference on Computer Communications and Networks*, pages 214–217.

Crovella, M. E. and Bestavros, A. (1997). Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–46.

Dovrolis, C. and Ramanathan, P. (1999). A case for relative differentiated services and the proportional differentiation model. *IEEE Network*, 13(5).

Hu, Y., Nanda, A., and Yang, Q. (1999). Measurement, analysis and performance improvement of the Apache web server. In *Proceedings of the 18th IEEE International Performance, Computing and Communications Conference*.

Iyer, R., Tewari, V., and Kant, K. (2000). Overload control mechanisms for web servers. In *Proceedings of the Workshop on Performance and QoS of Next Generation Networks*.

Kant, K. and Mohapatra, P. (2000). Scalable Internet servers: Issues and challenges. *ACM SIGMETRICS Performance Evaluation Review*, 28(2):5–8.

Menascé, D. A. and Almeida, V. A. F. (2003). *Planejamento de Capacidade para Serviços na Web: Métricas, modelos e métodos*. Campus.

Stallings, W. (2002). *High-Speed Networks and Internets: Performance and Quality of Service*. Prentice Hall, 2. edition.

Teixeira, M. M., Santana, M. J., and Santana, R. H. C. (2004). Using adaptive priority scheduling for service differentiation in QoS-aware web servers. In *Proceedings of the IEEE International Performance, Computing and Communications Conference. Workshop on End-to-End Service Differentiation (IEEE IPCCC 2004)*.

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

Welsh, M. and Culler, D. (2003). Adaptive overload control for busy internet servers. In *Proceedings of the 4th USENIX Conference on Internet Technologies and Systems (USITS 2003)*.