Escalonador de Métrica Única Combinada para a Implementação dos Serviços Proporcionais Diferenciados

Flavio B. Gonzaga, Ronaldo M. Salles

[1]Seção de Engenharia de Sistemas – Instituto Militar de Engenharia (IME)
Praça Gen Tibúrcio 80, 22290-270, Rio de Janeiro, R.J., Brazil

fbgonzaga@de9.ime.eb.br, salles@ieee.org

Abstract. *This work proposes the use of a single composite metric (MUC) in the implementation of Proportional Differentiated Services (PDS). MUC is obtained through the relationship between throughput and delay providing a more consistent PDS implementation than the ones based solely on delay. A new scheduler algorithm is also proposed to work with MUC. Simulation results show the advantages of the proposed approach when compared to other traditional schedulers considered for the PDS model, such as the Waiting Time Priority (WTP) and the Proportional Average Delay (PAD).*

Resumo. *Este trabalho propõe o uso de uma métrica única combinada (MUC) para a implementação dos Serviços Proporcionais Diferenciados (Proportional Differentiated Services - PDS). MUC é baseada na relação entre vazão e atraso observadas nas classses de serviço, proporcionando uma implementação PDS mais consistente do que aquelas baseadas apenas no atraso. Um novo algoritmo de escalonamento é proposto para trabalhar diretamente com MUC. Resultados de simulação mostram as vantagens da proposta quando comparada com escalonadores comumente empregados no modelo PDS, tais como Waiting Time Priority (WTP) e Proportional Average Delay (PAD).*

1. Introdução

No início da Internet a transferência de informações era basicamente do tipo texto, sendo compartilhadas pelos centros universitários da época. Contudo, com a popularização dos computadores, a Internet passou a estar acessível nos mais diversos lugares do mundo. Ao contrário do seu início, hoje a grande rede mundial conta com inúmeros tipos de aplicações, cada uma com características e necessidades específicas. Por exemplo, a Internet é atualmente também utilizada para trafegar dados provenientes de aplicações multimídia, VoIP, vídeo broadcast, P2P, vídeo-conferências, transações financeiras em tempo real, jogos e ambientes virtuais colaborativos, computação de alto desempenho, etc. O problema é que não é possível oferecer nenhum tipo de garantia com base na arquitetura legada, que é do tipo melhor esforço (*best-effort*) [P. Gevros and Bhatti 2001]. Ou seja, a Internet não realiza distinção entre os dados que trafegam nela, de forma que aplicações que exijam algum tipo especial de tratamento (para evitar problemas como atraso, perda de pacotes, etc..) terão seus pacotes tratados da mesma maneira que as demais aplicações que possuem um maior nível de tolerância a esses problemas.

Visando acrescentar mais essa funcionalidade à Internet, estudos tem sido realizados buscando uma maneira de oferecer a Qualidade de Serviço (*QoS*) adequada aos

diversos tipos de dados mesmo em condições de alta carga. Na literatura existem diversas propostas, nas quais as duas principais e mais bem aceitas são a arquitetura dos Serviços Integrados *(IntServ)* [R. Braden and Shenker 1994] [White 1997] e a dos Serviços Diferenciados *(DiffServ)* [Blake 1998].

A arquitetura IntServ é baseada em um modelo orientado a conexões. A principal restrição ao emprego do IntServ é quanto a sua baixa escalabilidade. O número excessivo de mensagens para se manter cada conexão gera um grande *overhead*, tornando o IntServ inviável em uma rede onde existam um número excessivo de nós com diversas conexões.

O modelo de Serviços Diferenciados foi desenvolvido buscando resolver os problemas de escalabilidade apresentados pelo IntServ. Essa arquitetura se caracteriza por não ser orientada a fluxos de dados (como o IntServ), mas sim a agregados de fluxos que possuam características semelhantes. Cada agregado de fluxo constitui uma classe de serviço. Para implementar o DiffServ é criado um domínio, que é constituído de nós de borda (que realizam a medição dos pacotes que chegam ao domínio, medidas de policiamento quando necessárias, marcação e encaminhamento dos mesmos para as classes correspondentes) e nós internos, que seguem transmitindo os pacotes nas respectivas classes no interior do domínio. Um usuário que deseje utilizar determinado nível de QoS oferecido por uma classe deve contratar o serviço.

Apesar de resolver os problemas do IntServ, a QoS oferecida pela arquitetura DiffServ ficava muito suscetível as condições da rede. Em determinadas situações, classes de serviço superior poderiam experimentar uma QoS pior que as demais classes descongestionadas naquele momento. Além disso, a diferenciação de QoS oferecida em cada classe não é bem definida.

Para resolver esse problema, Dovrolis em [Dovrolis 1999] [C. Dovrolis and Ramanathan 1999] propõe a arquitetura dos Serviços Diferenciados Proporcionais (*Proportional Differentiated Services - PDS*), baseada em duas importantes características:

- Previsibilidade: diferenciação deve ser consistente (por exemplo, classes superiores devem ser melhores ou pelo menos não serem piores do que as classes inferiores) independente das variações de carga;
- Controlabilidade: a rede deve permitir que o administrador configure a diferença de QoS entre as classes de acordo com o seu critério.

Dovrolis propõe ainda como métrica a ser utilizada para se medir a QoS o atraso, constituindo assim o denominado *Proportional Delay Differentiation Model (PDD)*.

O objetivo deste trabalho é dentro do modelo PDS, propor a utilização de uma nova métrica que considere não apenas o atraso como também a vazão experimentada pelas classes. Além disso, o trabalho propõe um novo escalonador para ser utilizado com a métrica única combinada (MUC) gerando implementações PDS mais consistentes.

O presente trabalho está organizado da seguinte maneira. Na seção II é apresentada uma revisão do modelo dos Serviços Proporcionais Diferenciados. Na seção III são mostrados os escalonadores *Priority Queue (PQ)*, *Waiting Time Priority (WTP)* e *Proportional Average Delay (PAD)*. As seções IV e V apresentam o escalonador proposto baseado na métrica MUC e os resultados obtidos na simulação. O trabalho termina com conclusão (seção VI) e as referências.

2. Modelo de Diferenciação Proporcional - PDS

O objetivo principal do PDS [Dovrolis 1999] é manter as seguintes relações válidas,

$$\frac{q_1}{q_2} = \frac{c_1}{c_2}, \quad \frac{q_2}{q_3} = \frac{c_2}{c_3}, \quad \frac{q_3}{q_4} = \frac{c_3}{c_4} \ldots \tag{1}$$

onde:

- q_i é a qualidade de serviço obtida na classe i;
- c_i é o parâmetro configurado pelo administrador da rede para a classe i;
- $i = 1, 2, 3, \ldots$ representa o número de classes existentes no sistema.

A partir da equação (1) pode-se dizer que o serviço é *previsível* uma vez que a proporção da qualidade de serviço é fixa para cada uma das classes, independente das variações de tráfego, e *controlável* uma vez que o administrador pode configurar os parâmetros c_i.

No caso específico do modelo de Diferenciação Proporcional do Atraso *(Proportional Delay Differentiation – PDD)* a parâmetro q_i escolhido para se medir a QoS de uma classe é o atraso do pacote a_i (tempo de espera). Para se implementar o modelo PDD alguns escalonadores tem sido propostos na literatura, como o *Proportional Average Delay (PAD)* e o *Waiting Time Priority (WTP)* [C. Dovrolis and Ramanathan 1999].

3. Escalonadores que Implementam o Modelo PDD

Nesta seção apresentamos três escalonadores já bem conhecidos na literatura (PQ, WTP e PAD), que foram utilizados nos cenários de simulação para a avaliação de desempenho.

3.1. Priority Queue (PQ)

Apesar de não ser um escalonador utilizado geralmente no modelo de Diferenciação Proporcional do Atraso (PDD), o escalonador PQ é apresentado neste trabalho por ser implementado em muitos roteadores, servindo assim para efeito ilustrativo.

No PQ cada fila possui uma determinada prioridade. Considerando um sistema com duas filas, enquanto a fila de maior prioridade tiver pacotes no buffer, a fila de menor prioridade não será atendida. Esse mecanismo de prioridade causa um estrangulamento na fila de prioridade mais baixa em condições de alta carga. Por esse motivo, esse modelo não é indicado para o modelo PDD.

3.2. Waiting Time Priority (WTP)

O escalonador WTP [Kleinrock 1976] é um dos melhores escalonadores conhecidos para implementar o PDD. Nesse escalonador cada classe possui um buffer para armazenar os seus pacotes e o WTP monitora o tempo de espera dos pacotes que se encontram na cabeça da fila. À medida que esse tempo aumenta, a prioridade do pacote aumenta linearmente de modo que,

$$\tilde{a}_i^t = \frac{a_i^t}{c_i} \tag{2}$$

- $\tilde{a}_i^t$ tempo de espera normalizado da cabeça da classe i no tempo t;
- a_i^t tempo de espera da cabeça da classe i no tempo t;
- c_i valor configurado pelo administrador da rede para a classe i.

Sempre que um pacote precisar ser atendido, o escalonador verifica a relação entre o tempo de espera do pacote localizado na cabeça de cada classe e o valor configurado pelo administrador da rede para a respectiva fila. O escalonador atenderá a classe cujo pacote localizado no topo da fila apresentar o maior tempo de espera normalizado. Desse modo o WTP busca manter a relação (1) minimizando os possíveis desvios.

3.3. Proportional Average Delay (PAD)

O escalonador PAD constitui uma outra alternativa para o modelo PDD. Esse escalonador trabalha verificando a média dos atrasos dos pacotes que já foram atendidos em cada classe. Considerando que N_i^t representa a sequência dos pacotes atendidos na classe i no tempo t, e a_i^m o atraso do pacote m em N_i. A média do atraso normalizado da classe i no tempo t é dada por:

$$\tilde{d_i^t} = \frac{1}{c_i} \frac{\sum_{m=1}^{|N_i^t|} a_i^m}{|N_i^t|} \tag{3}$$

Sempre que o escalonador PAD fica livre, ele verifica as classes e atende a classe que possuir a maior média de atraso normalizado. O intuito é então reduzir esse valor e, por conseguinte, minimizar os desvios no objetivo do PDS (1).

4. Escalonador Baseado na Métrica Única Combinada (MUC)

O escalonador MUC é proposto neste artigo. O presente escalonador é baseado na medida da Vazão/Atraso, que é também conhecida na literatura como *Power* [Jaffe 1981] [A. Giesler and Pade 1978]. A idéia da utilização da Métrica Única Combinada se baseia no fato de que em alguns cenários a utilização apenas do Atraso constitui uma medida incompleta. Por exemplo, em situações de sobrecarga de alguma das classes, um escalonador que considere apenas o atraso tenderá a atender demasiadamente essa classe, não percebendo que a sua vazão já está muito alta também.

Na implementação do escalonador, uma lista circular (janela) baseada em [Salles and Barria 2002] é usada para armazenar o tempo de chegada de cada pacote, o tempo de serviço ocupado pelo pacote ao ser atendido e seu tamanho. Um exemplo do funcionamento da janela é apresentado na Fig. 1.

- Fig.1(a): Chegada do primeiro pacote na classe i. O tempo de chegada do pacote é armazenado. O ponteiro Cabeça_da_Janela aponta para a posição onde o tempo de chegada do próximo pacote será gravado. O ponteiro Cabeça_da_Fila indica o elemento da classe a ser atendido pelo escalonador.
- Fig.1(b): Chegada do segundo pacote na classe i. O tempo de chegada do pacote é armazenado.
- Fig.1(c): Classe i é atendida, e o pacote que estava na posição apontada por Cabeça_da_Fila é atendido. O tempo de serviço ocupado pelo pacote é então armazenado, juntamente com o seu tamanho. O ponteiro Cabeça_da_Fila passa a apontar para a posição que contém o próximo pacote a ser atendido na classe.
- Fig.1(d): Classe i é atendida, e o segundo pacote é servido. O tempo de serviço ocupado pelo pacote é então armazenado, juntamente com o seu tamanho. O ponteiro Cabeça_da_Fila passa a apontar para a mesma posição que o ponteiro Cabeça_da_Janela. Quando ambos os ponteiros apontam para a mesma posição, indica que o buffer da classe i está vazio.

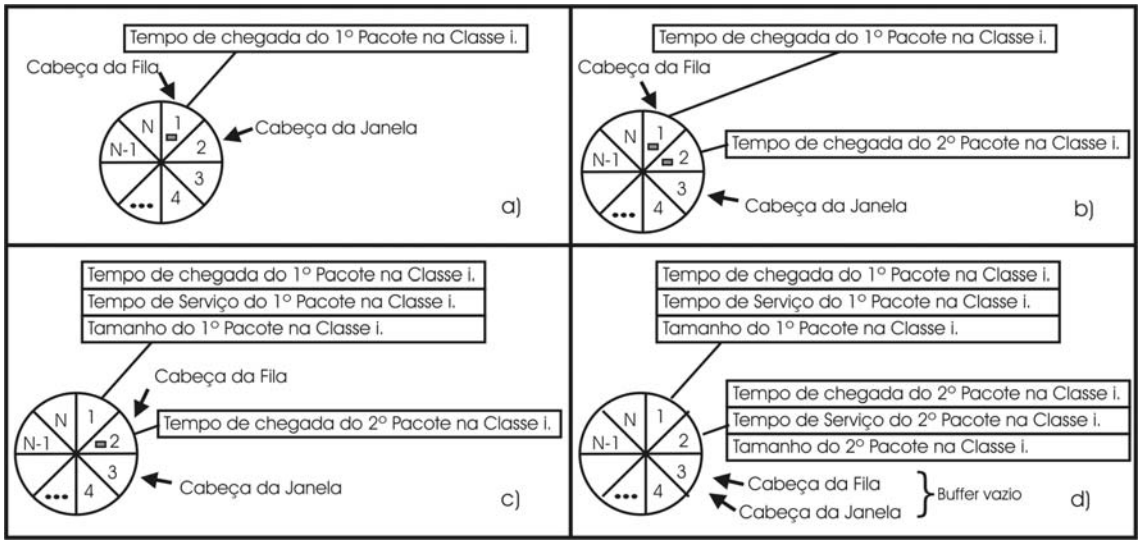

Figure 1. Lista Circular.

A partir da análise feita na Fig.1(d) podemos concluir ainda a situação onde a janela indicará que o buffer está cheio. Isso acontecerá quando o ponteiro Cabeça_da_Janela apontar para a posição anterior à apontada pelo ponteiro Cabeça_da_Fila.

4.1. Calculando a Vazão

Sendo *T* a soma do tamanho dos pacotes atendidos na janela, *C* o tempo de chegada do pacote mais antigo armazenado na janela e *t* o tempo atual. A vazão é então calculada da seguinte maneira:

$$V_i^t = \frac{T_i^t}{t - C_i} \tag{4}$$

Onde:

Table 1. Operações Realizadas

A cada pacote atendido	A cada valor antigo substituído na Janela
T ⇐ T + tamanho_do_pacote_atendido;	T ⇐ T - tamanho_do_pacote_no_campo_n; C ⇐ tempo_de_chegada_em_n+1;

4.2. Calculando o Atraso

Sendo *N* o número total de elementos na Janela (pacotes atendidos e pacotes que ainda estão no buffer), *B* o número total de elementos no buffer, *S* a soma do tempo de chegada dos pacotes que ainda estão no buffer e *A* a soma do atraso dos pacotes que já foram atendidos na Janela. O atraso é então calculado da seguinte maneira:

$$A_i^t = \frac{1}{N_i^t}\left(B_i^t t - S_i^t + A_i^t\right) \tag{5}$$

Onde as variáveis são atualizadas de acordo com a Tabela 2.

Table 2. Operações Realizadas

A cada pacote atendido	A cada chegada de pacote
B ⇐ B - 1; S ⇐ S - tempo_de_chegada_do_pacote_atendido; A ⇐ A + Atraso_do_pacote_atendido;	N ⇐ N + 1;* B ⇐ B + 1;** S ⇐ S + t;**

(*) até completar o tamanho da Janela
(**) até completar o tamanho do Buffer

4.3. Calculando o Valor da MUC

Uma vez obtidos os valores da vazão e do atraso, o cálculo da Métrica Única Combinada é feito conforme a equação (6):

$$M_i^t = \frac{V_i^t}{A_i^t} \tag{6}$$

Calculado o valor de MUC, o escalonador verifica a relação das proporções obtidas entre as classes e as proporções configuradas pelo Administrador da Rede (Conforme mostrado em 1), atendendo a classe que possuir o menor valor normalizado de MUC.

4.4. Complexidade do Algoritmo de Escalonamento

Outra questão a ser analisada é a complexidade do algoritmo do escalonador proposto. As análises mostraram que embora o mesmo trabalhe com 2 métricas, a sua complexidade é bem baixa. A tabela abaixo ilustra os passos seguidos pelo algoritmo, com as respectivas complexidades:

Table 3. Complexidade do Algoritmo

	Passos do Algoritmo	Complexidade
1.	Verifica Buffer para n classes	O(n)
2.	Calcula MUC para n classes	O(n)
3.	Verifica relação de MUC para (n-1) classes e define qual a classe atender	O(n)
	Complexidade Final	O($3n$)

Vale ressaltar ainda que n é o número de classes e que as análises acima feitas refletem a pior situação (quando todas as classes possuem pacotes a serem atendidos). Isso porque em determinado momento pode ser que de 3 classes por exemplo, apenas 2 possuam pacotes no buffer, reduzindo ainda mais a complexidade dos passos **2** e **3**, que poderão verificar apenas entre as "classes ativas" qual deverá ser atendida.

Considerando que o número de classes no modelo *PDS* não é elevado, constatamos que o algoritmo aqui proposto possui complexidade bem reduzida.

5. Avaliação de Desempenho

Nesta seção é realizada uma análise do desempenho apresentado pelo escalonador proposto MUC em comparação aos resultados obtidos pelos demais escalonadores (PQ, WTP e PAD). O parâmetro utilizado para se medir a performance para cada escalonador é o erro obtido pela diferença entre a proporção configurada pelo administrador da rede e a

proporção obtida para as respectivas classes, de acordo com o princípio do modelo PDS já apresentado na equação (1).

O erro resultante entre duas classes i e $i+1$ no tempo t pode ser obtido da seguinte forma:

$$E_{i,i+1} = \frac{MUC_i^t}{MUC_{i+1}^t} - \frac{c_i}{c_{i+1}} \tag{7}$$

onde c_i e c_{i+1} são os valores configurados pelo administrador.

Quando obtemos $E_{i,i+1} = 0$, dizemos que a proporção entre essas duas classes estão de acordo com a proporção configurada. Logo, o modelo PDS foi perfeitamente atendido. Qualquer desvio nos valores medidos da MUC irão produzir valores de $E_{i,i+1} > 0$ ou $E_{i,i+1} < 0$. Desse modo, o erro final resultante do sistema é melhor obtido através do somatório dos erros ocorridos entre as classes. A equação (8) ilustra esse cálculo.

$$E = \frac{\sum_{i=1}^{classes-1} \sqrt{\left(\frac{MUC_i^t}{MUC_{i+1}^t} - \frac{c_i}{c_{i+1}}\right)^2}}{classes - 1} \tag{8}$$

A equação (8) pode ser melhor entendida da seguinte forma. Consideremos um sistema com três classe no tempo *t=10s*, $E_{1,2} = +3$ e $E_{2,3} = -3$. Se simplesmente somarmos os erros obtidos entre as classes o erro resultante será igual a zero. Contudo, está claro que nesse caso o modelo PDS não está sendo aclançado. Porém, a partir de (8) temos $E = \frac{\sqrt{(+3)^2}+\sqrt{(-3)^2}}{2} = 3$, que é o desvio absoluto do modelo PDS.

5.1. Geradores de Tráfego

Foram utilizados nas simulações cinco geradores de tráfego diferentes, considerando carga alta e desbalanceada, além de diferentes tamanhos de janela. A seguir a explicação detalhada de cada um deles.

5.1.1. Poisson

O primeiro gerador apresentado é esse em função de já ser bem conhecido e servir de base em inúmeros trabalhos na literatura [Niyato and Hossain 2005] [C. Dovrolis and Ramanathan 1999]. O intervalo entre pacotes é obtido segundo a equação $\frac{-1}{\lambda}\log(a)$, sendo λ o valor da média desejada e *a* uma distribuição uniforme entre 0 e 1.

Para o tráfego Poisson gerado foi utilizada uma taxa de $\lambda = 0,6$ para a classe 0, 0,4 para a classe 1 e 0,5 para a classe 2. Tempo de serviço do escalonador seguindo distribuição exponencial com taxa μ = 1. Com essas configurações é possível analisarmos se existe teoricamente valores de μ capazes de atender no estado estacionário ao objetivo do modelo PDS (*E* = 0).

Considerando cada classe como uma fila independente *M/M/1/K*, obtemos o valor da MUC conforme a expressão abaixo:

$$MUC|_{M/M/1/K} = \frac{\lambda^2(1-\pi k)}{E[N]} \tag{9}$$

Onde:

$$\pi k = \pi 0 \, (\rho)^k \tag{10}$$

$$\pi 0 = \frac{1-\rho}{1-(\rho)^{K+1}} \tag{11}$$

$$E\,[N] = \frac{\rho}{1-\rho} - \frac{(K+1)\,\rho^{K+1}}{1-\rho^{K+1}} \tag{12}$$

$$\rho = \frac{\lambda}{\mu} \tag{13}$$

Para três classes de tráfego então nós temos (9) sujeito a:

$$\sum_{i=1}^{3} \mu_i \leq \mu \tag{14}$$

Para os parâmetros utilizados na nossa simulação verificamos que os seguintes valores de $\mu 1$ = 0,284696, $\mu 2$ = 0,212519, e $\mu 3$ = 0,086615 satisfazem (9).

Se existisse um escalonador ideal que atendesse às classes segundo as taxas obtidas, tal escalonador permitiria a alcançar o objetivo do modelo MUC/PDS no regime estacionário. Contudo, usando a Janela Circular de Pacotes nós queremos mostrar que o escalonador MUC obtém um *Erro* bem próximo do zero também em pequenos intervalos de tempo.

5.1.2. Processo de Poisson Interrompido – IPP

Os demais geradores são baseados no modelo IPP, que consiste no Processo de Poisson Interrompido [Adas 1997]. Neste sistema, mostrado na Fig. 2, o processo é composto por dois estados. Os estados indicam se haverá ou não chegada de pacotes. Sempre que estiver no estado ativo, teremos chegada de pacote seguindo distribuição de Poisson, com uma taxa λ.

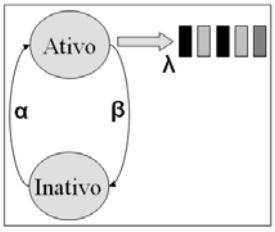

Figure 2. Processo de Poisson Interrompido

O modelo IPP foi escolhido como base para na implementação dos demais geradores em função do mesmo oferecer grandes variações na taxa de chegada de pacotes em cada uma das classes, simulando assim o tráfego em rajada. Foram implementados Processos independentes para cada uma das diferentes classes, de maneira que a alternância entre os estados *Ativo* e *Inativo* ocorra de maneira assíncrona entre elas. Vale ressaltar ainda que variações na taxa de chegada geram oscilações tanto no atraso das classes

quanto na vazão das mesmas, de modo que esse modelo coloca ainda mais em teste o escalonador MUC, que trabalha exatamente controlando a combinação dessas 2 métricas.

A seguir as 4 variações do modelo IPP utilizadas:

- **IPP1** Nesse gerador, temos o controle entre os estados *Ativo* e *Inativo* seguindo taxa exponencial α e β (mostradas na Fig. 2) igual a 0,1, tempo de serviço do escalonador igualmente exponencial com taxa $\mu = 1$, taxa de $\lambda = 1,0$ para a classe 0, 0,6 para a classe 1 e 0,8 para a classe 2, resultando em uma taxa média de chegada igual a 0,49, 0,29 e 0,39 respectivamente para cada uma das 3 classes.
- **IPP2** Esse gerador foi desenvolvido buscando acrescentar no estado *Ativo* um tráfego mais próximo do real de uma rede [M. K. H. Leung and Yau 2001] [Lai and Chang 2004]. No estado *Ativo*, pacotes são gerados seguindo os seguintes tamanhos e proporções pré-fixadas: 40% dos pacotes são de 40 Bytes, 50% dos pacotes são de 550 Bytes e 10% dos pacotes são de 1500 Bytes. O tempo de serviço do escalonador nesse modelo passa a ser determinístico.
  O controle entre os estados *Ativo* e *Inativo* segue taxa exponencial igual a 0,05 para Desativar (β na Fig. 2) e 0,95 para Ativar (α na figura 2).
  Foram simuladas nesse cenário ainda a geração de múltiplas conexões. Novas conexões são geradas seguindo uma taxa exponencial 5. O tempo de duração de cada conexão segue taxa exponencial de 0,05. A taxa de chegada é de $\lambda = 1,2$, 0,8 e 1,0 respectivamente para cada uma das classes, resultando em uma ocupação média de 459 Kbps considerando as 3 classes. O tamanho do link utilizado foi de 512 Kbps.
- **IPP3** Esse gerador foi utilizado buscando acrescentar uma outra distribuição ao já criado modelo **M2**. No modelo **M4**, a exemplo do **M2**, o tempo de serviço do escalonador segue uma distribuição exponencial com taxa $\mu = 1$. Contudo, o controle entre os estados *Ativo* e *Inativo* ocorre segundo a distribuição de Pareto, de maneira semelhante a [Selvaraj 2002].
  A função densidade de probabilidade Bounded Pareto $B(k, p, \alpha)$ é definida por [M. H. Balter and Murta 1999]:

$$f(x) = \frac{\alpha k^{\alpha}}{1 - \left(\frac{k}{p}\right)^{\alpha}} x^{-\alpha-1} \quad k \leq x \leq p \tag{15}$$

  onde:
  - expoente $\alpha = 1,9$ ($0 < \alpha < 2$) indica a dimensão fractal do processo sob investigação;
  - no caso de $1 \leq \alpha \leq 2$, o processo apresenta um segundo momento infinito;
  - *k=1* limite inferior;
  - *p=5* limite superior; [Adas 1997]

  A taxa de chegada é de $\lambda = 1,0$, 0,6 e 0,8 respectivamente, resultando em uma taxa média igual a 0,5, 0,3 e 0,4.
- **IPP4** Esse gerador foi desenvolvido tendo como base o modelo de tamanho pré-fixado de pacotes (conforme já mostrado no modelo **M3**). A diferença desse modelo se deve ao fato de que os estados *Ativo* e *Inativo* são regulados segundo a distribuição Bounded Pareto, explicada em **M4**. No estado ativo também ocorre a

geração de múltiplas conexões, como em **M3**. A taxa de chegada é de $\lambda = 1,3$, 1,0 e 1,1 respectivamente, resultando em uma ocupação média de 246 Kbps considerando as 3 classes. O tamanho do link utilizado foi de 256 Kbps.

Uma vez constituídos os 5 geradores (Poisson e IPPs de 1 a 4), 10 cenários foram gerados. Os diferentes cenários foram obtidos alterando-se o tamanho da janela de dados utilizada. Com a alteração do tamanho da janela busca-se verificar o desempenho do escalonador MUC para diferentes escalas de tempo.

A tabela a seguir mostra a relação entre os 5 geradores, os tamanhos de janela simulados e os 10 cenários obtidos.

Table 4. Cenários

	Janela de 60 posições	Janela de 100 posições
Poisson	**M1**	**M6**
IPP1	**M2**	**M7**
IPP2	**M3**	**M8**
IPP3	**M4**	**M9**
IPP4	**M5**	**M10**

A seguir a Fig. 3 ilustra os cenários gerados com base no Modelo IPP:

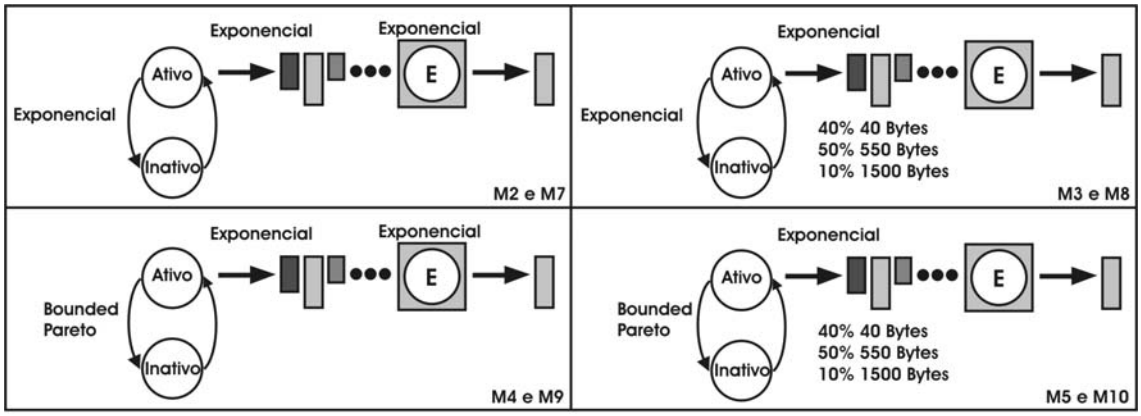

Figure 3. Geradores de Tráfego IPP

5.2. Resultados da Simulação

Durante as simulações a coleta de dados foi realizada a cada 500 segundos até chegar aos 10.000 segundos (tempo total de simulação). Para se configurar os valores de proporção do atraso nos escalonadores WTP e PAD procedeu-se da seguinte forma: primeiro foram realizadas as simulações do escalonador MUC, medindo-se o valor de atraso médio obtido para cada uma das classes. Computados os valores de atraso, os escalonadores WTP e PAD foram configurados com essas proporções, visando utilizar dessa maneira parâmetros justos de configuração em todos os escalonadores. Em todos os cenários foram utilizados os seguintes valores de proporção para o escalonador MUC:

- Classe 1/Classe 2 = Classe 2/Classe 3 = 2

Os escalonadores PAD e PQ não foram incluídos na análise gráfica em função de terem apresentado valores de erro muito elevados em comparação com os escalonadores MUC e WTP. Contudo, após os gráficos, são apresentados nas tabelas 5 e 6 os valores de erro médio computados no decorrer das simulações para todos os escalonadores. Diferente dos resultados apresentados nos gráficos, na composição das médias apresentadas nas tabelas as medições foram realizadas a cada 100 segundos.

Analisando os cenários com janela de tamanho igual a 60 posições, podemos observar que em Fig. 4 e Fig. 7 o escalonador proposto supera os demais com grande diferença de *Erro* obtido. Em Fig. 5 o escalonador MUC se mantém com valores de *Erro* inferiores ao WTP na maior parte da simulação. Os cenários **M3** e **M5**, exibidos em Fig. 6 e Fig. 8 já mostraram que ambos os escalonadores apresentaram muitos picos. Embora tenha sido superado em mais momentos, o escalonador MUC ainda se mantém com valores inferiores ao WTP na maior parte das simulações.

A seguir é apresentada na tabela 5 a média dos *Erros*, computada no decorrer das simulações com janela de tamanho igual a 60 posições. Os resultados mostram que o escalonador PQ realmente não é um escalonador que possa ser utilizado para se prover a Diferenciação Proporcional da Qualidade de Serviços. O PAD também apresentou valores elevados de erro em cenários como os simulados aqui, onde temos carga alta. Isso se deve em função do mesmo considerar apenas o atraso médio de pacotes já atendidos. Com isso, mesmo atendendo a classe superior, o atraso médio da mesma tende a aumentar ainda, causando um estrangulamento nas demais classes do sistema.

Table 5. Gerador de Tráfego e Média do Erro para Janela de 60 Posições

Algoritmo	MUC	PAD	PQ	WTP	WTP/MUC
M1 (Poisson)	0,226	1154,145	2566,949	0,916	4,048
M2 (IPP)	0,310	20,017	30,743	0,882	2,843
M3 (IPP)	1,111	1281,633	353,868	6,875	6,186
M4 (IPP)	0,348	72,048	64,232	1,114	3,194
M5 (IPP)	1,422	8,745	38,875	1,726	1,213

Nos cenários onde foi utilizada janela de tamanho igual a 100 posições, já pode-se perceber uma melhora em ambos os escalonadores, sobretudo do MUC, que não foi superado em nenhum ponto em Fig. 9, Fig. 10 e Fig. 12. Os cenários mostrados em Fig. 11 e em Fig. 13, a exemplo do que já havia acontecido nas simulações com janela de tamanho igual a 60 posições, ambos os escalonadores apresentaram muitos picos, com o MUC ainda se mantendo com valores inferiores de *Erro* em relação ao WTP na maior parte das simulações.

A seguir é apresentada na Tabela 6 a média dos *Erros*, computada no decorrer das simulações com janela de tamanho igual a 100 posições. Como podemos observar, em todos os cenários simulados, apenas no **M8** o WTP superou o MUC na média. Embora no gráfico desse cenário não se possa observar essa diferença, onde o MUC se mantém abaixo do WTP na maior parte, a média de *Erro* do MUC foi maior em função do mesmo apresentar valores mais elevados de picos em função do WTP, aumentando dessa forma consideravelmente o valor da sua média de *Erro*.

Table 6. Gerador de Tráfego e Média do Erro para Janela de 100 Posições

Algoritmo	MUC	PAD	PQ	WTP	WTP/MUC
M6 (Poisson)	0,086	328,842	1495,161	0,344	3,984
M7 (IPP)	0,127	10,431	13,688	0,643	5,054
M8 (IPP)	5,031	77,930	108,199	3,256	0,647
M9 (IPP)	0,063	8,969	34,481	0,362	5,744
M10 (IPP)	0,642	2,393	10,859	1,387	2,157

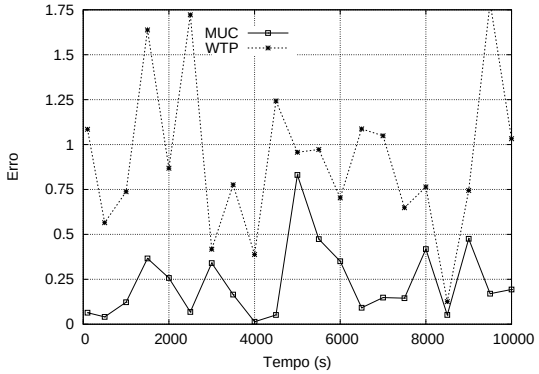

Figure 4. Cenário M1

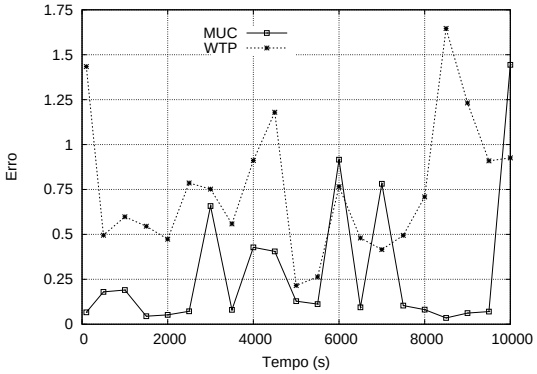

Figure 5. Cenário M2

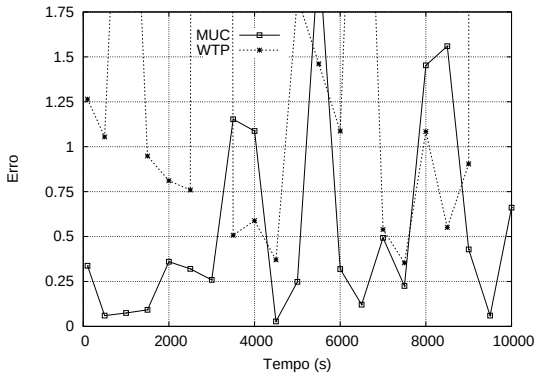

Figure 6. Cenário M3

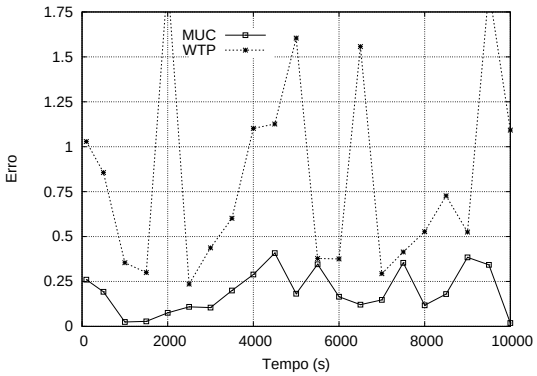

Figure 7. Cenário M4

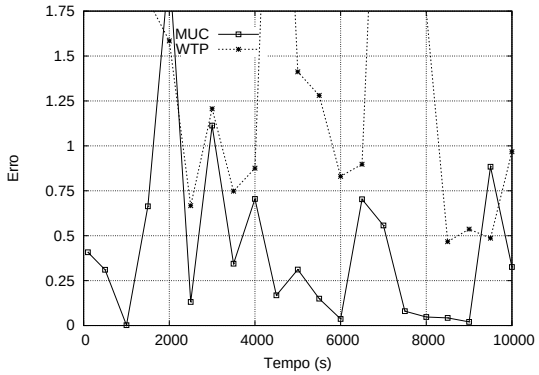

Figure 8. Cenário M5

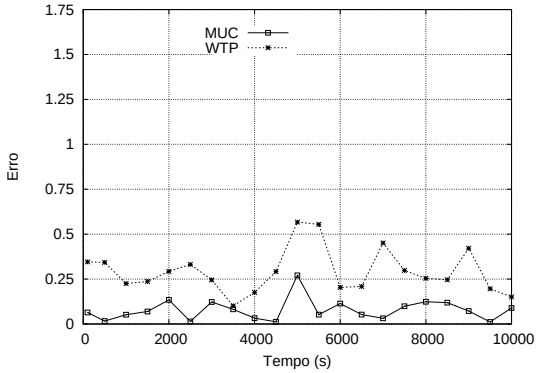

Figure 9. Cenário M6

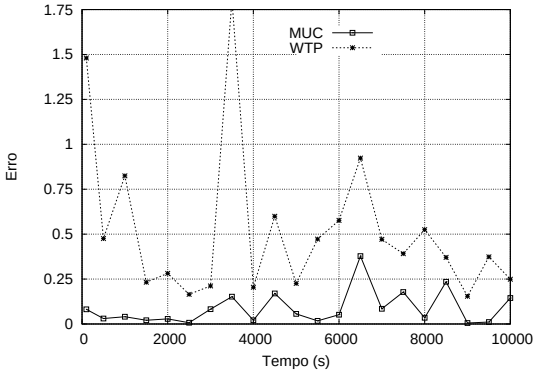

Figure 10. Cenário M7

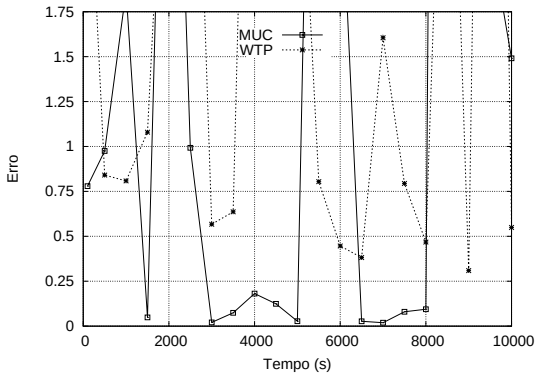

Figure 11. Cenário M8

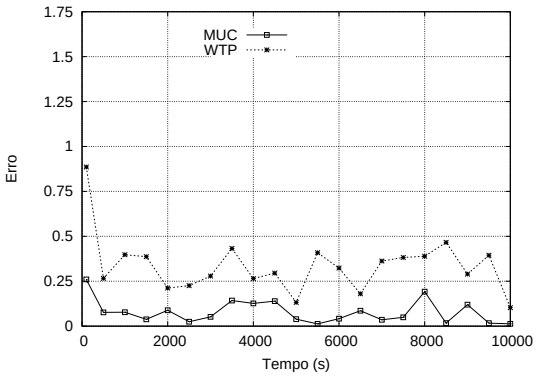

Figure 12. Cenário M9

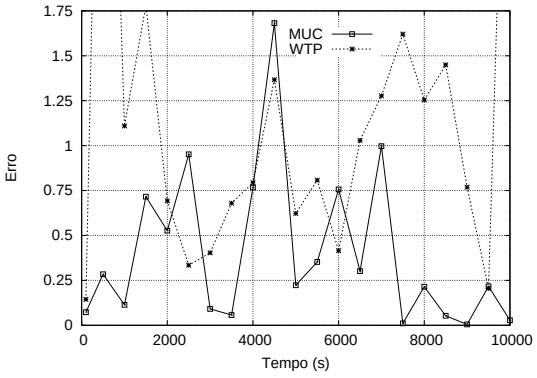

Figure 13. Cenário M10

6. Conclusão

Este trabalho propôs um novo algoritmo de escalonamento baseado na Métrica Única Combinada (MUC) para a implementação dos Serviços Proporcionais Diferenciados (PDS). O algoritmo foi testado em uma grande variedade de cenários, e os resultados de simulação demonstraram que o algoritmo proposto é capaz de implementar PDS com grande precisão. Durante a fase de avaliação de desempenho o algoritmo foi comparado com outras propostas disponíveis na literatura. Além de ter apresentado um melhor desempenho que as propostas atuais, o uso da MUC possibilita maior controle de QoS por parte do operador visto que a MUC é baseada em dois dos principais parâmetros para medidas de QoS. Com isso, acreditamos que o modelo PDS/MUC proporconará maior

valor para os operadores de rede poderem oferecer serviços mais consistentes dentro do modelo PDS. Como trabalhos futuros podemos citar uma possível extensão da MUC para considerar outros parâmetros de QoS, como por exemplo o descarte de pacotes.

References

A. Giesler, J. Hänle, A. K. and Pade, F. (1978). Free buffer allocation-an investigation by simulation. *Computer Networks*.

Adas, A. (1997). Traffic models in broadband networks. *IEEE Communications Magazine*, 35:82–89.

Blake, S. (1998). An architecture for differentiated services. *IETF RFC 2475*.

C. Dovrolis, D. S. and Ramanathan, P. (1999). Proportional differentiated services: Delay differentiation and packet scheduling. *ACM SIGCOMM Computer Communication Review*, 29:109–120.

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

Jaffe, J. M. (1981). Flow control power is nondecentralizable. *IEEE Transactions on Communications*, 29:1301–1306.

Kleinrock, L. (1976). *Queueing Systems*. Wiley, 2nd edition.

Lai, Y. and Chang, A. (2004). A non-work conserving scheduler to provide proportional delay differentiated services. *Global Telecommunications Conference*, 3:1723–1727.

M. H. Balter, M. E. C. and Murta, C. D. (1999). On choosing a task assignment policy for a distributed server system. *Journal of Pararllel and Distributed Computing*, 59:204–228.

M. K. H. Leung, J. C. S. L. and Yau, D. K. Y. (2001). Adaptive proportional delay differentiated services: Characterization and performance evaluation. *IEEE/ACM Transactions on Networking*, 9:801–817.

Niyato, D. and Hossain, E. (2005). Analysis of fair scheduling and connection admission control in differentiated services wireless networks. *IEEE International Conference on Communications*, 5:3137–3141.

P. Gevros, J. Crowcroft, P. K. and Bhatti, S. (2001). Congestion control mechanisms and the best-effort service model. *IEEE Network Magazine*, 15:16–26.

R. Braden, D. C. and Shenker, S. (1994). Integrated services in the internet architecture: an overview. *IETF RFC 1633*.

Salles, R. and Barria, J. A. (2002). Utility-based scheduling disciplines for adaptive applications over the internet. *IEEE Communications Letters*, 6:217–219.

Selvaraj, M. (2002). Scheduling for proportional differentiated services on the internet. Master's thesis, Faculty of Mississippi State University.

White, P. P. (1997). Rsvp and integrated services in the internet: A tutorial. *IEEE Communications*, 35:100–106.