UM ESTUDO COMPARATIVO DOS ALGORITMOS DE CONTROLE DE ADMISSÃO RED E *DROP TAIL* SOBRE UM MODELO DE SERVIDOR *WEB* DISTRIBUÍDO

A COMPARATIVE STUDY OF ADMISSION CONTROL ALGORITHMS RED AND *DROP TAIL* ON A DISTRIBUTED *WEB* SERVER MODEL

Dayse Silveira de Almeida
Ricardo Nogueira Figueiredo
Regina Helena Carlucci Santana
Marcos José Santana*

Resumo

Este artigo descreve e compara dois algoritmos de controle de admissão, Drop Tail e RED, implementados em um modelo de servidor Web com diferenciação de serviços. Duas classes de serviço foram usadas e uma carga de trabalho sintética em que se varia a taxa de chegada de requisições ao sistema. Verificaram-se dois pontos importantes: 1) independentemente da carga e do algoritmo de controle de admissão utilizado, a classe de maior prioridade obtém os melhores serviços, ou seja, menor número de descartes e menor tempo de resposta em relação à classe de prioridade inferior e; 2) o melhor aproveitamento do RED em relação ao Drop Tail, descartando-se menor número de requisições, independentemente da configuração utilizada.

Termos Gerais: Algoritmos. Desempenho. Experimentação.
Palavras-Chave: Controle de admissão. Servidor Web. Diferenciação de serviços.

Abstract

This paper describes and compares two algorithms for admission control, Drop Tail and RED, implemented on a Web server model with service differentiation. Two classes of service were used and a synthetic workload that varies the arrival rate of requests to the system. We verified two important points: 1) regardless of the load and admission control algorithm used, the highest priority class gets the best services, i.e., fewer discards and faster response time for a class of lower priority and; 2) RED has more advantage over the Drop Tail, discarding the least number of requests, regardless of the configuration used.

General Terms: Algorithms. Performance. Experimentation.
Keywords: Admission control. Web Server. Service differentiation.

---

* Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo. São Carlos – SP – Brasil, 13560-970. daysesa,ricardo,rcs,mjs@icmc.usp.br.

Introdução

O número de usuários da Internet cresceu após o final da década de 90, com o surgimento da *World Wide Web* (WWW), fazendo com que se tornasse uma grande fonte de informações e de serviços. A *Web* atualmente comporta serviços multimídia, transações bancárias e comerciais, e o serviço de melhor esforço (*Best Effort*) da Internet já não se mostra mais adequado para tais aplicações. Além disso, estas aplicações concorrem de forma igualitária com aplicações convencionais da Internet, em que a necessidade de serviço possui menor exigência, e podem ser atendidas com menor prioridade.

Para distinguir as aplicações que necessitam de melhores serviços, das aplicações que não necessitam, é preciso haver uma forma de diferenciação de serviços. Na camada de rede, existem duas especificações disponíveis, de maior importância, a arquitetura de Serviços Integrados (IntServ) [3] e a de Serviços Diferenciados (DiffServ) [2], para provisão de Qualidade de Serviço (*Quality of Service* - QoS).

No entanto, é necessário que a camada de aplicação também ofereça algum tipo de diferenciação de serviços. Na maioria dos servidores *Web* atualmente, é usada uma política de atendimento de requisições por ordem de chegada (*First In First Out* - FIFO), não oferecendo nenhum tipo de QoS, e inutilizando os esforços desprendidos pela camada de rede.

Para contornar esta situação, uma arquitetura de serviços diferenciados se mostra adequada. Assim, é necessário aplicar algoritmos que ofereçam diferentes níveis de serviços a diferentes classes de usuários.

Além disso, as políticas de controle de admissão são essenciais para que os servidores *Web* consigam fornecer QoS a seus clientes, prevenindo a sobrecarga do sistema e permitindo que Acordos de Nível de Serviços (*Service Level Agreements* - SLA) firmados com os clientes sejam respeitados. Assim, diante de uma sobrecarga, torna-se necessário o uso de um método que decida quais requisições serão descartadas, e esse é o papel dos algoritmos de controle de admissão.

Este trabalho avalia e analisa o comportamento de dois algoritmos de controle de admissão, o *Drop Tail* e o *Random Early Detection* (RED), utilizando-se um modelo de Servidor *Web* com Diferenciação de Serviços (SWDS) [20], e duas classes de serviços.

Este trabalho está organizado da seguinte forma: na Seção 1, são apresentados alguns algoritmos de controle de admissão e modelos de servidores *Web* com serviços

diferenciados. Na Seção 2 é mostrada a metodologia utilizada na avaliação de desempenho dos algoritmos. Na Seção 3 é descrito o funcionamento dos algoritmos *Drop Tail* e RED. Na Seção 4 são discutidos os resultados obtidos, e finalmente, na Seção 5, as conclusões e trabalhos futuros.

1 Trabalhos relacionados

A diferenciação de serviço na *Web* iniciou-se na camada de rede empregando a arquitetura de Serviços Diferenciados e Serviços Integrados.

No nível de aplicação destacam-se os trabalhos de [20] e [14], sendo estes trabalhos que motivaram o desenvolvimento deste artigo. Em [20] é proposto um modelo de servidor *Web*, composto por um classificador, um módulo de controle de admissão e um *cluster* de servidores *Web*, que possui como objetivo prover serviços diferenciados para diferentes classes de requisições ou usuários. Em [14] é desenvolvido um protótipo do Servidor *Web* com Diferenciação de Serviços (modelo SWDS) em ambiente real, visando fornecer QoS relativa.

Dentre os trabalhos encontrados na literatura que desenvolveram algoritmos de controle de admissão em servidores *Web* com suporte a serviços diferenciados, destacam-se os seguintes.

Em [17] é apresentado um mecanismo de para prover QoS em *clusters* de servidores *Web*, que possui três funções principais: balancear a carga imposta aos servidores, proporcionar diferenciação de serviços e, utilizar os recursos disponíveis de maneira eficaz. Para que a última função seja garantida, os recursos alocados para as diferentes classes de serviços são divididos dinamicamente, em três estados: compartilhados, exclusivos ou saturados. No estado compartilhado, o *cluster* dedicado a uma classe pode aceitar requisições de outras classes, sem comprometer os contratos estabelecidos; no modo exclusivo o cluster recusa requisições de outras classes, e, no estado saturado, nenhuma nova requisição é aceita.

Em [1] é desenvolvido um algoritmo de controle de admissão baseado em lógica *fuzzy* e simulado na arquitetura SWDS.

Em [5] é proposto um algoritmo de controle de admissão e balanceamento de carga para tráfego *Web*. O algoritmo distribui o tráfego em *clusters* de servidores *Web*, a fim de prover qualidade de serviço. O objetivo do algoritmo é, então, evitar situações

em que os *sites Web* proporcionam rendimento abaixo do desejado, devido a um congestionamento nos servidores.

Em [13] é proposto um algoritmo de controle adaptativo, implementado com um servidor de *proxy* entre clientes e servidores *Web*. O *proxy* monitora métricas como, tempo de resposta e requisições aceitas no sistema. Com essas métricas, é empregado um controle baseado em realimentação para determinar a quantidade de requisições aceitas a fim de não permitir a sobrecarga do *cluster*.

Em [21] é destacado o uso do algoritmo RED (*Random Early Detection*) dentro do contexto de servidor *Web*, onde o método é usado para balancear a carga baseando-se nas próprias informações de carga, tendo como resultado um maior aproveito computacional e uma redução na variação da carga de trabalho de cada servidor *Web*.

Com base nos trabalhos citados anteriormente, verifica-se a busca por servidores *Web* com suporte a serviços diferenciados, utilizando técnicas para controle de admissão, a qual complementa a QoS oferecida em nível de rede.

2 Metodologia

Para a avaliação de desempenho dos algoritmos, foi utilizado o modelo SWDS proposto em [20], e mostrado na Figura 1. Este Servidor *Web* com Diferenciação de Serviços foi modelado através de rede de filas e é composto por três módulos: classificador, controle de admissão e *cluster* de servidores *Web*. O classificador é responsável por dividir as requisições que chegam ao sistema em classes de serviços. O controle de admissão decide sobre a aceitação das requisições pelo sistema. E, os servidores do *cluster* processam as requisições admitidas e retornam as respostas para os clientes que as realizaram.

A parametrização utilizada considera um *cluster* homogêneo com seis servidores *Web*, em que cada servidor é modelado individualmente, com sua própria CPU, disco e interface de rede. A capacidade de processamento do classificador é definida como 8000 requisições/s e a capacidade do controle de admissão como 4000 requisições/s. Esses valores foram definidos de acordo com os estudos realizados pelo autor do modelo SWDS [20], tomando como base observações feitas a partir de *benchmarks*, bem como trabalhos relacionados. Para cálculo do tempo de serviço das requisições estáticas, os discos foram parametrizados com taxa de transferência de 300 Mbps e o

tempo de busca de 8,9 ms de acordo com os discos Samsung HD161HJ SATA II [16], e para as requisições dinâmicas, o tempo de serviço é definido como 10 ms [20].

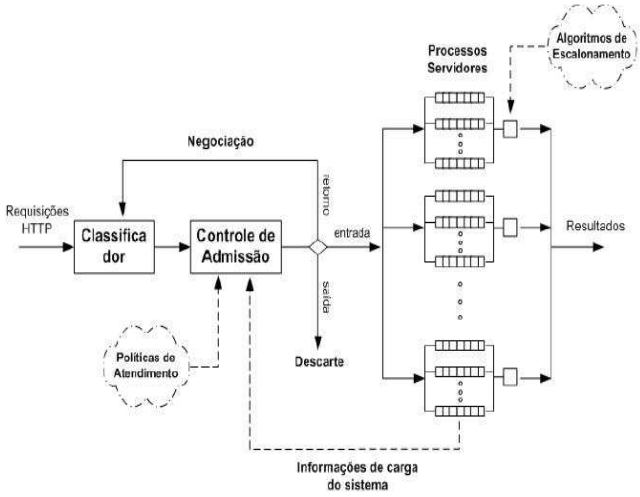

Figura 1. Modelo SWDS [20]

A solução de um modelo pode ser obtida utilizando-se tanto uma ferramenta matemática, levando a uma solução analítica, ou a utilização de um programa de simulação que represente o modelo adequadamente e forneça os dados para a avaliação do desempenho [11] [12] [10].

As soluções analíticas são atrativas, principalmente considerando que, quando disponíveis, permitem a avaliação e a inferência de diversos resultados com esforço computacional pequeno. No entanto, nem sempre se têm soluções analíticas adequadas, uma vez que elas podem ser altamente complexas, levando à necessidade de simplificações sucessivas no modelo do sistema, o que pode levar a resultados sem grandes aplicações práticas [12] [19].

No uso de simulação, tem-se uma possibilidade de solução dos modelos altamente flexível, uma vez que, elaborado um programa de simulação que represente corretamente o modelo, podem ser testadas diversas possibilidades de parametrização do sistema, com resultados computados automaticamente pelo programa de simulação [12].

Para realização da simulação, abordagem utilizada para validar o modelo, foi utilizado o SimpackJ [7], um pacote escrito na linguagem JAVA composto por vários métodos voltados à simulação de Rede de Filas.

Como carga de trabalho, foram geradas 50000 requisições para cada classe de serviço. Esse valor representa a quantidade usuários realizando requisições e, nesses experimentos, representa também o tempo de simulação. Foram utilizadas duas classes de prioridade relativa, sendo a classe A de maior prioridade e a classe B de menor prioridade.

A carga de trabalho foi obtida a partir de uma ferramenta desenvolvida, cujo intervalo de chegada das requisições é obtido similarmente ao HTTPerf [9], utilizando-se uma função de distribuição uniforme. Como não é possível inserir a carga real gerada pelo *benchmark* HTTPerf em um modelo de simulação representando um servidor *Web*, uma carga similar é gerada, utilizando-se as mesmas funções de distribuição do HTTPerf, e gravada em um arquivo *log*. Assim, esse *log* é usado como entrada para o programa de simulação.

O programa utilizado para simular o modelo SWDS requer uma carga de trabalho contendo quatro campos: intervalo de chegada, tipo de objeto requisitado, código de resposta do servidor e tamanho do objeto. O campo intervalo de chegada é obtido baseando-se na carga de trabalho gerada pelo HTTPerf, como descrito anteriormetne. Os demais campos são também gerados sinteticamente, e gravados no arquivo *log* de saída.

O conteúdo de cada requisição é gerado de acordo com [18], em que foram criadas nove classes para agrupar os diferentes tipos de arquivos requisitados.

O código de resposta é gerado de acordo com [15], que apresenta as porcentagens de cada classe de código de resposta obtido dos servidores *Web* baseadas em estudos de *logs*, ou seja, códigos 200 e 206 representando a classe ‘ok’, 301 representando ‘redirecionamento’, 500, 502, 503 e 204 classificados como ‘erro do servidor’ e, 401, 403, 404 e 407 representando a classe ‘proibido’.

O tamanho do objeto, ou seja, o tamanho do arquivo transferido pelo servidor em kbytes, foi gerado de acordo com [4], considerando-se os acessos às páginas iniciais e de nível 1, ou seja, as páginas referenciadas pela página inicial.

Nos experimentos realizados, foram aplicados três tipos de carga, em que a variação da carga ocorre na taxa de requisições geradas por segundo. Essa taxa foi intensificada de 300 para 600 e para 1000 requisições/s, a fim de se verificar o comportamento do sistema, e dos algoritmos de controle de admissão utilizados, diante do aumento na carga. Os resultados obtidos correspondem a dez simulações realizadas para cada taxa, e um intervalo de confiança de 95%.

3 Algoritmos para controle de admissão: Drop Tail e RED

O *Drop Tail* é um algoritmo usado por roteadores de Internet para decidir quando descartar pacotes de rede. Na camada de aplicação esse algoritmo é usado para realizar descartes de requisições em servidores *Web*. Assim, quando a fila de um servidor *Web* está abaixo de um limite pré-estabelecido, novas requisições são aceitas, e quando a fila atinge este limite, as demais requisições são descartadas [6]. Esse algoritmo de controle de admissão foi implementado no modelo SWDS e pode ser considerado o mecanismo mais simples desenvolvido no sistema. A Figura 2 mostra graficamente seu funcionamento.

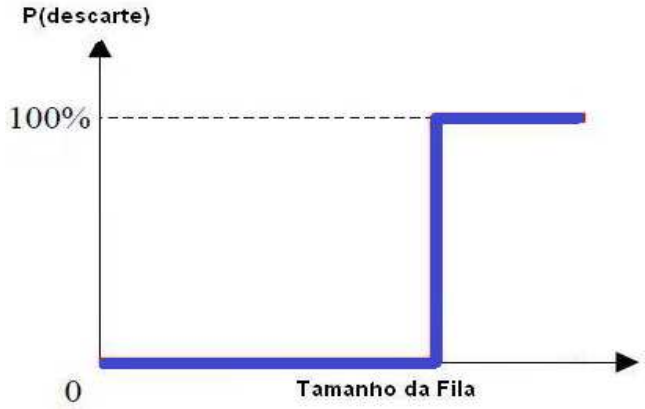

Figura 2. Funcionamento do algoritmo *Drop* Tail.

A fim de realizar uma comparação com o algoritmo anterior, e superar sua rigidez em relação a utilização de um limite para descarte de requisições, será também utilizado, o RED (*Random Early Detection*) [8] para controle de admissão.

Assim como *Drop Tail*, o algoritmo RED foi proposto para descartar pacotes na camada de rede e, posteriormente, foi levado para a camada de aplicação a fim de realizar controle de admissão em servidores *Web*.

No processamento de requisições, o RED calcula o tamanho médio da fila *avg* e o compara com um limite mínimo THMIN e um limite máximo THMAX. Se *avg* for menor que THMIN, a requisição é admitida no sistema. Se *avg* for maior que THMAX, a requisição é descartada. E ainda, se *avg* estiver entre os limites THMIN e THMAX, a probabilidade de descarte *Pa* é calculada e, então, a requisição é descartada com essa probabilidade ou admitida no sistema com a sua complementar.

O tamanho médio da fila *avg* é calculado utilizando-se uma média móvel dos tamanhos das filas anteriores. Usa-se um parâmetro fixo *wq* (peso da fila) que determina

quão rápido *avg* modifica em resposta a uma alteração no tamanho atual da fila. O cálculo do *avg* é dado pela seguinte fórmula:

$$\mathbf{avg = (1 - wq) * avg' + wq * q,}$$

em que, *q* é a média do tamanho atual da fila dos servidores do *cluster*, e *avg'* representa o histórico dos tamanhos das filas.

A Figura 3 mostra esquematicamente o funcionamento do RED.

Quando o *avg* está entre os limites THMIN e THMAX, quanto mais próximo estiver de THMAX, maior a probabilidade de descarte. Mantém-se também, uma variável contadora (*count*) para determinar o número de requisições consecutivas que escaparam do descarte; quanto maior o valor de *count*, maior a probabilidade de descarte. A fórmula da probabilidade de descarte *Pa* é dada por:

$$\mathbf{Pa = Pb / (1 - count * Pb),}$$

com: $\mathbf{Pb = Pmax * [(avg - THMIN) / (THMAX - THMIN)],}$

em que, *Pb* é uma probabilidade temporária usada no cálculo de *Pa* e, *Pmax* é o valor máximo que *Pb* pode atingir.

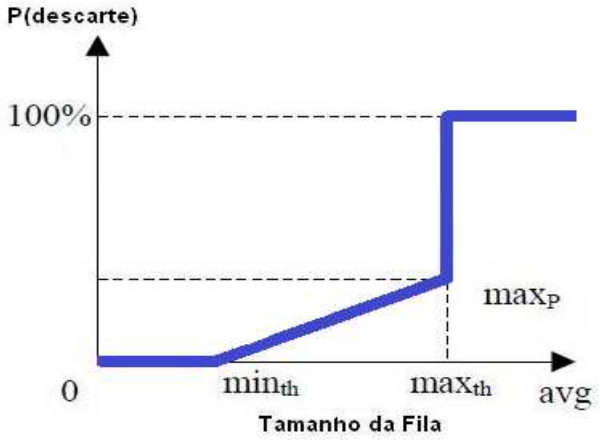

Figura 3. Funcionamento do algoritmo RED.

4 Resultados experimentais

Nos experimentos realizados com o modelo SWDS e os algoritmos *Drop Tail* e RED, foi utilizado um *cluster* homogêneo com seis servidores, e o algoritmo de reserva de recursos, RSV (Reserva de Recursos) [20], fixando quatro servidores para a classe de maior prioridade (classe A) e dois servidores para a classe de menor prioridade (classe B). A probabilidade auxiliar *Pmax* do algoritmo RED foi fixada em 0,2 de acordo com [21] em que foi realizado um estudo para determinar o seu valor ótimo. Já os parâmetros THMIN e THMAX e peso da fila *wq* variaram bem como o limite máximo para o tamanho da fila do algoritmo *Drop Tail*. Os limites para o algoritmo RED foram escolhidos de forma que a sua média fosse igual ao limite máximo do algoritmo *Drop Tail*.

Para facilitar a compreensão e a visualização dos gráficos, a Tabela 1 apresenta um resumo dos experimentos realizados, associando o seu nome com o algoritmo e a configuração utilizada.

Tabela 1. Configuração dos experimentos realizados

Experimento	Algoritmos e Parâmetros
Teste 1	Drop Tail - Limite = 400
Teste 2	RED - THMIN = 200; THMAX = 600; wq = 0,2
Teste 3	RED - THMIN = 200; THMAX = 600; wq = 0,42
Teste 4	Drop Tail - Limite = 600
Teste 5	RED - THMIN = 400; THMAX = 800; wq = 0,2
Teste 6	RED - THMIN = 400; THMAX = 800; wq = 0,42
Teste 7	Drop Tail - Limite = 700
Teste 8	RED - THMIN = 550; THMAX = 850; wq = 0,2
Teste 9	RED - THMIN = 550; THMAX = 850; wq = 0,42

A Figura 4 mostra o gráfico do tempo de resposta para um cenário em que são enviadas 300 requisições/s. Observa-se que nos três primeiros testes, nos quais os limites são baixos, o *Drop Tail* (teste 1) obteve o menor tempo de resposta para a classe B, comparado aos demais testes. Este fato se deve a uma grande quantidade de requisições descartadas, 27,59% ± 0,50%, como mostrado na Figura 3. Já o RED com wq = 0,2 obteve o maior tempo de resposta em resposta ao menor número de descarte. O RED obteve melhor aproveitamento que o *Drop Tail*, para a classe B, de 20,94% para o teste 2, e 8,24% para o teste 3.

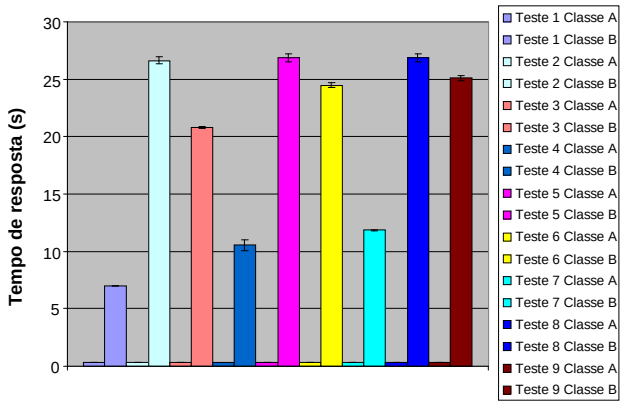

Figura 4. Tempo de resposta para carga com taxa de 300 requisições/s.

Nos testes de 4 a 6, observa-se a mesma relação entre as configurações, que nos testes de 1 a 3. Observa-se no gráfico da Figura 5, que no teste 5 para a classe B, o número de descartes se aproximou de zero (0,09% ± 0,05%) devido à configuração dos limites mais alta, sem, no entanto, haver aumento no tempo de resposta do teste 2 para o teste 5. Verifica-se também um melhor aproveitamento do RED em relação ao *Drop Tail*, para ambos os testes (5 e 6), obtendo um 25,87% para o teste 5, e 10,35% para o teste 6. Isto mostra a rigidez no descarte do algoritmo *Drop Tail*. Nos testes de 7 a 9, as relações entre os algoritmos também foram as mesmas, com o número de descartes da classe B, no teste 8 também próximo a 0% (0,12% ± 0,04%), o que também não influenciou no tempo de resposta causando diferenças entre o teste 5 e o 8.

Já para a classe A, de maior prioridade, o número de descartes foi aproximadamente 0% para todos os testes, apresentando o mesmo tempo de resposta, o

que mostra que a carga não foi suficiente para saturar todos os recursos disponíveis para essa classe.

Com uma carga relativamente mais alta submetida ao sistema, 600 requisições/s, verificam-se alguns comportamentos característicos de cada algoritmo. Como mostrado no gráfico da Figura 6, em todos os grupos de testes (1 a 3, 4 a 6 e 7 a 9), o algoritmo *Drop Tail* obteve menor tempo de resposta em relação ao algoritmo RED, considerando-se as duas configurações. Isto ocorre devido ao maior número de descarte, como mostrado no gráfico da Figura 7.

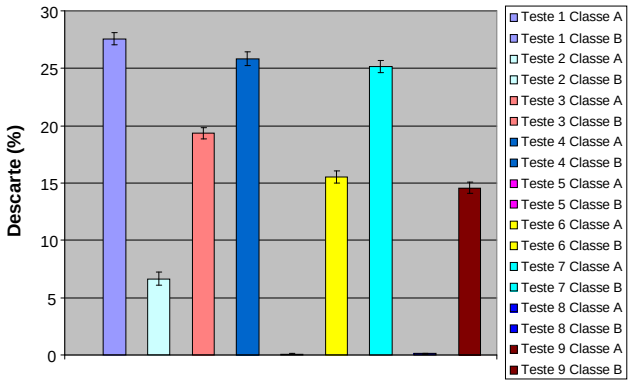

Figura 5. Porcentagem de descarte para carga com taxa de 300 requisições/s.

Verifica-se também que, no algoritmo RED, o peso da fila (*wq*) influencia nos resultados obtidos. Verifica-se ainda, neste cenário que, no último grupo (testes 7 a 9), que para os limites maiores, o número de descarte para a classe B do RED se manteve o mesmo, obtendo-se o mesmo tempo de resposta. A classe A do teste 8 conseguiu obter 0% de descarte, sacrificando o tempo de resposta. O mesmo ocorre no grupo anterior para a classe A.

No último cenário, é realizada uma sobrecarga, enviando 1000 requisições/s ao sistema. O comportamento e relação entre os algoritmos apresentados nos cenários anteriores se acentuaram como mostrado nos gráficos das Figuras 8 e 9.

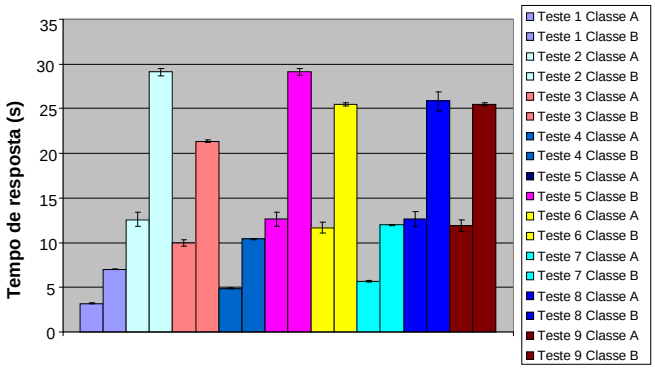

Figura 6. Tempo de resposta para carga com taxa de 600 requisições/s.

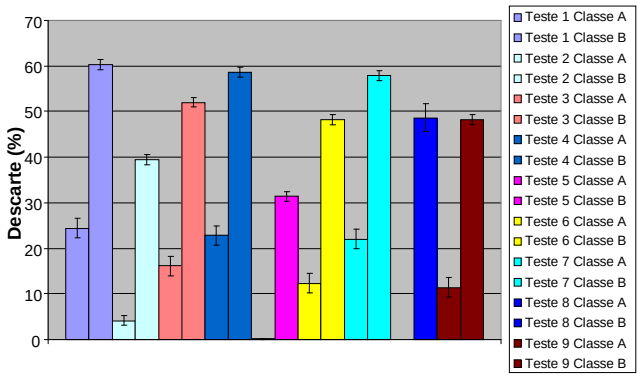

Figura 7. Porcentagem de descarte para carga com taxa de 600 requisições/s.

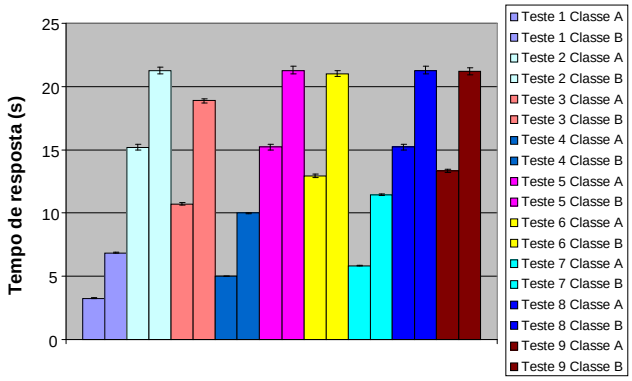

Figura 8. Tempo de resposta para carga com taxa de 1000 requisições/s.

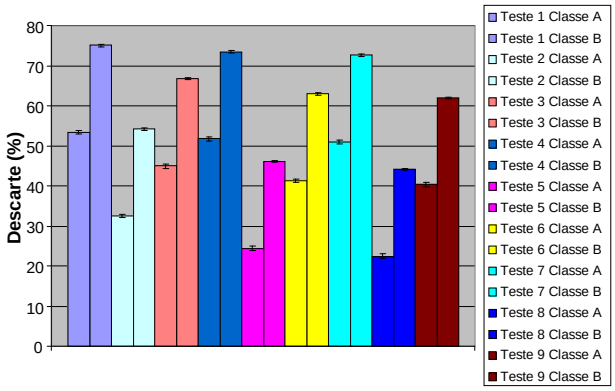

Figura 9. Porcentagem de descarte para carga com taxa de 1000 requisições/s.

5 Conclusões e trabalhos futuros

Neste artigo, foram avaliados os algoritmos de controle de admissão, *Drop Tail* e RED, ambos inseridos em um modelo de Servidor *Web* com Diferenciação de Serviços (SWDS). Nos experimentos, foram utilizadas diferentes configurações para os algoritmos e duas classes de prioridade, a fim de se alcançar a diferenciação de serviços. Para isso, considerou-se uma QoS relativa, em que não se utiliza limites rígidos de tempo de resposta e porcentagem de descartes firmados em SLAs, mas a prioridade de uma classe em relação às demais.

Para priorizar uma classe de serviço em relação à outra, foi utilizado um algoritmo de particionamento de recursos, o RSV, direcionando quatro servidores do *cluster* para a classe de maior prioridade, e duas para a classe inferior.

Verificou-se que a classe de maior prioridade (classe A), obteve melhor qualidade de serviço para todos os valores de carga, tendo valores de tempo de resposta e número de descartes inferiores à classe de menor prioridade (classe B).

Verificou-se ainda que, em todos os cenários, o *Drop Tail* obteve um tempo de resposta menor que o RED em todas as configurações, no entanto, o número de requisições atendidas foi inferior.

Estes resultados mostram o comportamento dos algoritmos utilizados para controle de admissão. Como o RED possui uma região crítica, isso faz com que ele atenda um número maior de requisições, mostrando-se mais cauteloso ao descartar. Como conseqüência do maior número de aceitação, tem-se um tempo de resposta maior em relação ao *Drop Tail*. No entanto, em um cenário com carga de trabalho intensa,

como 1000 requisições/s, a diminuição na porcentagem de descarte do RED é consideravelmente alta, em relação ao aumento no tempo de resposta.

Por exemplo, quando se considera o *Drop Tail* com limite 600 e o RED com limites THMIN = 400 e THMAX = 800, e wq = 0,2 e wq = 0,42, no último cenário. Há uma diminuição de 10% na taxa de descarte do RED (wq = 0,42) em relação ao *Drop Tail*, para a classe B, e um aumento de 12 segundos no tempo de resposta. Comparando o RED (wq = 0,42) com o *Drop Tail*, para a classe B, há uma diminuição de 25% na porcentagem de descarte, e um aumento também de 12 segundos no tempo de resposta.

Verifica-se então, que outra questão a ser observada em relação ao desempenho do RED, é quanto aos valores usados para peso da fila (*wq*).

Na escolha de um dos algoritmos para realização de controle de admissão em nível de aplicação, deve-se ponderar então, entre o tempo de resposta e a tolerância a uma porcentagem maior de descartes.

Considerando como exemplo, um ambiente em que os usuários da classe A (maior prioridade) têm um custo financeiro maior, eles devem obter uma qualidade de serviço melhor. Nesse caso, o algoritmo que melhor se adéqua é o *Drop Tail*. Esse manteria um pequeno grupo de usuários com um tempo de resposta menor, enquanto o RED manteria um maior número de usuários, com penalidades no tempo de resposta.

Os resultados obtidos com este trabalho motivam a investigação dos algoritmos *Drop Tail* e RED em conjunto com outros algoritmos de balanceamento de carga e/ou particionamento de recursos, bem como a investigação de outros algoritmos de controle de admissão. Outra possibilidade seria desenvolver variações do RED, como a inserção do conceito de sessões, já que em vários ambientes como *e-commerce* as requisições de um usuário estão relacionadas entre si e contidas em uma sessão, e aplicá-los no contexto de servidores *Web*.

Agradecimentos

Agradecimentos ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro dado a este trabalho.

Referências

[1] BARROS, V. H.; OLIVEIRA, A. C. M.; TEIXEIRA, M. M.; A Fuzzy Admission

Controller in a QoS-Aware Web Server Architecture, Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on, p.453-458, 10-12, Sept. 2008 doi: 10.1109/HIS.2008.73
[2] BLAKE, S., BLAK, D., CARLSON, M., DAVIES, E., WANG, Z.; WEISS, W. (1998). An Anchitecture for Differentiated Services. In: RFC 2475. IETF.
[3] BRADEN, R., CLARK, D.; SHENKER, S. (1994). Integrated Services in the Internet Architecture. In RFC 1633. IETF.
[4] CHEHADEH, Y. C., HATAHET, A. Z., AGAMY, A. E., BAMAKHRAMA, M. A.; BANAWAN, S. A. (2006). Investigating Distribution of Data of HTTP Traffic: An Empirical Study. In *Proceedings* of the. Innovations in Information Technology, Dubai, United Arab Emirates (UAE), pages 1–5. IEEE Communications Society.
[5] DE LA SIERRA, K. G. (2009). An adaptive admission control and load balancing algorithm for a QoS-aware Web system. Tese de doutorado, Universitat de les Illes Balears, Palma de Mallorca, Spain.
[6] FIROIU, V.; BORDEN, M.; A study of active management for congestion control, INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings. IEEE, v. 3, p.1435-1444, 26-30 Mar 2000, doi: 10.1109/INFCOM.2000.832541.
[7] FISHWICK, P. A. (2005). SimPackJ. University of Florida. Disponível em http://www.cise.ufl.edu/_fishwick/simpackj/. Acesso em abril de 2011.
[8] FLOYD, S. and Jacobson, V. (1993). Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, p. 397–413.
[9] HTTPERF (2009). Disponível em: <http: // www. hpl. hp. com/ research/ linux/ httperf/> . Acesso em: abr. 2011.
[10] JAIN, R. (1991). The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation and Modeling. Wiley - Interscience, New York, NY.
[11] KOBAYASHI, H. (1978). Modeling and Analysis - An Introduction to System Performance Evaluation. Addison - Wesley Publishing Company.
[12] MACDOUGALL, M. H. (1987). Simulating Computer Systems Techniques and Tools. The MIT Press.
[13] MATHUR, V., PATIL, P., APTE, V.; MOUDGALYA, K. (2009) Adaptive admission control for *web* applications with variable capacity. In Quality of Service, IWQoS. 17th International Workshop on, p. 1 –5.
[14] MESSIAS, V. R. (2007). Servidor Web Distribuído com Diferenciação de Serviços - Implementação e Avaliação de um Protótipo. Master's thesis, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos, SP, Brasil.
[15] MODESTO, M., PEREIRA JR., A. R., ZIVIANI, N., CASTILHO, C.; BAEZA-YATES, R. (2005). Um novo retrato da Web brasileira. In *Proceedings* of *SEMISH'05:* Seminário Integrado de Software e Hardware, São Leopoldo, RS, Brasil. XXV Congresso da Sociedade Brasileira da Computação.
[16] SAMSUNG (2010). Disponível em: <http://www.samsung.com/ br/> . Acesso em: abr. 2011.
[17] SERRA, A., GAÏTI, D., CARDOSO, K., BARROSO, G.; RAMOS, R. (2005). Controle de Admissão e Diferenciação de Serviços em Clusters de Servidores Web. XXIII Simpósio Brasileiro de Redes de Computadores (SBRC'05). Fortaleza, CE, Brasil.
[18] SILVA, L. H. C. (2006). Caracterização de Carga de Trabalho para Testes de Modelos de Servidores Web. Master's thesis, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos, SP, Brasil.

[19] SOARES, L. F. G. (1990). Modelagem e Simulação Discreta de Sistemas. VII Escola de Computação, São Paulo.
[20] TEIXEIRA, M. A. M. (2004). Suporte a Serviços Diferenciados em Servidores Web: Modelos e Algoritmos. PhD thesis, Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo, São Carlos, SP, Brasil.
[21] ZHENG, B.; ATIQUZZAMAN, M. (2008). A framework to determine the optimal weight parameter of RED in next-generation Internet routers. International Journal of Communication Systems, 21(09): 987-1008. DOI: 10.1002/dac.932.