Uma Abordagem de Diferenciação de Serviços em Servidores Web com Mapeamento PHB AF

Tiago Semprebom[1], **Rômulo de Oliveira**[1], **Carlos Montez**[1]

[1]Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PPGEAS)
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Santa Catarina – SC – Brasil.

Abstract. *This paper proposes an approach for service differentiation in web servers that preserves the end-to-end mapping in conformance with PHB AF group. This novel approach uses dynamic scheduling techniques and selective discard of requests to achieve a proportional service differentiation. A prototype was built on Apache web server and experiments were carried out aiming to evaluate the approach in front of other traditional techniques of services differentiation.*

Resumo. *Neste artigo é proposta e implementada uma abordagem para diferenciação de serviços em servidores web que preserva o mapeamento fim a fim do grupo PHB AF. A abordagem, de caráter inovador, implementa a diferenciação proporcional de serviços através de técnicas de escalonamento dinâmico e descartes seletivos de requisições. Um protótipo foi construído sobre um servidor web Apache e foram realizadas medições objetivando avaliar a abordagem frente a outras técnicas tradicionais de diferenciação de serviços.*

1. Introdução

A Internet vem experimentado nos últimos anos um grande crescimento impulsionado, principalmente, pelo surgimento da *web* e pelo aumento do número de provedores comerciais. Entretanto, a Internet e seus protocolos originalmente não foi concebida para o uso verificado atualmente, nem para suportar a quantidade de carga que lhe é imposta.

Serviços oferecidos pela Internet usualmente baseiam-se no modelo de melhor esforço. Segundo esta política, todo o tráfego é tratado de maneira uniforme, sem qualquer tipo de diferenciação ou priorização [Henriksson et al. 2004, Vasiliou and Lutfiyya 2000, Kang et al. 2003, Rashid et al. 2005]. Porém, nem todo tipo de fluxo da Internet possui os mesmos requisitos de qualidade de serviço (QoS), existindo, portanto, a necessidade de priorizar alguns tipos de tráfego [Dovrolis and Ramanathan 1999, Joutsensalo et al. 2003, Chen and Mohapatra 2003, Tham and Subramaniam 2002, Ferrari 2000, Murta and Corlassoli 2003].

Considerando a inadequação do modelo de melhor esforço oferecido pela Internet, existem algumas propostas de fornecimento de QoS às aplicações. Estas propostas são elaboradas sob a coordenação da IETF *(Internet Engineering Task Force)*, a qual vem buscando o desenvolvimento de padrões abertos aplicados à infra-estrutura da rede, como os da arquitetura DiffServ [Blake et al. 1998]. Entretanto, pouco adianta o emprego dessas soluções na infra-estrutura da rede, caso as aplicações nos pontos finais também não se adequem a esta abordagem.

Na Internet os servidores *web* são os responsáveis pelo atendimento às requisições HTTP dos usuários, processadas nos elementos finais. Por conseguinte, recentemente vêm surgindo diversas pesquisas que buscam oferecer qualidade de serviço nesses servidores [Abdelzaher and Bhatti 1999, Anderson et al. 2003,

Rashid et al. 2005, Semprebom et al. 2006, Vasiliou and Lutfiyya 2000]. Algumas dessas pesquisas propõem agrupar as requisições *web* em diferentes classes de serviço, e implementar políticas de atendimento diferenciado a essas classes, prevendo, inclusive, controles de admissão e descartes de ativações.

Neste artigo é proposta, e avaliada através de um protótipo, uma abordagem para diferenciação de serviços em servidores *web* com mapeamento do grupo PHB AF *(Assured Forwarding)*, o qual foi originalmente proposto na arquitetura DiffServ com objetivo de fornecer qualidade de serviço na infra-estrutura de rede. O modelo de provisão de QoS adotado prevê a existência de classes de serviço. Com fim de manter conformidade com a definição do grupo PHB AF – facilitando um possível mapeamento do comportamento especificado na infra-estrutura da rede –, cada uma das classes especificadas implementa até três precedências de descartes diferentes.

O restante deste artigo está organizado da seguinte maneira: a Seção 2 traz um levantamento sobre a arquitetura de Serviços Diferenciados. Na Seção 3 são apresentados alguns modelos de qualidade de serviço em servidores *web*, destacando as principais características de cada um destes. A Seção 4 apresenta alguns trabalhos relacionados. Na Seção 5 são introduzidos um modelo de diferenciação de serviços em servidores *web* com descarte seletivo. A Seção 6 apresenta a implementação do modelo em um servidor *web* Apache. Na Seção 7 são discutidos os resultados experimentais deste trabalho e as métricas utilizadas para avaliar os modelos apresentados. Por fim, na Seção 8 destacam-se as principais conclusões deste trabalho, juntamente com as perspectivas futuras.

2. Arquitetura de Serviços Diferenciados

A arquitetura de serviços diferenciados (DiffServ) [Blake et al. 1998] já está bem consolidada no meio acadêmico e apresenta, hoje, produtos comerciais. A escalabilidade dessa arquitetura vem do fato que, apesar das características ou do comportamento desejado serem aplicados em cada nodo, as funções são implementadas apenas nos nodos de borda da rede. Este comportamento é denominado *Per-Hop Behavior* (PHB), onde para cada conjunto de tráfegos *(aggregate traffic)* – o qual foi previamente marcado utilizando o campo DS existente no cabeçalho do pacote – um tratamento diferenciado será dado.

Desde a publicação das especificações desta arquitetura, alguns PHBs padrões já foram definidos, entre eles o PHB *default*, PHB AF *(Assured Forwarding)* e PHB EF *(Expedited Forwarding)*. Neste artigo, o interesse se situa no padrão de grupo PHB AF [Heinanen et al. 1999]. A especificação deste modelo define classes, sendo que cada uma contem três precedências de descarte de pacotes: baixa, média e alta. Em cada classe AF busca-se o fornecimento de uma quantidade mínima de largura de banda e *buffering*. Os valores numéricos recomendados para uso no AF são identificados na Tabela 1. Os três primeiros bits identificam a classe de transmissão e os três últimos a precedência de descarte.

Tabela 1. Tabela de classificação e descarte do PHB AF.

Precedência	AF1	AF2	AF3	AF4
Baixa	001010	010010	011010	100010
Média	001100	010100	011100	100100
Alta	001110	010110	011110	100110

3. Modelos de Qualidade de Serviço em Servidores Web

O crescimento do uso de aplicações que utilizam a Internet como meio de tráfego de informações (ex. *e-commerce, e-banking*), e o surgimento de aplicações com diferentes

exigências de qualidade de serviço *(ex. VoIP, IPTV)*, têm impulsionado o desenvolvimento de novas políticas de atendimento às requisições de usuários mais eficientes e com suportes a QoS. O servidor *web* é o elemento principal no armazenamento e processamento destas requisições. Um servidor *web* congestionado, por exemplo, resulta em tempos de resposta inadequados às requisições de usuários. Clientes que aguardam um longo período de atendimento às suas requisições certamente abandonarão o *site* em questão. Recursos alocados nos sistemas finais e na infra-estrutura da rede são desperdiçados, causando um grande prejuízo ao sistema.

3.1. Modelo de Melhor Esforço

A maioria dos servidores *web* utilizam políticas de atendimento às requisições de seus usuários baseadas no modelo de melhor esforço, onde as requisições são atendidas por sua ordem de chegada (FIFO). Este modelo de serviço não oferece qualquer esquema de provimento de qualidade de serviço para seus clientes. A Figura 1 apresenta os três elementos principais na composição do modelo de melhor esforço: as requisições de chegada enviadas pelos clientes, uma fila FIFO para armazenamento das requisições e finalmente o servidor *web* responsável pelo processamento das requisições e envio das respostas.

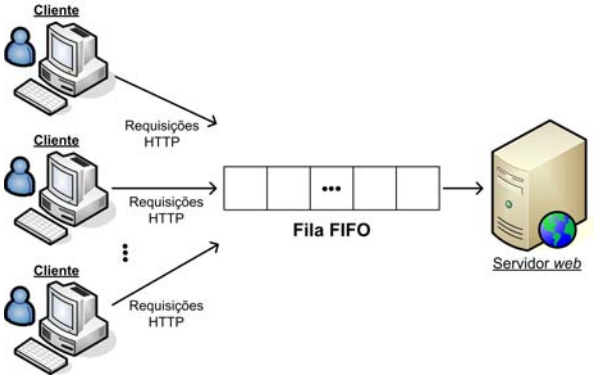

Figura 1. Modelo de provimento de QoS FIFO.

Os servidores baseados segundo o modelo de melhor esforço não apresentam mecanismos de controle de admissão. Portando, se as requisições dos clientes são inseridas na fila do servidor *web* mais rapidamente do que são removidas para processamento, ocorre uma condição de sobrecarga, resultando em atrasos nos tempos de resposta das requisições e rejeições de requisições devido a *timeouts*.

3.2. Modelo Básico de Diferenciação de Serviços

No modelo de diferenciação de serviços básico, as requisições são agrupadas em classes de serviço com prioridades diferentes. O servidor sempre retira da fila para processamento as requisições pertencentes as filas mais prioritárias, servindo posteriormente as requisições com prioridades inferiores. A Figura 2 apresenta o modelo básico de diferenciação de serviços, onde foram classificadas as requisições em categorias, representadas por tipos de metais, considerando o nível de serviço oferecido pelo servidor, proporcional ao valor intrínseco do metal. Cada uma das filas, por sua vez, atende suas requisições segundo uma política FIFO. Neste modelo estático, as requisições menos prioritárias serão atendidas apenas se não houver mais requisições na fila de alta prioridade. Novamente, como no modelo de melhor esforço, não existe mecanismo de controle de admissão no modelo de diferenciação básico de serviços.

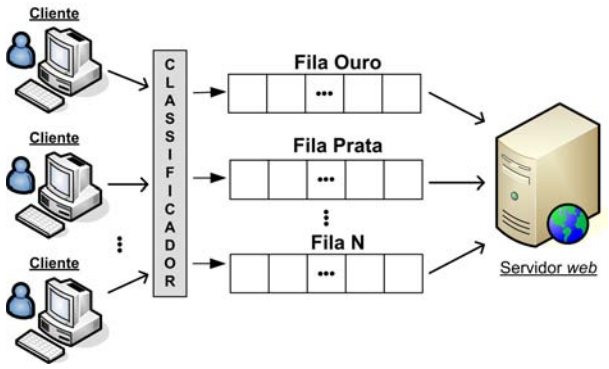

Figura 2. Modelo estático de diferenciação de serviços.

Como as requisições das classes menos prioritárias só são atendidas caso não haja requisições de classes mais prioritárias, este tipo de abordagem pode nunca atender uma requisição menos prioritária, levando essa classe a uma situação de inanição (ou *starvation*). Para ilustrar essa situação, experimentos foram realizados em um servidor *web* Apache[1] (Figura 3). Nestes experimentos foram utilizados quatro classes de serviço: Diamante, Ouro, Prata e Bronze, em ordem decrescente de prioridade, respectivamente. O eixo x representa a passagem tempo, enquanto o eixo y representa a carga de utilização do servidor.

Este servidor recebeu por algum tempo, requisições dos quatro tipos de classes de serviços. Em um primeiro momento, enquanto o servidor encontrava-se com baixa carga, entre os instantes 1 e 4, todas as requisições puderam ser atendidas. A partir do instante 5, agora com o servidor sobrecarregado, apenas as requisições da classe Diamante passaram a ser atendidas. Após o instante 21, quando cessaram as requisições da classe Diamante, passaram a ser executadas as requisições da classe Ouro. Somente após o instante 38, as requisições da classe Prata foram executadas; e as requisições pertencentes a classe Bronze puderam a ser atendidas somente após o tempo 57, quando não havia mais requisições das classes mais prioritárias.

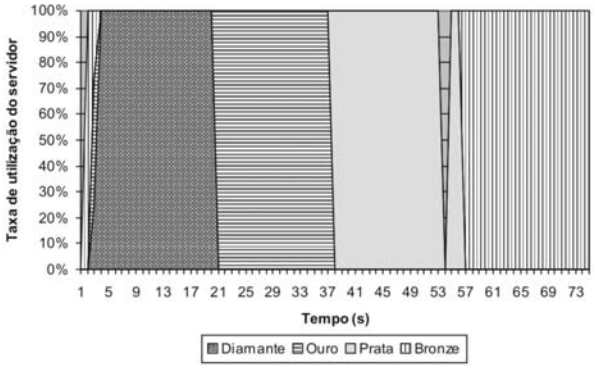

Figura 3. Experimento com prioridades estáticas.

3.3. Modelo com Controle de Admissão e Escalonamento Dinâmico de Requisições

Em ambientes com características dinâmicas como a Internet, onde o comportamento de requisições para servidores *web* não pode ser previamente antecipado, faz-se necessária a utilização de mecanismos de controle de admissão [Anderson et al. 2003] e de escalonamento dinâmico das requisições (Figura 4). A necessidade de escalonamento dinâmico

[1] www.apache.org

de requisições decorre da característica apresentada na Figura 3 quando se emprega abordagens estáticas em cargas dinâmicas (como é o caso de clientes *web*).

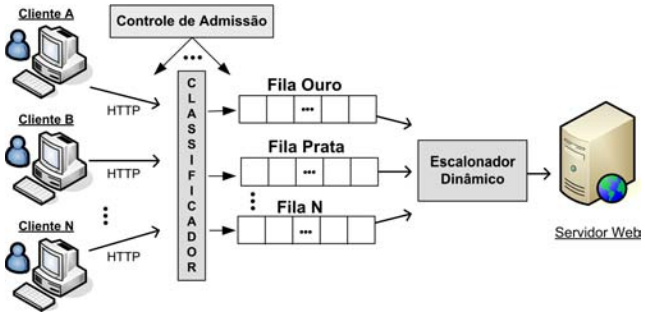

Figura 4. Modelo com controle de admissão e escalonamento dinâmico.

O controle de admissão pode ser estático (antes do módulo de classificação de requisições) ou seletivo (dinâmico, depois do classificador). Mecanismos de controle de admissão estáticos usualmente são ativados com base na taxa de ocupação da fila de requisições no servidor. Este monitoramento da taxa de ocupação busca antecipar condições de sobrecarga no servidor, e descartar requisições, de forma que as requisições já aceitas no servidor não sejam prejudicadas pelas requisições que estão chegando e que ainda não foram aceitas. Quando é detectada uma condição de sobrecarga, as próximas requisições que chegarem ao servidor são descartadas, evitando que este fique ainda mais sobrecarregado, e comprometa a execução das requisições que já foram aceitas.

Um problema verificado no modelo de controle de admissão estático é que a inferência sobre as condições de carga do servidor é usualmente feita sobre um teste simples da taxa de ocupação das filas. Além disso, a requisição descartada é sempre a última que chegou, ignorando-se completamente a classe que a requisição pertence; e ignorando-se também como o servidor atendeu as últimas requisições de cada classe.

O mecanismo de controle de admissão com descartes seletivos possui características dinâmicas, podendo implementar diferentes políticas de descarte. O controle de admissão é realizado após o módulo de classificação de requisições no servidor. Desta forma, os descartes ocorrem segundo alguma política baseada na classe da requisição e no histórico recente de descartes ocorridos na classe.

O escalonamento dinâmico de requisições pode ser implementado de diversas formas. Pode-se, por exemplo, controlar dinamicamente os recursos alocados para uma classe de requisições. Como exemplo, é possível controlar o número de *threads* que atende a cada classe (em uma abordagem com *pool de threads*), alocando mais *threads* para as classes que necessitam de mais qualidade de serviço naquele determinado momento. Contudo, a abordagem mais usual é lidar dinamicamente com as prioridades de cada classe [Tham and Subramaniam 2002, Joutsensalo et al. 2003]. Conceitualmente, pode-se definir que cada fila associada a uma classe de requisições possui uma prioridade, e essa prioridade é alterada no decorrer do tempo. Um exemplo dessa abordagem o PCV (*Proportional Cumulative Value Attribution*) [Semprebom et al. 2006].

4. Trabalhos Relacionados

Recentemente, vêm surgindo diversos trabalhos que atuam de forma complementar aos serviços oferecidos na infra-estrutura da rede [Lu et al. 2001, Chen and Mohapatra 2003, Ferrari 2000, Robertson et al. 2004, Semprebom et al. 2006]. Esses trabalhos buscam

oferecer qualidade de serviço nos elementos finais da rede – os servidores *web* – responsáveis pelo armazenamento do conteúdo e processamento das requisições dos clientes.

O trabalho introduzido por Mohapatra [Chen and Mohapatra 2003] propõe um algoritmo de escalonamento dinâmico *(DWFS - Dynamic Weighted Fair Sharing)* para controlar situações de sobrecarga em servidores *web*. O algoritmo proposto baseia-se na observação do uso de sessões HTTP, onde o algoritmo evita o processamento de requisições pertencentes a uma sessão que possivelmente será descartada em um futuro próximo (ultrapassaram seus valores de *timeout*). A avaliação do algoritmo foi realizada a partir de modificações realizadas no servidor *web* Apache. A geração de carga foi realizada utilizando o gerador de carga sintética *WebStone*.

O artigo apresentado em [Ferrari 2000] analisa os efeitos da diferenciação fim a fim utilizando dois algoritmos: PQ *(Priority Queuing)* e WFQ *(Weighted Fair Queuing)*. O modelo foi avaliado utilizando pacotes EF *(Expedited Forwarding)* previamente marcados. A métrica utilizada na avaliação do modelo foi o atraso dos pacotes.

O modelo desenvolvido em [Semprebom et al. 2006] propõe um modelo de provimento de qualidade de serviço em servidores *web* baseado em diferentes classes de serviços. Para lidar com situações de sobrecarga, o modelo prevê a utilização de técnicas de escalonamento adaptativo com prioridades dinâmicas e computação imprecisa (múltiplas versões para páginas *web*). A estrutura proposta no modelo permite que as requisições recém-chegadas ao servidor sejam classificadas e encaminhadas a filas de prioridades diferenciadas, conforme o nível de serviço. O modelo contempla também um mecanismo de controle de admissão estático que atua em situações de sobrecarga severas, descartando requisições, evitando assim que o sistema fique ainda mais sobrecarregado.

Quando a taxa de chegada de requisições *web* cresce acima da capacidade, as filas e os tempos de resposta aumentam demasiadamente. Um usuário que experimenta longos tempos de resposta em atendimento a uma requisição, abandonará o *site*, desperdiçando assim todos os recursos utilizados do sistema. Portando, a adição de mecanismos de controle de admissão em servidores *web* é de grande importância e tem motivado diversos grupos de pesquisa a atuarem nesse sentido.

O trabalho apresentado em [Robertson et al. 2004] introduz um mecanismo de controle de admissão utilizando técnicas de controle não linear. Segundo o autor uma importante escolha no desenvolvimento do mecanismo de controle de admissão é a variável a ser controlada, ou seja, qual dos gargalos do sistema será mensurado (fila, processador, acesso a disco, tamanho da fila, etc). O modelo foi avaliado utilizando o servidor *web* Apache e um módulo de controle de admissão externo. A geração da carga HTTP foi realizada através do gerador de carga sintética *S-Client*.

Diferentemente dos trabalhos citados acima, este trabalho implementa um mecanismo de controle de admissão com descarte seletivo, utilizando o mapeamento do grupo PHB AF.

5. Modelo de provisão de diferenciação de serviços PBH AF

O Modelo adotado neste trabalho admite módulos de controle de admissão e escalonamento de requisições, ambos efetuados dinamicamente. É o mesmo modelo representado na Figura 4, contudo com o controle de admissão atuando somente após a classificação da requisição (descartes seletivos). O objetivo é desenvolver uma aborda-

gem para diferenciação de serviços em servidores *web* seguindo o mapeamento do grupo PHB AF, especificado pela IETF [Blake et al. 1998].

No modelo proposto, podem existir até quatro tipos de classes de serviço, cada uma com até três precedências de descarte. Esse mapeamento das precedências de descartes das requisições baseia-se na Tabela de precedência de descarte do PHB AF (Tabela 1). Segundo este modelo, em situações de sobrecarga, antes de as requisições serem atendidas, verifica-se a precedência de descarte da classe a qual a requisição pertence. Caso uma classe esteja programada com baixa precedência de descarte esta requisição dificilmente será descartada, caso contrário a probabilidade de descarte é alta.

A Tabela 2 apresenta um exemplo de uma aplicação deste mapeamento. São apresentadas quatro classes de serviço. A classe Prata, por exemplo possui três valores de precedência de descarte (baixa: 10%, média: 30% e alta: 80%), caso uma requisição da classe Prata com baixa precedência de descarte chegue ao servidor em uma situação de sobrecarga, esta terá baixa probabilidade de ser descartada (no caso, 10% das requisições poderão ser descartadas na sobrecarga), enquanto o mesmo ocorrendo com uma requisição pertencente a alta precedência de descarte certamente será descartada (no caso, 80% de probabilidade de descarte). Para as demais classes de serviço foram adotadas apenas uma precedência (precedência de descarte média: 30%).

Tabela 2. Um mapeamento PHB em servidor web com diferenciação de serviço.

Precedência	Diamante	Ouro	Prata	Bronze
Baixa	30%	30%	10%	30%
Média	30%	30%	30%	30%
Alta	30%	30%	80%	30%

5.1. Atribuição de deadlines e valores cumulativos a ativações

Considerando que os clientes (ex. navegadores *web)* não irão aguardar as respostas de suas requisições indefinidamente, é possível assumir uma condição faltosa, quando uma requisição ultrapassar um valor de tempo máximo para ser atendida, esse valor é concebido como *deadline soft*, ou seja um valor de tempo, a partir do qual o benefício oferecido decresce com o passar do tempo (ver Figura 5).

Uma requisição atendida com tempo de resposta considerado adequado – ou seja, dentro do seu deadline – adiciona um benefício ao sistema. Já uma resposta que ultrapassa seu deadline tem seu valor de benefício ao sistema reduzido podendo até mesmo não adicionar qualquer benefício ao sistema. Por conseguinte, adotamos, neste trabalho, calcular o valor de benefício agregado às classes de serviço conforme os tempos de respostas das requisições. Esse valor de benefício agregado é denominado valor cumulativo. O valor cumulativo – originalmente introduzido por [Baruah et al. 1992] em um contexto de escalonamento tempo real – é uma métrica utilizada para mensurar o tempo despendido por um servidor na execução de uma requisição.

Na abordagem empregada, os valores cumulativos de cada classe são atribuídos com base nos tempos de resposta de cada requisição, e podem receber valores entre [0,1] (0% a 100%), proporcionais ao benefício recebido na ativação. Neste sentido, uma função benefício [Mercer 1992, Jensen 1997] é utilizada para modelar o valor cumulativo a ser atribuído para as classes de serviço em função dos tempos de resposta das requisições. A Figura 5 apresenta a função benefício utilizada. São adotados dois deadlines: *d1 – deadline soft*, a partir do qual o benefício agregado ao sistema se reduz com o tempo e *d2*

– *deadline firm*, a partir do qual não há mais benefício ao sistema. Para cada classe, as execuções das requisições irão receber um valor cumulativo, segundo a seguinte fórmula:

- $V = 1$ se a requisição é executada antes do *deadline soft*.
- $V = \left(\frac{d2 - t}{d2 - d1} \right)$ se a requisição é executada entre os *deadlines* d1 e d2.
- $V = 0$ se a requisição é executada após o *deadline firm* d2.

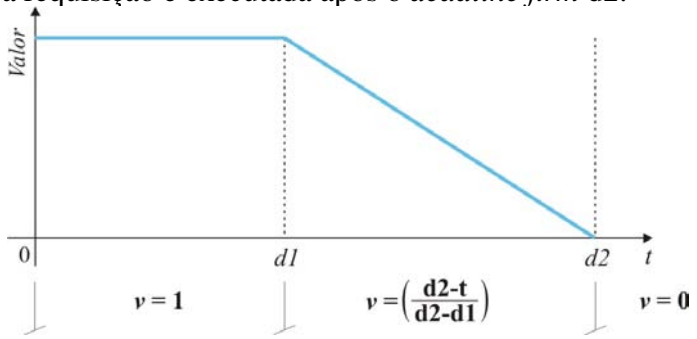

Figura 5. Atribuição de valores cumulativos utilizando função benefício.

Pode-se observar que com um tempo de resposta entre *d1* e *d2*, à medida que este se aproxima de *d2*, o valor cumulativo é reduzido até alcançar o valor zero. Após esse tempo limite (*deadline firm*), o benefício oferecido ao sistema permanece em zero.

Em um modelo que admite descartes de ativações, como o adotado neste trabalho, é importante garantir que o valor cumulativo não reflita apenas os tempos de resposta. Como um exemplo extremo, considere uma abordagem de atendimento a requisições *web* que simplesmente descarte todas as requisições. Essa estratégia iria obter tempos de resposta excelentes dando a falsa impressão de ser uma boa abordagem. Por conseguinte, para o nosso modelo adicionamos a seguinte regra:

- $V = 0$ se a requisição é descartada antes de começar sua execução.

5.2. Algoritmo PCV com Descarte Seletivo

A inovação proposta neste artigo está centrada na heurística de escalonamento PCV com Descarte Seletivo – ilustrada na Listagem 1 –, e que implementa o modelo de diferenciação de serviços segundo o mapeamento PHB AF em servidores *web*.

Para a implementação do modelo, considera-se que cada classe de serviço possui um Valor de Qualidade (Q) associado. Esse valor é uma percentagem atribuída estaticamente pelo operador da rede, e que representa a qualidade almejada para aquela classe, proporcionalmente às outras classes. A soma dos valores de qualidade de todas as classes deve totalizar 1 (100)%. A heurística atua continuamente buscando manter os valores cumulativos V de cada classe de serviço próximas aos valores de qualidade especificados (busca manter V igual a Q). O trabalho [Montez and Fraga 2002] discute como atribuir valores de qualidade (Q) às classes de serviço.

Na Listagem 1, a inicialização da heurística é representada pelas linhas 1-4, onde todas as classes recebem a mesma prioridade (a menor prioridade do sistema). É possível observar três fluxos de execução no laço de execução principal: O primeiro fluxo (linhas 6-13) é responsável por receber as requisições, classificá-las e detectar, através do mecanismo de controle de admissão, situações de sobrecarga e nesse caso descartar as requisições recém chegadas, respeitando o mapeamento do PHB AF. Cada requisição descartada é contabilizada como uma perda de *deadline firm* para a sua respectiva classe de serviço (ou seja, benefício zero).

O segundo fluxo de execução (linhas 14-18) é responsável por retirar as requisições de suas respectivas filas. Neste fluxo de execução, a *thread* que irá executar a requisição receberá a mesma prioridade da classe.

No terceiro e último fluxo (linhas 19-23), quando uma requisição termina sua execução, cada classe recebe uma nova prioridade. Esta prioridade (prioridade dinâmica) é escolhida baseada no valor cumulativo das classes. A classe que estiver mais distante do valor de qualidade (Q) almejado receberá maior prioridade do sistema. Classes que possuam seu valor cumulativo maior ou igual ao valor de qualidade irão receber a menor prioridade do sistema (no caso, o valor zero).

Listagem 1 Heurística de escalonamento PCV com Descarte Seletivo.

```
 1: for all classe_j do
 2:     V_j ← Q_j
 3:     prio_j ← 0 {menor prioridade}
 4: end for
 5: loop
 6:     □ Evento 1: Chegada de uma nova requisição
 7:     if chegou requisição para uma classe_j then
 8:         if ocupação fila ≥ limite then
 9:             descarta requisição com a precedência de descarte PHB AF
10:         else
11:             enfileira requisição na fila de chegada da classe
12:         end if
13:     end if
14:     □ Evento 2: Existem requisições a serem executadas
15:     if ∃ requisição na cabeça da fila de requisicoes para classe_j then
16:         retira requisição de sua fila de chegada
17:         executa requisição com prio_j
18:     end if
19:     □ Evento 3: Término da execução de uma requisição
20:     if terminou execução da requisição X para uma classe_j then
21:         testa se perdeu deadline e calcula V_j
22:         atribui prioridades: prio_j ← max(0, (Q_j − V_j))
23:     end if
24: end loop
```

6. Implementação do Protótipo

Um protótipo construído sobre o Apache foi utilizado para avaliar a implementação do modelo proposto. O Apache é projetado para trabalhar com uma ampla variedade de plataformas e ambientes. Isto é possível devido à sua implementação modular. A versão 2 do Apache introduziu os modelos de processos MPM *(Multi-Processing-Modules)*, responsáveis por gerenciar as portas de comunicação, aceitar conexões e alocar processos ou *threads* para atendimento das requisições.

O MPM utilizado na implementação desse trabalho foi o MPM Worker (Figura 6). Este MPM implementa um servidor multi-processo multi-thread. Alterações feitas neste módulo apache, juntamente com a implementação de um módulo DSO *(Dynamic Shared Object)*, possibilitaram a implementação do modelo proposto. Neste trabalho adotou-se a última versão estável do Apache, versão 2.2. Essa nova versão oferece algumas

funcionalidades interessantes como a possibilidade de se utilizar a API padrão oferecida pelo Apache.

A classificação dos pedidos de conexão pode ser implementada baseada no endereço IP do cliente, e identificada pelo servidor através da extração dessa informação contida na mensagem HTTP enviada pelo cliente. Uma *thread* ouvinte aguarda novos pedidos de conexão vindos do serviço de comunicação TCP/IP (*A* na Figura 6) e os insere na fila de requisições (*B* na Figura 6), marca cada requisição com sua respectiva classe de serviço (ex. ouro, prata, etc) e atribui à requisição um valor de *deadline*.

O controle de admissão também é implementado nessa fase com base no tamanho da fila de requisições, respeitando a precedência de descarte do PHB AF. Alguns pedidos de conexão são descartados, evitando que o servidor fique ainda mais sobrecarregado.

As *threads* de atendimento de requisições (*C* na Figura 6) retiram as requisições da fila baseadas nas classes das requisições e nas prioridades de cada classe. Estas requisições então são processadas pelos módulos manipuladores.

As trocas de informações entre os módulos MPM e o módulo manipulador são feitas utilizando áreas de memória disponibilizadas pelo Apache, conhecidas como *pools* e através do uso de memória compartilhada. Para a compilação e execução do módulo DSO desenvolvido utilizou-se a ferramenta de construção de módulos dinâmicos oferecida pelo Apache 2.2, apxs *(Apache eXtenSion tool)*.

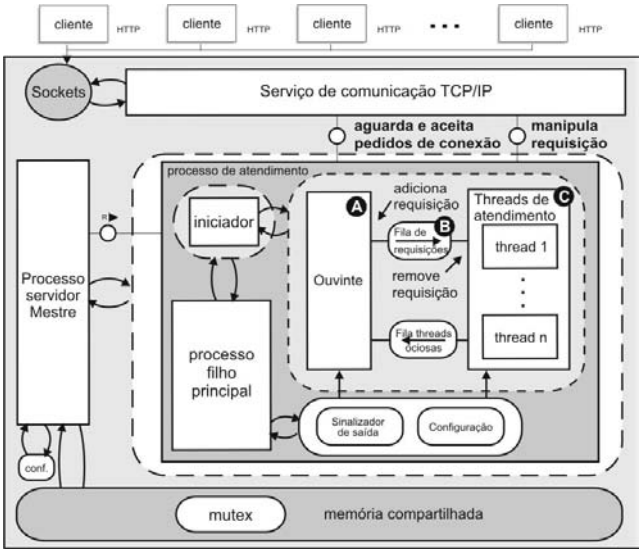

Figura 6. Configuração do MPM Worker.

7. Resultados Experimentais

A Figura 7 apresenta o ambiente de medições. Na formulação do ambiente de testes foram utilizados computadores Pentium IV 3.2 GHz, 1024 MB RAM e sistema operacional Ubuntu Gnu/Linux versão 6.10 *Kernel* 2.6.17.

Na geração de carga dos clientes utilizou-se o gerador de carga sintética *httperf*[2] com número de 400 requisições por cliente a uma taxa de 25 requisições por segundo e *timeout* de 100 segundos. Neste trabalho, o tamanho da fila do servidor é de 1000 requisições, e considera-se que o sistema encontra-se em sobrecarga quando a ocupação

[2]Ferramenta desenvolvida nos laboratórios da HP – www.hpl.hp.com/research

da fila é superior a 600 requisições (60%). Foram usados valores de *deadline soft (d1) e firm (d2)*, 5 e 20 segundos, respectivamente.

O modelo com o algoritmo PCV com Descarte Seletivo foi comparado com outros três modelos: Melhor Esforço, Prioridades Estáticas e PCV Estático. O modelo de Melhor Esforço foi usado para comparação como um *baseline*, pois é o modelo onde não há qualquer oferecimento de QoS. O modelo PCV Estático é implementado pelo algoritmo apresentado na Listagem 1, contudo, sem o descarte seletivo. Neste caso, os descartes ocorrem antes da classificação dos pacotes.

Nos modelos de Melhor Esforço e Prioridades Estáticas não há controle de admissão, portanto, sem tratamento adequado às situações de sobrecarga. No PCV Estático os descartes ocorrem quando o sistema se encontra em sobrecarga (quando o tamanho da fila de requisições do servidor ultrapassa 60%). Finalmente, no PCV com Descarte Seletivo os descartes ocorrem em conformidade com precedências de descarte das classes, previamente estipuladas pelo operador de rede, conforme descrito na seção 2.

Adotou-se para os experimentos a existência de quatro classes de serviços, e para os modelos PCV estático e PCV com Descarte Seletivo considerou-se os seguintes valores de qualidade (Q): 50%, 30%, 15% e 5% para as classes Diamante, Ouro, Prata e Bronze, respectivamente. Assumiu-se, também que cada classe possui apenas uma precedência de descarte: 30%, 30%, 10% e 30%, para as classes Diamante, Ouro, Prata e Bronze, respectivamente. Ou seja, cada requisição Prata, independente de sua precedência de descarte – baixa, média ou alta –, terá 10% de probabilidade de ser descartada na sobrecarga.

Neste trabalho foram utilizadas três métricas para avaliar a qualidade de serviço nos modelos apresentados: número de descartes por classe de serviços, tempo médio de espera no atendimento das requisições e o valor cumulativo.

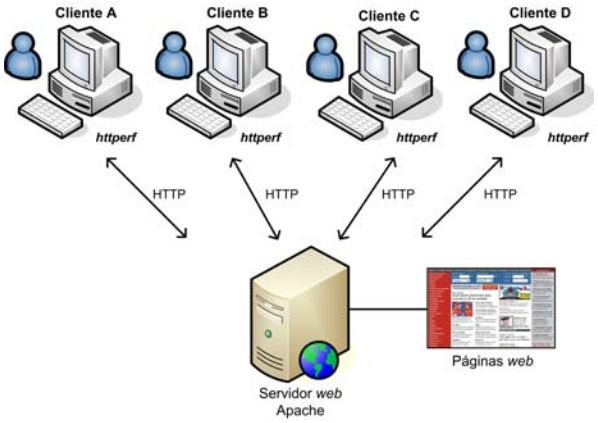

Figura 7. Topologia utilizada nas medições.

Os resultados dos tempos de resposta coletados nas medições referentes aos modelos de Melhor Esforço, Prioridades Estáticas e PCV com descartes estáticos são apresentados na Tabela 3. Pode-se verificar que os valores de tempo de resposta para a abordagem de melhor esforço foram muito próximos, não apresentando diferenciação de serviço entre as classes de serviço. No modelo de prioridades estáticas as classes mais prioritárias receberam os menores tempos de resposta (ex. classe Diamante), enquanto as classes de serviço menos prioritárias tiveram que aguardar longos períodos de tempo para serem atendidas. Na abordagem PCV utilizando controle de admissão estático, os tempos de resposta foram reduzidos, devido ao seu mecanismo de controle de admissão.

Tabela 3. Tempos de resposta e descartes de requisições por classes.

	Tempos resp.			Descartes		
Classe	**Melhor esforço**	**Prior. estática**	**PCV estático**	**Melhor esforço**	**Prior. estática**	**PCV estático**
Diamante	14325 ms	3866 ms	1021 ms	24.5%	25.8%	25.2%
Ouro	15024 ms	18900 ms	10236 ms	25.3%	23.5%	25.3%
Prata	14080 ms	35248 ms	22444 ms	24.2%	25.8%	24.6%
Bronze	14489 ms	49796 ms	33791 ms	26.0%	25.5%	24.9%

Em todos os modelos apresentados na Tabela 3, os descartes não apresentam diferenciação entre as classes. Este comportamento ocorre porque nenhuma das abordagens oferece a possibilidade de distinguir as requisições dos clientes antes delas serem descartadas.

Com o intuito de ampliar o modelo, oferecendo diferenciação entre os descartes das requisições, introduziu-se o modelo PCV com Descarte Seletivo, segundo mapeamento PHB AF. Na Tabela 4 são apresentados os tempos de resposta fim a fim e porcentagem de descartes por classes referentes às medições sobre este modelo. Para este, houve uma diferenciação proporcional de serviços e os valores de descartes obtidos ficaram bem próximos do especificado.

Tabela 4. PCV com Descarte Seletivo, mapeamento do PHB-AF.

Classe	**Tempos de resposta**	**Valor de Qualidade (Q)**	**Descartes obtidos**	**Descartes almejados**
Diamante	2390 ms	50%	29.4%	30%
Ouro	12234 ms	30%	29.3%	30%
Prata	21739 ms	15%	11.0%	10%
Bronze	33159 ms	5%	30.1%	30%

A Tabela 5 apresenta alguns dados coletados no servidor no modelo PCV com Descarte Seletivo. Os valores observados correspondem a um intervalo de 16 segundos, escolhido justamente no momento que o servidor verificou uma condição de sobrecarga, ou seja, quando o servidor começou a descartar requisições. Diferentemente do tempo de resposta fim a fim (Tabela 4) os valores de tempo de resposta apresentados agora refletem o tempo de processamento no servidor (incluindo o tempo de espera na fila de requisições), desconsiderando o tempo de propagação na rede. Os descartes são apresentados por classe de serviço.

Tabela 5. Valores cumulativos no intervalo de execução.

Classe	**Tempos de resposta**	**Descartes obtidos**	**Valor de Qualidade (Q)**	**Valor Cumulativo (V)**
Diamante	5500 s	29.5%	50%	49%
Ouro	11960 s	30.0%	30%	29%
Prata	24590 s	10,5%	15%	17%
Bronze	33880 s	29.5%	5%	6%

O valor cumulativo, cuja média é apresentada na última coluna da tabela, é uma métrica que consegue capturar o valor de qualidade oferecido pelo servidor, considerando não apenas os tempos de resposta, mas também levando em consideração os descartes de requisições. Os valores obtidos foram próximos aos valores de qualidade especificados. A Figura 8 apresenta o comportamento do Valor Cumulativo em um intervalo de tempo no qual o servidor se encontrava em uma situação de sobrecarga (Tabela 5).

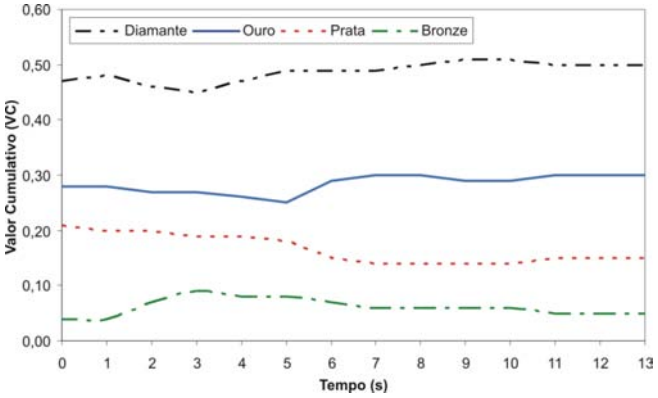

Figura 8. Comportamento do Valor Cumulativo.

Outros experimentos foram efetuados. Desta vez considerando diferentes precedências de descarte em uma única classe e avaliando o comportamento da heurística em uma situação de sobrecarga. Por exemplo, considerando a classe Prata com três diferentes precedências de descarte – alta: 80%, média: 30% e baixa: 10% – com requisições para essas classes geradas aleatoriamente em três computadores clientes, as porcentagens de descarte obtidas durante um período de sobrecarga para as requisições com precedências de descarte alta, média e baixa, foram, 77,4%, 24,4% e 9%, respectivamente.

Os resultados obtidos neste trabalho mostram que é possível implementar descartes seletivos em servidores *web*. Diferentemente da abordagem PCV com controle de admissão estático, no modelo PCV com Descarte Seletivo houve diferenciação também nos descartes das requisições e no atendimento às requisições.

8. Conclusões

Este artigo apresentou e avaliou as abordagens de melhor esforço, prioridades estáticas (ou absolutas), PCV com controle de admissão estático, e PCV com Descarte Seletivo em servidores *web*. Todas as abordagens citadas neste trabalho foram implementadas através da instrumentalização de um servidor *web* Apache. Os resultados foram obtidos através de medições, conseguindo mensurar melhor os *overheads* existentes no servidor *web*.

Foram definidas quatro classes de serviço para diferentes requisições. As métricas utilizadas para avaliar este trabalho foram: a quantidade de descartes das requisições, tempo médio de espera pelo atendimento das requisições e para as políticas PCV, adotou-se, também, o valor cumulativo. Esta métrica consegue, diferentemente do tempo de resposta, capturar valores para requisições descartadas.

Este trabalho também estendeu o modelo apresentado inicialmente em [Semprebom et al. 2006], adicionando um mecanismo de controle de admissão com descarte seletivo, utilizando o mapeamento do grupo PBH AF para o servidor *web*. Segundo este modelo, classes de serviço contêm precedências de descarte associadas, efetuando assim o descarte de forma diferenciada entre as classes.

Dentre as contribuições deste trabalho descata-se a implementação do modelo de diferenciação de serviços relativa (ou proporcional) [Dovrolis and Ramanathan 1999], onde são mantidos valores proporcionais de QoS entre as classes, independentemente da variação de carga de cada agregado. Faz-se uso também de abordagens de tempo real como funções benefício para melhor modelagem dos valores cumulativos e assim conseguir medir efetivamente a qualidade de serviço oferecida a cada classe e não somente tempos de resposta. Os autores deste trabalho não têm conhecimento de outros traba-

lhos que ofereçam QoS proporcional fim a fim, que permitam integrar atendimento de requisições web com abordagem PHB AF da infra-estrutura da rede.

Referências

Abdelzaher, T. F. and Bhatti, N. (1999). Web server qos management by adaptive content delivery. *IEEE Computer Networks Amsterdam, Netherlands*, 31(11–16):1563–1577.

Anderson, M., Kihl, M., and Robertsson, A. (2003). Modelling and design of admission control mechanisms for web servers using non-linear control theory. In *Proc. Proceedings of ITCom*.

Baruah, S., Koren, G., Mao, D., Mishra, B., Raghunathan, A., Rosier, L., Shasha, D., and Wang, F. (1992). On the competitiveness of on-line real-time task scheduling. *Journal Real-Time Systems*, 4(2):1573–1383.

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. (1998). An architecture for differentiated services. *IETF RFC 2475*.

Chen, H. and Mohapatra, P. (2003). Overload control in qos-aware web servers. *Comput. Networks*, 42(1):119–133.

Dovrolis, C. and Ramanathan, P. (1999). A case for relative differentiated services and the proportional differentiation model. *IEEE Network*, 13(5):26–34.

Ferrari, T. (2000). End-to-end performance analysis with traffic aggregation. *Computer Networks (Amsterdam, Netherlands: 1999)*, 34(6):905–914.

Heinanen, J., Baker, F., Weiss, W., and Wroclawski, J. (1999). Assured forwarding phb group. *IETF RFC 2597*.

Henriksson, D., Lu, Y., and Abdelzaher, T. (2004). Improved prediction for web server delay control. *In ECRTS´04 Procedings of the 16th Euromicro Conference on Real-Time Systens*, pages 61–68.

Jensen, D. E. (1997). Eliminating the 'hard'/'soft' real-time dichotomy. *Computing and Control Engineering Journal*, 8(1):15–19.

Joutsensalo, J., Hämäläinen, T., Pääkkönen, M., and Sayenko, A. (2003). Adaptive weighted fair scheduling method for channel allocation. *Communications, 2003. ICC '03. IEEE Int. Conference*, 1:228–232.

Kang, K. D., Son, S. H., and Stankovic, J. A. (2003). Differentiated real-time data services for e-commerce applications. *In Eletronic Commerce Research*, 3(1-2):113–142.

Lu, C., Abdelzaher, T. F., Stankovic, J. A., and SO, S. H. (2001). A feedback control approache for guaranteeing relative delays in web servers. In *7th IEEE (RTAS´S 2001)*, pages 51–62, Taipei, Taiwan.

Mercer, C. W. (1992). An introduction to real-time operating systems: Scheduling theory.

Montez, C. and Fraga, J. (2002). Implementing quality of service in web servers. In *IEEE SRDS 2002*, pages 32–40, Osaka, Japan.

Murta, C. D. and Corlassoli, T. P. (2003). Fastest connection first: A new scheduling policy for web servers. In *18th International Teletraffic Congress (ITC-18)*, Berlin, Germany.

Rashid, M. M., Alfa, A. S., Hossain, E., and Maheswaran, M. (2005). An analytical approach to providing controllable differentiated quality of service in web servers. In *IEEE Trans. Parallel Distrib. Syst.*, pages 1022–1033.

Robertson, A., Wittenmark, B., Kihl, M., and Andersson, M. (2004). Design and evaluation of load control in web server systems. In *Proceedings of the 2004 American Control Conference*, pages 1980–1985.

Semprebom, T., Oliveira, R., and Montez, C. (2006). Classes de serviço em servidores web apache através de escalonamento adaptativo e controle de admissão. In *XII WebMedia 2006*, pages 273–282, Natal, RN.

Tham, C. K. and Subramaniam, V. (2002). Integrating web server and network qos to provide end-to-end service differentiation. *ICON, 10th IEEE International Conference on Networks*, pages 389–394.

Vasiliou, N. and Lutfiyya, H. (2000). Providing a differentiated quality of service in a world wide web server. *SIGMETRICS Perform. Eval. Rev.*, 28(2):22–28.