Implementação de Mecanismos de Policiamento para Redes de Alta Velocidade

Célio Vinicius Neves de Albuquerque[1] *Marcio Faerman*[1]

Otto Carlos Muniz Bandeira Duarte

Grupo de Teleinformática e Automação - GTA
COPPE / EE - Programa de Engenharia Elétrica
Universidade Federal do Rio de Janeiro
CP 68504 - CEP 21945-970 Rio de Janeiro, RJ
E-mail: {celio, faerman, otto}@coe.ufrj.br

Resumo

Este artigo propõe implementações para os principais mecanismos de policiamento de tráfego para redes de alta velocidade. As implementações são baseadas em um ambiente de desenvolvimento que privilegia o desempenho e são descritas através de diagramas de blocos comentados. Foram considerados os mecanismos Balde Furado com suas variações, Janela Saltitante, Janela Saltitante Sincronizada, Janela Deslizante Contínua, Janela Deslizante Discretizada e Janela Móvel Particionada. Os mecanismos foram implementados nos ambientes UNIX e DOS. São apresentados dados comparativos enfocando a complexidade de implementação em termos da memória requerida, do número de temporizadores e da sua precisão, do tempo médio de processamento do sistema de comunicação e do tempo médio de processamento específico dos mecanismos. Os resultados confirmam a simplicidade de implementação dos mecanismos Balde Furado, Janela Saltitante e Janela Móvel Particionada.

Abstract

This paper proposes implementations of the well-known traffic policing mechanisms for high speed networks. The implementations are based on a high performance development environment and are described through commented block diagrams. There considered mechanisms are Leaky Bucket with variations, Jumping Window, Triggered Jumping Window, Continuous Sliding Window, Discrete Sliding Window and Time-Slotted Moving Window. These mechanisms were implemented in UNIX and DOS environments. A comparison of the results focusing the implementation complexity is presented considering required memory, number of timers and its precision, mean processing time of the communication system and mean mechanism processing time. The results confirm the low implementation complexity of Leaky Bucket, Jumping Window and Time-Slotted Moving Window mechanisms.

---

[1] Alunos de Mestrado do Programa de Engenharia Elétrica da COPPE/UFRJ.

1 Introdução

Devido à crescente demanda por serviços integrados, compostos das mais variadas mídias (voz, dados assíncronos e vídeo), o modo de transferência adotado para a Rede Digital de Serviços Integrados - Banda Larga (RDSI-BL) deve ser ao mesmo tempo flexível e eficiente. O Modo de Transferência Assíncrona - *Asynchronous Transfer Mode* (ATM) veio atender estes requisitos.

Para se obter uma maior eficiência da RDSI-BL, as fontes de tráfego devem ser multiplexadas estatisticamente, o que implica no fato de cada chamada dispor de uma banda passante alocada abaixo de sua taxa máxima de transmissão. Conseqüentemente, surge o complexo desafio de disciplinar e regular o tráfego através da rede, evitando-se situações de congestionamento causadoras de perda de células e atrasos indesejáveis.

A RDSI-BL deve atender de forma eficiente ao maior número possível de usuários e garantir a veiculação do tráfego contratado. Parâmetros de Qualidade de Serviço (*Quality of Service* - QoS), como taxa de perda de células, atraso, variação do atraso, etc., devem ser definidos, monitorados e garantidos pela rede.

O controle de tráfego da RDSI-BL é sem dúvida um dos problemas mais complexos a ser resolvido. A fim de garantir um bom desempenho da RDSI-BL é necessário um conjunto de mecanismos de controle de tráfego. Vários controles são previstos [1] tais como: controle de admissão de conexão, controle de prioridade, controle de congestionamento e controle de parâmetros de utilização/rede.

Este trabalho concentra-se no controle de parâmetros de utilização/rede que refere às ações efetuadas pela rede, durante a fase de transferência de informação, a fim de monitorar e controlar o tráfego do usuário. O principal objetivo deste controle é a proteção dos recursos da rede contra tráfegos "mal comportados" intencionais ou não. Estes tráfegos são aqueles que violam os parâmetros negociados durante a fase de estabelecimento de conexão podendo inclusive alterar a qualidade de serviço de outras conexões. Os mecanismos usados por este controle têm a função de detectar as possíveis violações dos parâmetros contratados pela conexão. Os mecanismos que executam esta função também são conhecidos como mecanismos de policiamento.

Além de monitorar e controlar as fontes de tráfego de forma eficaz, o mecanismo de policiamento deve demandar baixo custo de implementação e apresentar flexibilidade de parametrização. Vários mecanismos de policiamento têm sido propostos. Entre os mais citados na literatura tem-se o balde furado (*Leaky Bucket*) [2,3] e suas extensões [4], o contador de pico [5] e os mecanismos baseados em janela [2,5,6].

Este artigo propõe implementações para os mecanismos de policiamento de tráfego balde furado convencional e a sua variação com suavização do tráfego de saída, janela saltitante simples e sincronizada, janela deslizante contínua e discretizada e janela móvel particionada. As implementações são baseadas em um ambiente de desenvolvimento que privilegia o desempenho e são descritas através de diagramas de blocos comentados. Os mecanismos foram implementados em linguagem C nos ambientes UNIX e DOS. São apresentados dados comparativos enfocando a complexidade de implementação em termos da

memória necessária, do número de temporizadores e suas precisões e dos tempos médios de processamento de cada célula, geral e específico de cada mecanismo.

Na seção 2 os mecanismos de policiamento considerados neste trabalho são descritos. A seção 3 apresenta a arquitetura de implementação e os seus módulos. A seção 4 propõe implementações para os mecanismos. A seção 5 apresenta os resultados numéricos obtidos na implementação e compara os mecanismos. Finalmente as conclusões são apresentadas na seção 6.

2 Mecanismos de Policiamento

Dado que uma chamada é admitida, a rede se protege de fontes abusivas través de um controle de parâmetros, denominado policiamento, garantindo que a fonte de tráfego não violará os parâmetros contratados, como taxa máxima, taxa média e comprimento de rajada. Além disso, classes de serviço com menor sensibilidade ao atraso podem ter seu tráfego moldado ou suavizado em esquemas de controle de taxa.

As células que compõem a parcela de tráfego considerada pelo controle de policiamento como violadora, são simplesmente descartadas ou "marcadas". Nesta situação, as células marcadas são transferidas, caso não sejam causadoras de congestionamento para a rede, com possível sobretaxa para o usuário.

Entre os mecanismos de policiamento mais citados na literatura têm-se: Balde Furado, Janela Saltitante e Janela Deslizante ou Retangular.

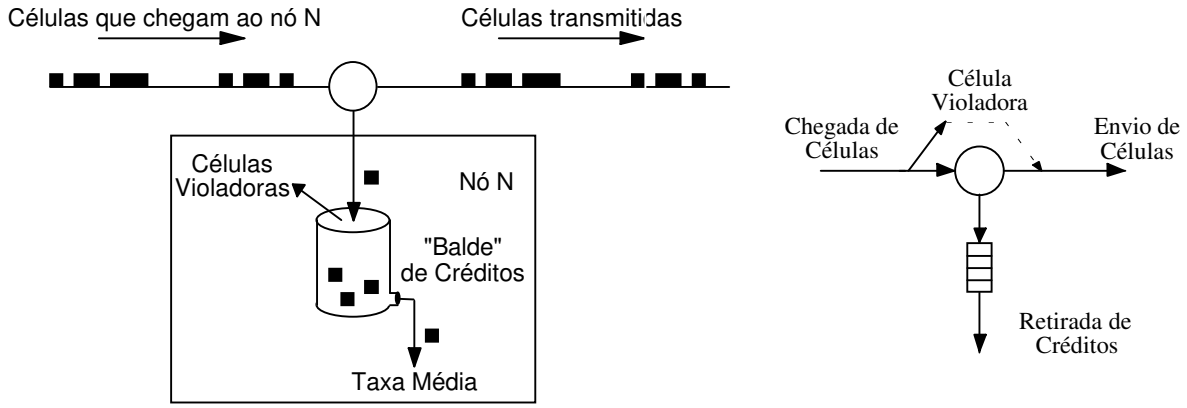


Figura 1: Representações do mecanismo Balde Furado.

Balde Furado

Uma das técnicas para policiamento mais citadas na literatura é a conhecida como Balde Furado - BF (*Leaky Bucket* - LB). Na técnica do balde furado uma célula só é transmitida quando ela pode depositar um crédito em um depósito de créditos, o balde furado (figura 1). O depósito de créditos tem um tamanho finito denotado por *C*. Se o depósito de créditos estiver cheio, então a célula é considerada violadora, podendo ser descartada ou marcada. Os créditos são retirados ou "vazam" do depósito à taxa média contratada. A capacidade do depósito pode ser encarada como o comprimento máximo de rajada permitido (desconsiderando-se créditos

retirados durante a rajada) já que somente *C* células poderão ser transmitidas durante um intervalo menor que o intervalo de atualização de créditos.

Janela Saltitante

O mecanismo Janela Saltitante - JS (*Jumping Window* **-** JW) considera janelas com períodos de tempo fixos e iguais a *T* justapostas ao longo do tempo. Assim, uma janela tem início imediatamente após o término da janela anterior. Durante uma janela somente *C* células poderão ser submetidas a rede. Caso *C* células já tenham sido recebidas pela rede durante uma janela, as células seguintes serão consideradas violadoras e, conseqüentemente, descartadas ou marcadas (figura 2).

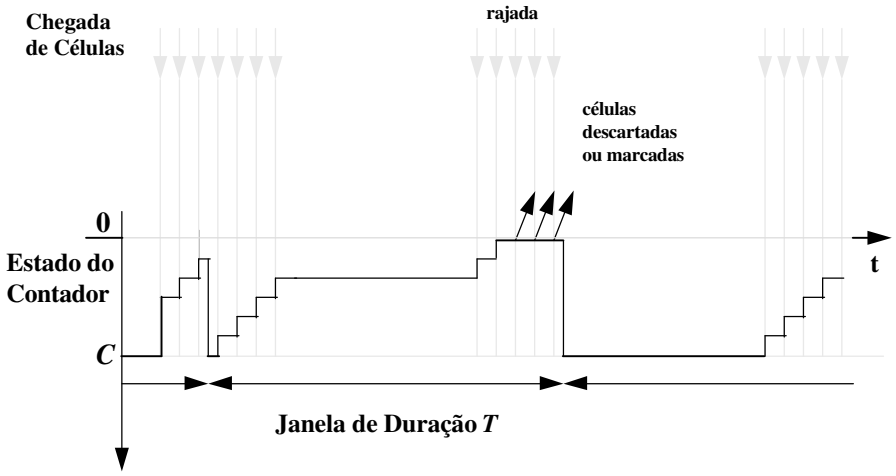


Figura 2: O mecanismo Janela Saltitante.

Janela Saltitante Sincronizada

O mecanismo Janela Saltitante Sincronizada - JSS (*Triggered Jumping Window* - TJW) é semelhante ao mecanismo Janela Saltitante porém o início de uma janela está condicionada a (sincronizada com a) chegada de uma célula. Este procedimento evita períodos de silêncio no início de uma janela permitindo assim um controle mais acurado do tráfego.

Janela Delizante Contínua

O mecanismo de Janela Deslizante Contínua - JDC (*Sliding Window* - SW) também limita em *C* o número máximo de células submetidas a rede durante uma janela com duração *T*. A diferença em relação a janela saltitante consiste na forma como são consideradas as janelas. O início de uma janela está associado ao instante de chegada de cada célula e seu término exatamente *T* unidades de tempo depois. Assim, várias janelas superpostas vão sendo abertas e fechadas ao longo do tempo e todas restringindo o tráfego em *C* células no máximo, em qualquer período de tempo *T*.

No entanto, sua complexidade de implementação é relativamente alta, tendo em vista a necessidade do armazenamento dos instantes de chegadas das células, disparando um temporizador para cada chegada de célula, além de armazenar o número de chegadas, de até *C*

células. Isto pode ser extremamente custoso para janelas com intervalo de longa duração. A figura 3 ilustra o comportamento deste mecanismo.

Janela Deslizante Discretizada

A Janela Deslizante Discretizada - JDD é semelhante à Janela Deslizante Contínua. A diferença consiste no fato dos instantes das chegadas serem aferidos em segmentos de tempo, cuja duração é igual ao tempo de transmissão de uma célula, ou seja, a aferição das chegadas é feita no tempo discreto. Uma das vantagens deste esquema é relativa ao número de temporizadores necessários a sua implementação, que será explicada na seção 4. O esquema de armazenamento dos instantes de chegadas é realizado a partir da associação de um *bit* para cada posição de *T* onde a célula pode ter chegado. Assim como o esquema anterior, para grandes valores de *C*, este esquema também apresentará um custo elevado.

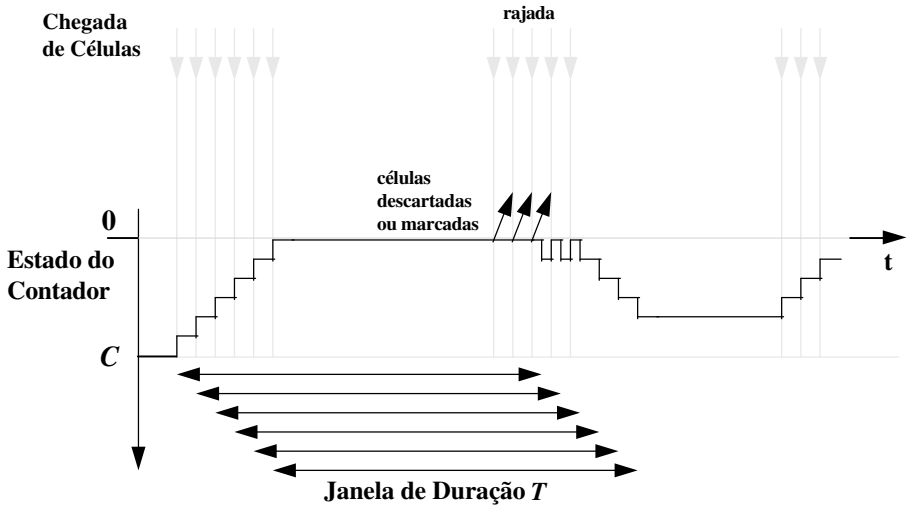


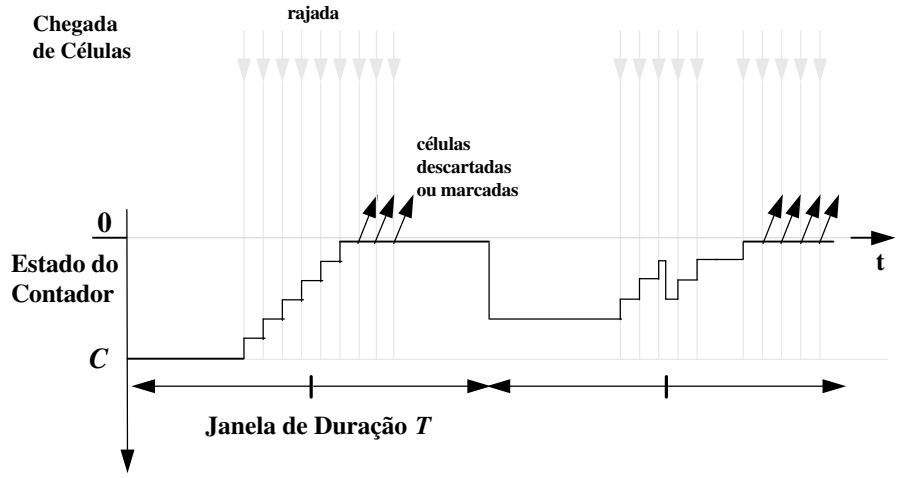
Figura 3: O mecanismo Janela Deslizante Contínua.

Janela Móvel Particionada

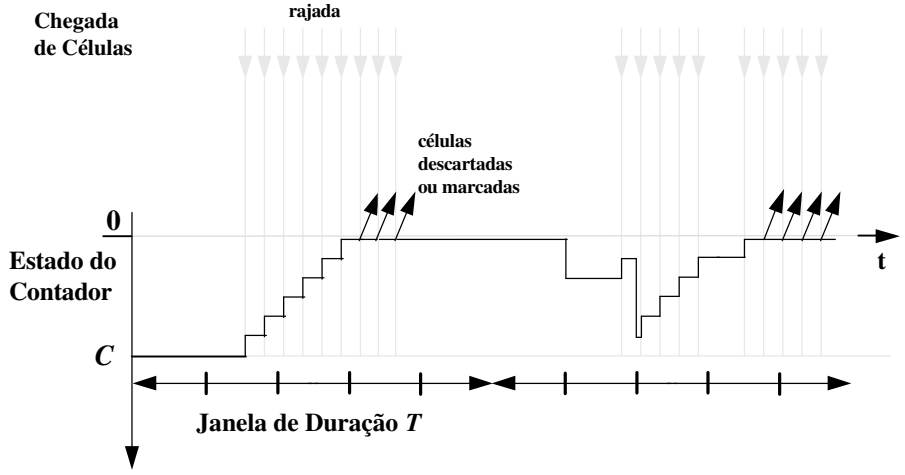
No mecanismo Janela Móvel Particionada - JMP [18] a janela de duração *T* é subdividida em *P* períodos de tempo iguais, denominados partições. Analogamente ao demais mecanismos baseados em janela, a cada chegada de célula um contador, se positivo, é decrementado. O contador é incrementado do número de células que chegaram durante uma partição $p_i.T$ unidades de tempo após o início desta partição (figura 4).

Observa-se que para o número de partições *P* igual a um, este mecanismo é equivalente à janela saltitante. Quando *P* tende a infinito, a duração de cada partição tende a zero e este mecanismo aproxima-se do mecanismo janela deslizante.

O JMP busca um compromisso entre a simplicidade da janela saltitante e a eficiência da janela deslizante em relação a memorização do tráfego, policiamento acurado da taxa média e restrição a comprimento de rajadas.



(a)



(b)

Figura 4: O mecanismo Janela Móvel Particionada para a) 2 partições e b) 5 partições.

Suavização e controle das taxas média e máxima

Normalmente os mecanismos de policiamento controlam apenas uma taxa de transmissão. Para policiar a taxa máxima e a taxa média simultaneamente é necessário interconectar dois módulos de mecanismos de policiamento. De uma maneira geral, isto é obtido através do cascateamento do mecanismo na sua forma convencional com um outro que tenha a função de espaçador de células à taxa máxima. A figura 5(a) mostra este esquema para o balde furado, onde o espaçador consiste de um balde furado com depósito de tamanho $C_{máx} = 1$ e com taxa de atualização de créditos igual a taxa máxima contratada.

Uma outra classe de controle de tráfego é a moldagem de tráfego (*traffic shapping*). Neste tipo de mecanismo "molda-se" o tráfego de forma a respeitar os parâmetros negociados. Em relação a taxa negociada a moldagem se comporta como um suavizador de tráfego (*traffic smoothing*) e pode ser realizada pela simples introdução de um *buffer* que tenha a finalidade de armazenar as células que seriam descartadas. Embora não ocorra o descarte das células, este armazenamento tem o inconveniente de introduzir atrasos. A figura 5(b) apresenta o BF com

suavização do tráfego na entrada e a figura 5(c) com suavização na saída. As células consideradas violadoras são armazenadas até que possam ser transmitidas sem penalização.

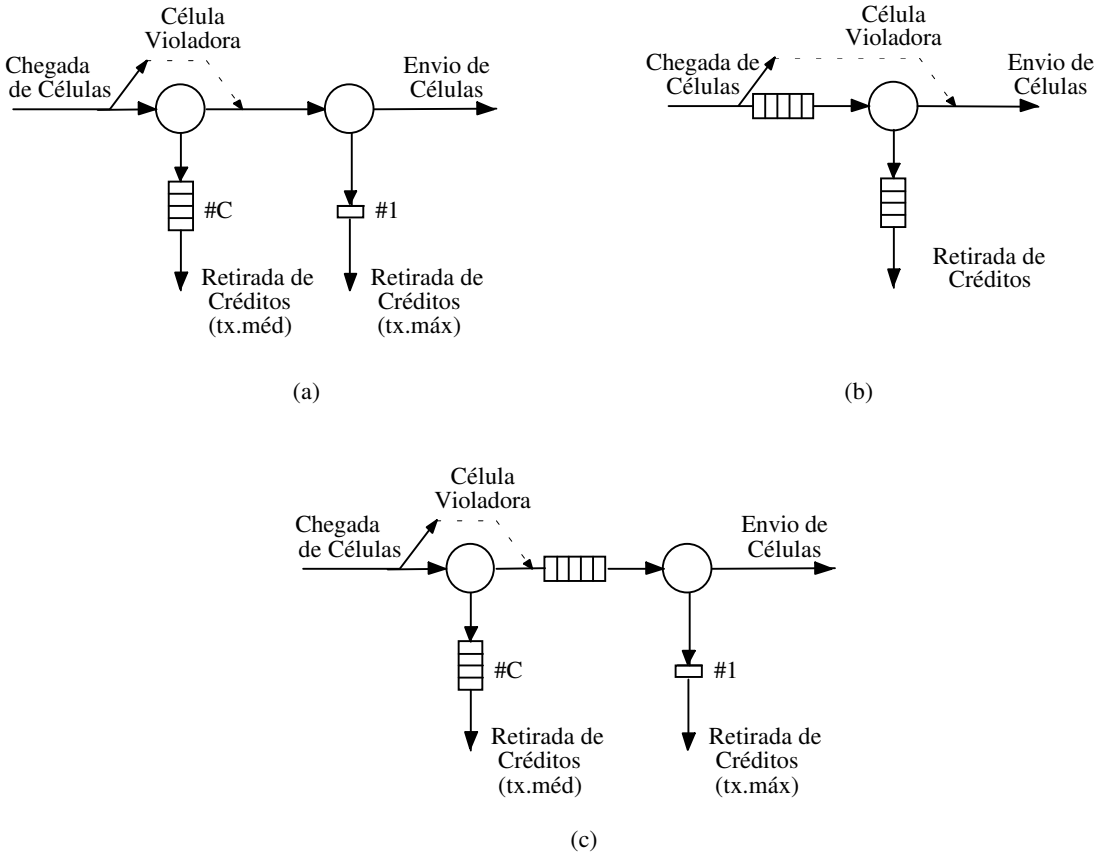


Figura 5: Variações do mecanismo Balde Furado.

3 Arquitetura de Implementação

A implementação dos mecanismos de policiamento fez uso de um ambiente de desenvolvimento de sistemas de comunicação de alto desempenho com garantias de qualidade de serviço (figura 6). Esta arquitetura de implementação consiste de três módulos básicos utilizados no desenvolvimento de todo sistema de comunicação e de alguns módulos específicos, opcionais, que definem as características do sistema de comunicação a ser desenvolvido. Os módulos básicos são o Módulo de Gerenciamento de Memória (GM), o Módulo de Escalonamento de Tarefas (ET) e do Módulo de Gerenciamento de Temporizações (GT). Os módulos específicos são o Módulo de Gerenciamento do Sistema, o Protocolo de Comunicação, o Módulo de Sincronização e o Módulo de Garantia de Qualidade de Serviço que engloba o Módulo de Controle de Tráfego e Policiamento.

Os módulos específicos visam oferecer um conjunto de facilidades para a implementação do sistema O módulo de gerenciamento do sistema configura os demais módulos de acordo com as características do sistema desejado. O protocolo de comunicação [11] visa oferecer o conjunto de funcionalidades existentes nos principais protocolos de alta velocidade. As funcionalidades são configuráveis ou por primitivas de serviços ou pelo módulo de gerenciamento do sistema. O módulo de sincronização visa coordenar a ordenação temporal de informações multimídia (voz, vídeo, gráficos, dados, etc.). Através de Redes de

Petri temporizadas [12], este módulo implementa os tipos de sincronização inter e intra-mídia. O módulo de Garantia de Qualidade de Serviço possui implementados os mecanismos *Virtual Clock* [13], *Stop-and-Go* [14], *Jitter-Earliest-Due-Date* [15] e *Hierarchical Round Robin* [16]. Este módulo engloba ainda as implementações dos mecanismos de policiamento de tráfego balde furado e balde furado com suavização do tráfego de saída, janela saltitante, janela saltitante sincronizada, janela deslizante contínua e janela móvel particionada.

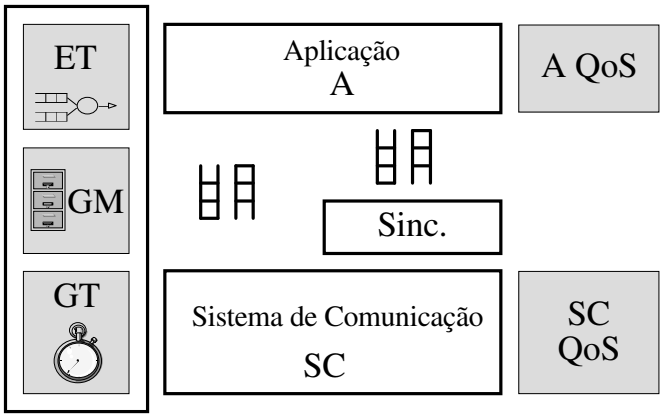
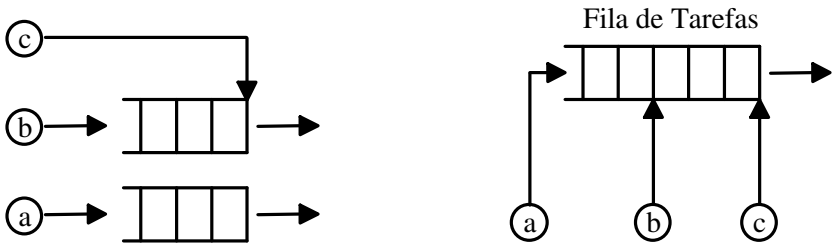


Figura 6: A arquitetura de implementação.

O Módulo de Gerenciamento de Memória, de maneira a evitar excesso de fragmentação, divide a memória em regiões de transmissão e recepção. Este modelo permite um uso mais ordenado da memória tornando o gerenciamento mais rápido e eficiente. Isto é possível porque os fluxos de transmissão e recepção são independentes. Estas regiões são ainda subdivididas em quatro partes. Duas estão reservadas para estruturas de dados de tamanho fixo das interfaces entre camadas adjacentes e entre os módulos básicos. Uma região é alocada para estruturas de tamanho variável usadas na passagem de parâmetros entre camadas de um mesmo sistema. A quarta região de memória alocada de acordo com cada fluxo estabelecido e contém estruturas de dados de tamanho fixo, denominados blocos, que são usados pelos dados que serão efetivamente transmitidos. Diferentemente dos gerenciadores de memória de propósito geral, este gerenciador armazena as informações de controle de alocação separado dos dados, permitindo a implementação eficiente e sem cópias de funções de segmentação, remontagem, concatenação, separação, liberação de parte dos dados, entre outras.

O Módulo de Escalonamento de Tarefas armazena e determina a seqüência das tarefas a serem executadas. Além disso, possui funções para permitir níveis de prioridade para os fluxos em tempo real. O comportamento da fila de tarefas é modelado como o de duas filas onde um elemento da fila de menor prioridade só é servido quando a fila de maior prioridade estiver vazia. De forma a priorizar o atendimento de tarefas de altíssima prioridade, como por exemplo o estouro de temporizadores, estas tarefas são inseridas na primeira posição da fila de tarefas (ponto **c** da figura 7). Este ponto é também o ponto de saída de todas as tarefas. Um segundo nível de prioridade é dado às tarefas dos fluxos em tempo real inseridas no ponto **b**. As demais tarefas de menor prioridades são inseridas no final da fila de tarefas (ponto **a**). Implementando o modelo de duas filas com prioridades através de uma única fila com três pontos de entrada e um único ponto de saída evitamos testes a cada retirada da fila para verificar se a fila de maior prioridade está vazia ou não.



Modelo Implementação

a - ponto de entrada das tarefas de menor prioridade.
b - ponto de entrada das tarefas de maior prioridade.
c - ponto de entrada das tarefas de prioridade máxima
e de saída de todas as tarefas.

Figura 7: Módulo de escalonamento de tarefas.

O Módulo de Gerenciamento de Temporizações é bastante eficiente. Ele é baseado na proposta de Varghese & Lauck [17], no qual uma analogia a um relógio real é simulada através de um contador em módulo. Esta implementação privilegia o desempenho, evitando testes desnecessários a cada interrupção, permitindo portanto a criação de um grande número de temporizadores dinamicamente. São fornecidas funções para disparar temporizadores simples e auto-disparáveis, reiniciar, interromper e obviamente sinalizar o estouro de um temporizador. Nesta arquitetura, o estouro de um temporizador é tratado como uma primitiva de serviço, ou seja, a indicação de estouro (evento) é colocada na fila apropriada da interface, e a tarefa correspondente é colocada na fila de prioridade máxima do escalonador de tarefas.

4 Implementação de Mecanismos de Policiamento

O objetivo desta seção é apresentar propostas de implementações dos mecanismos de policiamento considerados neste trabalho. O procedimento de execução dos mecanismos é descrito através da forma de diagramas de blocos, com base na arquitetura de implementação utilizada. A complexidade das implementações será analisada em seqüência.

Balde Furado

O mecanismo balde furado pode ser implementado por meio de um contador de créditos e de um temporizador auto-disparável. Ao se completar o estabelecimento da conexão e se determinar os parâmetros do mecanismo, dispara-se o temporizador auto-disparável responsável por incrementar os créditos a cada ($1/\lambda_{méd}$) unidades de tempo, onde $\lambda_{méd}$ é a taxa média policiada. O contador de créditos é incrementado até o limite máximo *Cmáx*. A cada célula submetida ao mecanismo, se o crédito for positivo, estes são decrementados e a célula é transmitida. Este procedimento está ilustrado no fluxograma da figura 8.

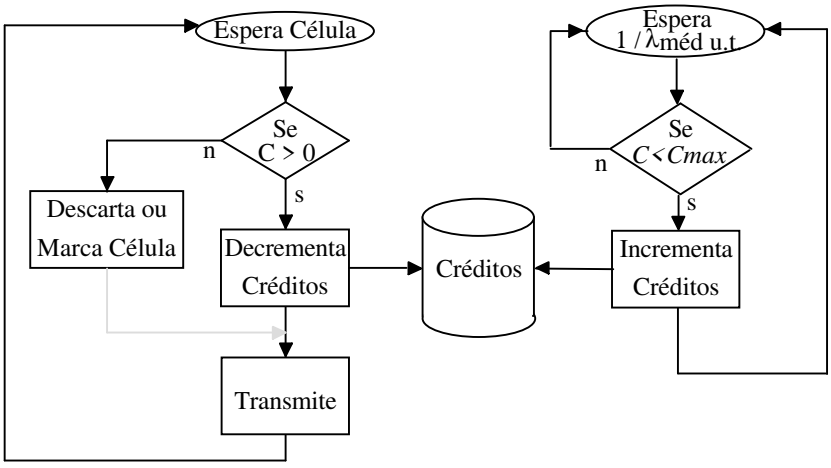


Figura 8: Diagrama em blocos da implementação do balde furado.

Uma outra implementação deste mecanismo, proposta por Dittmann *et al* [6], é baseada em marcações de tempo de forma que a atualização do contador é feita através de uma subtração a cada chegada de célula. O valor a ser subtraído dos créditos é calculado de acordo com o tempo decorrido desde a chegada da última célula. Este esquema é descrito através do seguinte pseudocódigo, e possui as desvantagens de requerer marcas de tempo demasiadamente precisas e de executar operações de subtração não-unitárias.

A cada chegada de célula faça:
1. Δt := tempo_ultima_chegada - tempo_atual;
2. tempo_ultima_chegada := tempo_atual;
3. se $C < \Delta t \times \lambda_{méd}$
   então $C := 0$;
   senao $C := C - \Delta t \times \lambda_{méd}$;
4. se $C > C_{máx} - 1$
   então descarta célula;
   senao $C := C + 1$;

Balde Furado com Suavização do Tráfego de Saída

Foi implementado ainda o mecanismo do balde furado com limitação da taxa máxima de transmissão, ou com suavização do tráfego de saída. Esta variação do balde furado é implementada pela adição ao mecanismo original de um temporizador auto-disparável de controle da taxa máxima de saída e de uma fila FIFO. Por simplicidade de implementação foi utilizada uma lógica de evolução do contador inversa à descrita na seção 2. Neste esquema (figura 9), a cada célula entregue ao mecanismo, se o crédito for positivo, estes são decrementados e a célula é enviada para a fila FIFO de saída. O temporizador de suavização do tráfego de saída é responsável por retirar as células da fila FIFO e enviá-las para serem transmitidas a cada $1/\lambda_{máx}$ unidades de tempo, onde $\lambda_{máx}$ é a taxa máxima de transmissão contratada.

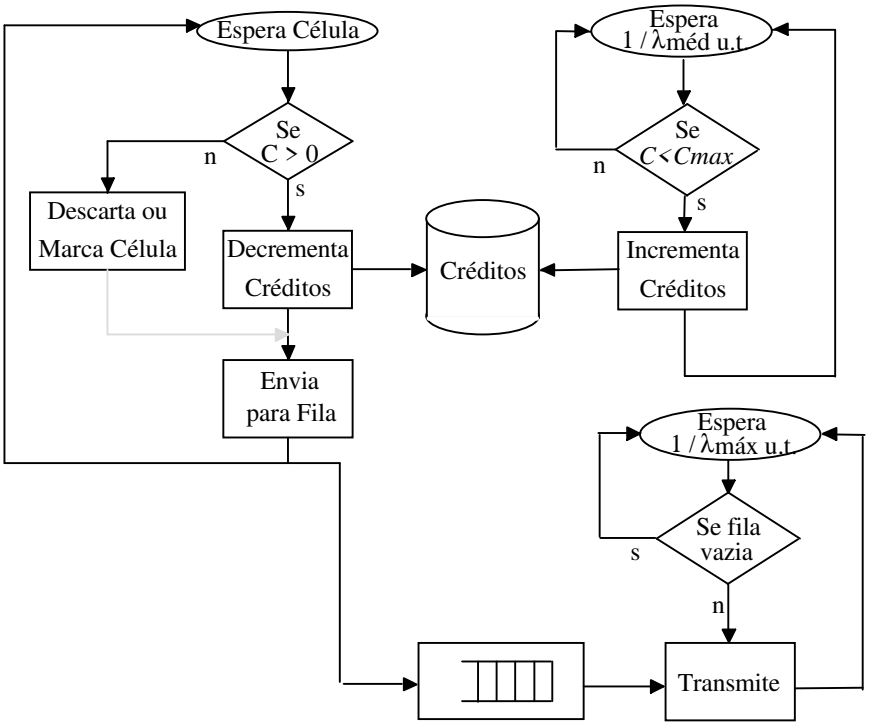


Figura 9: Diagrama em blocos da implementação do balde furado com suavização do tráfego de saída.

Janela Saltitante

O mecanismo de janela saltitante pode ser implementado por meio de 1 temporizador e 1 contador, de acordo com o seguinte procedimento. Ao concluir a fase de estabelecimento da conexão e de determinação dos parâmetros do mecanismo, o JS cria um temporizador auto-disparável responsável pela atualização do credito a cada *T* unidades de tempo até o termino da conexão. Paralelamente, a cada célula entregue ao JS, este testa a variável crédito e se o resultado for positivo, o mecanismo decrementa os créditos e entrega a célula para ser transmitida multiplexada com as demais conexões; caso contrário, a célula é considerada violadora e neste caso, optou-se por descartar a célula.

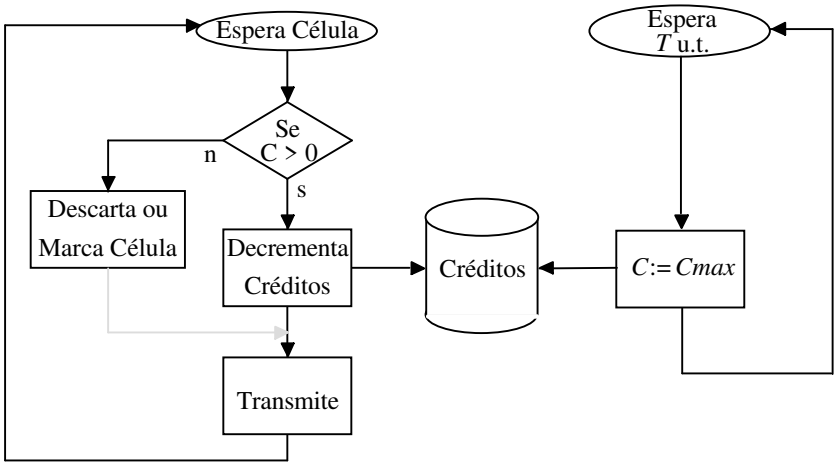


Figura 10: Diagrama em blocos da implementação da janela saltitante.

Janela Saltitante Sincronizada

Este mecanismo foi implementado de forma similar a janela saltitante, no entanto o temporizador de atualização de créditos é criado no instante da chegada da primeira célula de cada janela de comprimento *T*. Isto é implementado de forma bastante simples através de uma pequena alteração na máquina de estados do mecanismo, onde após o estabelecimento da conexão e após cada estouro do temporizador, a janela saltitante sincronizada fica no estado "Espera Primeira Célula". Ao receber uma célula neste estado, dispara o temporizador, decrementa o crédito e passa para o estado de transmissão normal.

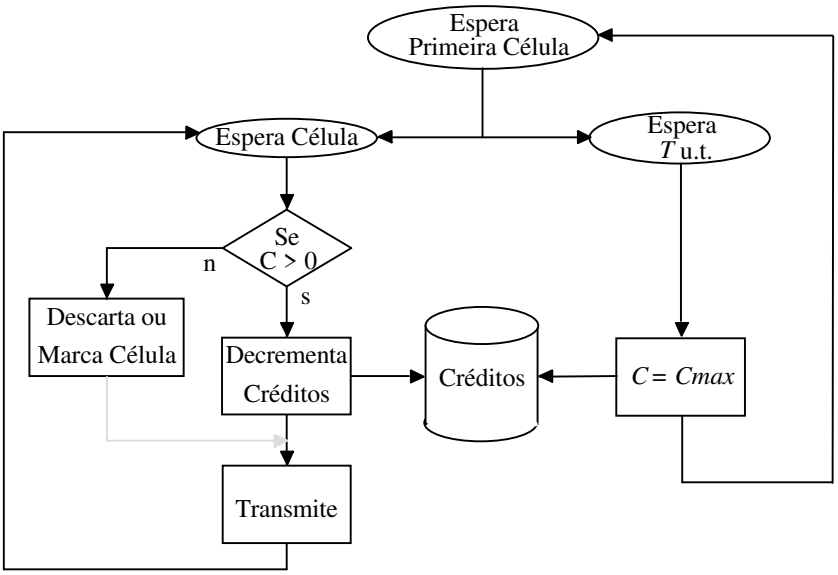


Figura 11: Diagrama em blocos da implementação da janela saltitante sincronizada.

Janela Deslizante Contínua

O mecanismo de janela deslizante contínua- JDC (*Sliding Window* - SW) pode ser implementado por meio de diversos temporizadores e por um contador. A cada célula entregue ao JDC, se o crédito for positivo, o mecanismo decrementa o crédito e dispara um temporizador com duração *T* unidades de tempo. No estouro de cada temporizador a variável crédito é incrementada. A utilização de um grande e variável número de temporizadores implica em uma maior utilização da memória e da CPU, além de dificultar uma reserva de recursos eficiente para a conexão.

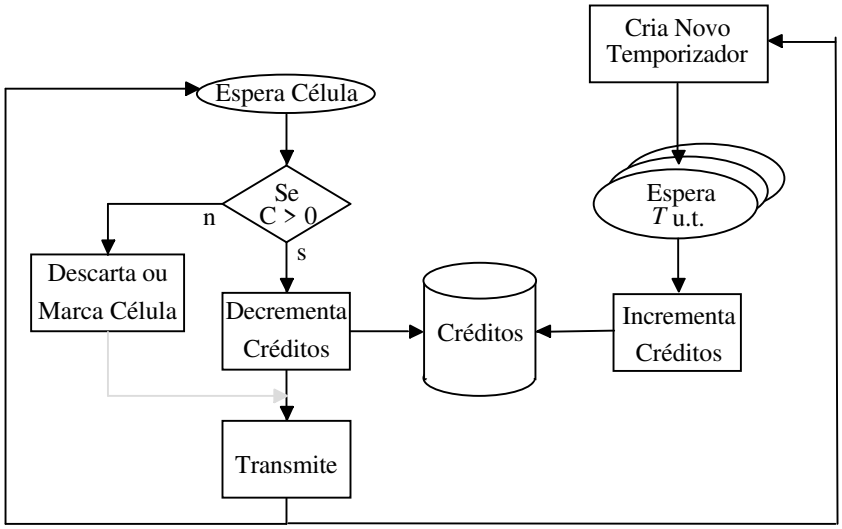


Figura 12: Diagrama em blocos da implementação da janela deslizante contínua.

Janela Deslizante Discretizada

O mecanismo janela deslizante discretizada - JDD é implementado de forma semelhante ao JDC, porém com o uso de um único temporizador e de um vetor de deslocamento de *bits*, no qual cada bit deste vetor registra a chegada de uma célula na unidade de tempo correspondente, onde em cada unidade de tempo (p. ex. t_{cel} = 2,7μs para taxa de transmissão de 155 Mbps) permite a chegada de apenas uma célula. Neste esquema, além do vetor de *bits*, existe a necessidade de apenas um temporizador e um contador. Este esquema permite uma melhor utilização da memória e de CPU.

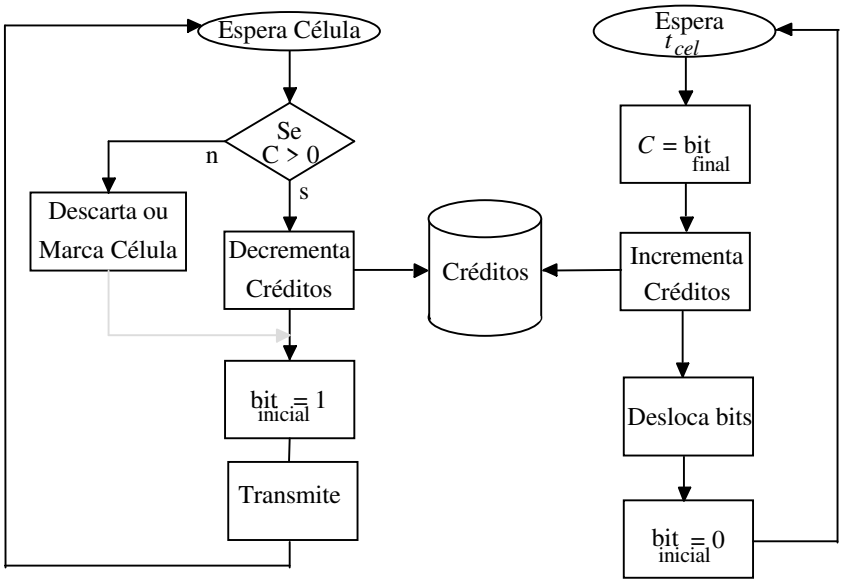


Figura 13: Diagrama em blocos da implementação da janela deslizante discretizada.

Janela Móvel Particionada

O mecanismo janela móvel particionada - JMP foi implementado por meio de um temporizador auto-disparável e de um vetor FIFO de tamanho fixo alocado no estabelecimento

da conexão de acordo com os parâmetros do tráfego e do mecanismo. Diferentemente dos outros mecanismos baseados em janela, além dos parâmetros *T* e *C*, este mecanismo define o parâmetro *P* que indica o número de partições em cada janela de comprimento *T*. Este parâmetro é extremamente importante na definição da utilização da memória e da CPU. Quanto maior o número de partições *P*, maior será a memória necessária para alocação do vetor FIFO, maior será a utilização da CPU e mais próximo este mecanismo estará do JDD. Quanto menor o número de partições, menor será a memória necessária, menor será a utilização da CPU e mais próximo estará do mecanismo JS.

O temporizador é responsável pela atualização do crédito a cada *T/P* unidades de tempo. Esta atualização corresponde ao incrementado do crédito do número de células que chegaram durante a partição p_i iniciada *T* unidades de tempo antes desta atualização. Este valor está armazenado na última posição do vetor FIFO.

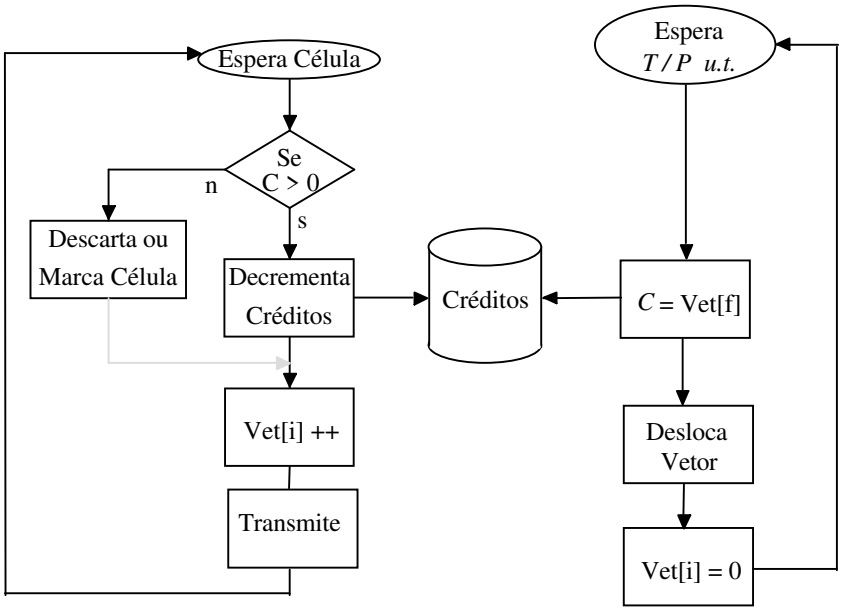


Figura 14: Diagrama em blocos da implementação da janela móvel particionada.

5 Resultados

Esta seção visa apresentar dados comparativos obtidos a partir de uma implementação específica, enfocando a complexidade de implementação dos mecanismos. São analisados a quantidade de memória necessária, o número de temporizadores e a sua precisão, o tempo médio de processamento total de cada célula e o tempo médio de processamento específico do mecanismo para cada célula.

A tabela 1 contém informações comparativas entre os mecanismos. A notação $\{x\}$ representa o número de *bits* necessário para representar o valor *x*. O tempo de processamento de cada célula, geral e específico de cada mecanismo, foi medido em uma estação de trabalho SUN SPARC 2. Os tempos de todos os mecanismos foram obtidos através da média de 1000 medições, todas feitas nas mesmas condições de operação da máquina. A estação estava dedicada para a execução dos processos dos mecanismos. A computação dos tempos foi obtida através de um relógio virtual que computa separadamente o tempo de processamento de

cada processo. Quanto aos resultados obtidos, observa-se que os mecanismos balde furado, janela saltitante e janela saltitante sincronizada são os mais eficientes, com tempos de processamento, geral e específico, da ordem de 160 μs e 140μs. O mecanismo janela deslizante contínua é um dos mais custosos em relação ao tempo de processamento, 192 μs e 167 μs, devido ao disparo dos temporizadores a cada chegada de célula. O mecanismo balde furado com suavização do tráfego de saída teve um custo de processamento alto devido à manutenção da fila de saída. Já o mecanismo janela móvel particionada mostrou possuir um tempo médio de processamento de cada célula relativamente independente do número de partições e de valor intermediário ao JS e ao JDD, da ordem de 176 μs e 150 μs.

Tabela 1: Comparação entre os mecanismos.

Mecanismo	**BF**	**BF suav.**	**JS**	**JSS**	**JDC**	**JDD**	**JMP**
Tempo de Proc. Total	165 μs	194 μs	160 μs	165 μs	192 μs	176 μs	176 μs
Tempo de Proc. do Mecanismo	143 μs	170 μs	141 μs	142 μs	167 μs	150 μs	150 μs
Número de Temporizadores	1	2	1	1	C	1	1
Precisão dos Temporizadores	variável	variável	baixa	baixa	alta	alta	ajustável
Memória para Temporizadores	{Tmp.}	{Tmp.}	{Tmp.}	{Tmp.}	$C \times$ {Tmp.}	{Tmp.}	{Tmp.}
Memória para Contadores e Parâmetros	$\{C\}$ + $\{C_{\max}\}$	$\{C\}$ + $\{C_{\max}\}$	$\{C\}$ + $\{C_{\max}\}$	$\{C\}$ + $\{C_{\max}\}$	$\{C\}$ + $\{C_{\max}\}$	$\{C\}$ + $\{C_{\max}\}$	$\{C\}$ + $\{C_{\max}\}$ + $\{P\}$

Um fator importante a ser considerado é a precisão requerida pelos temporizadores. Observa-se que o mecanismo janela deslizante contínua necessita do maior valor, C temporizadores, além dos mesmos exigirem uma altíssima precisão, devendo ser disparados na chegada de cada célula, e portanto, em qualquer instante de tempo. Deve-se ressaltar que quanto maior o número de temporizadores, maior será a utilização da CPU. A janela deslizante discretizada já é menos crítica, porém há ainda a necessidade de uma precisão bastante alta, por exemplo 2.7 μs para a taxa de transmissão de 155 Mbps. O balde furado necessita de temporizadores da ordem do inverso da taxa média policiada e da taxa máxima de transmissão. Para taxas de por exemplo 100 células / segundo, o temporizador deve ser de 10 ms. Já a janela saltitante utiliza de um temporizador da ordem de T unidades de tempo, e portanto sua precisão, em média, não será crítica. O mecanismo janela móvel particionada permite a utilização de temporizadores com toda a faixa de precisão, desde a utilizada na JS até a utilizada na JDC, conforme o número de partições utilizadas.

A quantidade de memória necessária para cada mecanismo pode ser dividida na memória utilizada pelos temporizadores, na memória utilizada pelos contadores e parâmetros dos mecanismos e na memória de controle auxiliar. Esta última é necessária apenas para os mecanismos janela deslizante e janela móvel particionada, que necessitam, respectivamente, de

$$\left\lceil \frac{T}{t_{cel}} \right\rceil \text{ e } P \times \left\{ T \Big/ \left(P \times t_{cel} \right) \right\} \text{ bits}$$

para memorização do tráfego. As demais regiões de memória estão ilustradas na tabela 1. Pode-se reparar que a janela deslizante contínua necessita de $C \times \{\text{Tmp.}\}$ *bits* para os temporizadores, enquanto a janela móvel particionada necessita de {P} *bits* para armazenar o número de partições.

6 Conclusões

Este trabalho apresentou a implementação de mecanismos de policiamento de tráfego. As implementações foram baseadas em uma arquitetura de implementação que priviliegia o desempenho, através de módulos otimizados de gerenciamento de memória, de escalonamento de tarefas com prioridades e de gerenciamento eficiente de temporizações, além de módulos opcionais para qualidade de serviço e sincronização de mídias.

Foram implementados os mecanismos balde furado convencional e a sua variação com suavização do tráfego de saída, janela saltitante simples e sincronizada, janela deslizante contínua e discretizada e janela móvel particionada. As implementações foram feitas em linguagem C nos ambientes DOS e UNIX.

Foi analisada a complexidade de implementação dos mecanismos, baseando-se nos resultados obtidos da implementação, em termos da quantidade de memória necessária, do número de temporizadores e da sua precisão, do tempo médio de processamento total de cada célula e do tempo médio de processamento específico dos mecanismos para cada célula.

Os resultados obtidos demonstram a maior complexidade do mecanismo janela deslizante contínua e a simplicidade de implementação dos mecanismos balde furado e janela saltitante. A janela móvel particionada apresenta a vantagem de ser flexível que de acordo com a sua parametrização oferece um conjunto de funcionalidades que varia desde a janela saltitante, com sua simplicidade, até a janela deslizante contínua, com seu acurado controle da taxa média. Além disso os mecanismos balde furado, janela saltitante e janela móvel particionada possuem a vantagem de permitir uma implementação com apenas um único temporizador auto-disparável. Esta característica em comparação com os mecanismos janela saltitante sincronizada e janela deslizante contínua requerem uma menor complexidade e menor precisão em termos de temporização.

Deve-se atentar para o fato de a variação do mecanismo balde furado com suavização do tráfego de saída necessitar de uma alta complexidade de implementação, devendo as inúmeras variações deste mecanismo ser cuidadosamente analisadas a fim de se avaliar o compromisso entre as vantagens e os custos de implementação de tais variações.

7 Referências

[1] ITU-T Recomendação I.311, B-ISDN General Networks Aspects, Genebra, 1991.

[2] E. Rathgeb, "Modeling and Performance Comparision of Policing Mechanisms for ATM Networks", IEEE JSAC, pp.325-334, vol. 9, no. 3, abril de 1991.

[3] M. Butto, E. Cavallero, A. Tonietti, "Effectiveness of the Leaky Bucket Policing Mechanism in ATM Networks", IEEE JSAC, pp. 335-342, vol. 9, no. 3, abril de 1991.

[4] A. Berger,"Performance Analysis of a Rate Control Throttle Where Tokens and Jobs Queue", IEEE INFOCOM, pp. 30-38, 1990.

[5] J. Junior, J. Monteiro, "On The Efficiency of Policing Mechanisms for ATM Networks", IEEE ITS'94, pp.470-474, agosto de 1994.

[6] L. Dittman, S. Jacobsen, K. Moth, "Flow Enforcement Algorithms for ATM Networks", IEEE JSAC, pp. 343-349, vol 9. no. 3, abril de 1991.

[7] C. V. Albuquerque, M. L. Lanza, O. C. Duarte, "Sistemas de Comunicação Abertos em Redes Locais: Uma Implementação e Medidas de Desempenho em Ambientes de Computadores Pessoais", XIII Congresso Sociedade Brasileira de Computação (SEMISH), pp. 327-329, Florianópolis, Santa Catarina, setembro de 1993.

[8] C. V. Albuquerque, M. D. Nunes, O. C. Duarte, "Performance Measurements in a Manufacturing Communication System", IEEE International Conference on Circuits and Systems ISCAS' 94, pp.3.69-3.72, Londres, Inglaterra, maio de 1994.

[9] L. F. Baginsky, O. C. Duarte, "Um Modelo de Implementação de Alto Desempenho para Protocolos de Comunicações Estruturados em Camadas", Revista da Sociedade Brasileira de Computação", vol. 6, pp. 46-56, abril de 1991.

[10] L. F. Baginsky, O. C. Duarte, "Um Modelo de Implementação de Alto Desempenho para Sistemas Abertos", IX Congresso da Sociedade Brasileira de Telecomunicações, São Paulo, pp. 19.3.1-19.3.5, setembro de 1991.

[11] M. D. Nunes, O. C. Duarte, "Análise de Mecanismos para Protocolos de Alto Desempenho", VI Simpósio Brasileiro de Arquitetura de Computadores e Computação de Alto Desempenho, Caxambú, Minas Gerais, agosto de 1994.

[12] M. Woo, N. U. Qazi, A. Ghafoor, "A Synchronization Framework for Communication of Pre-Orchestrated Multimedia Information", IEEE Network, pp.52-61, janeiro de 1994.

[13] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", ACM Trans. on Computer Systems, vol. 9, no. 2, maio 1991.

[14] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management", SIGCOMM Symposium, Communications Architecture & Protocols, pp. 8-18, Filadelphia Pensilvania, setembro de 1990.

[15] D.Verma, H. Zhang, D. Ferrari, "Delay Jitter Control for Real Time Communication in a Packet Switching Network", Proceedings of TriComm 91, pp. 35-46, Chapel Hill, abril de 1991.

[16] H. Zhang, S. Keshav, "Comparison of Rate-Based Service Disciplines", Proceedings of ACM SIGCOMM, pp. 113-121, setembro de 1991.

[17] G. Varghese, T. Lauck, “Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility”, in Proc. 11th ACM SIGOPS Symp. on Operating Systems Principles, pp. 25-38, Austin, TX, 1987.

[18] M.. Faerman, C. V. Albuquerque, O. C. Duarte, “Janela Móvel Particionada: Proposta de um Novo Mecanismo de Policiamento de Tráfego”, Relatório Técnico do Grupo de Teleinformática e Automação da COPPE/UFRJ, 1995.