# Capítulo V

# Implementação de Mecanismos para Garantia de Qualidade de Serviço

## V.1 Introdução

O objetivo deste capítulo é apresentar propostas de implementações de alguns dos mecanismos de controle de tráfego e de garantia de qualidade de serviço considerados neste trabalho. As propostas são baseadas na arquitetura de implementação descrita no capítulo IV. O conjunto de funcionalidades provido pela arquitetura facilitou sobremaneira a implementação dos mecanismos, conforme apresentado a seguir. Em especial, o Módulo de Gerenciamento de Temporizações teve um papel fundamental na implementação de todos os mecanismos. O procedimento de execução dos mecanismos é descrito através da forma de diagramas de blocos, com base na arquitetura de implementação utilizada. Por simplicidade de implementação dos mecanismos de policiamento foi utilizada uma lógica de evolução dos contadores de crédito inversa à descrita no Capítulo III. A complexidade das implementações, incluindo medidas do tempo de processamento dos mecanismos será analisada na seção V.13.

## V.2 Balde Furado

O mecanismo balde furado (*leaky-bucket*) 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 pode ser

incrementado até o limite máximo $C_{máx}$. A cada célula submetida ao mecanismo, se o crédito for positivo, este é decrementado e a célula é transmitida. Este procedimento está ilustrado no fluxograma da Figura V.1.

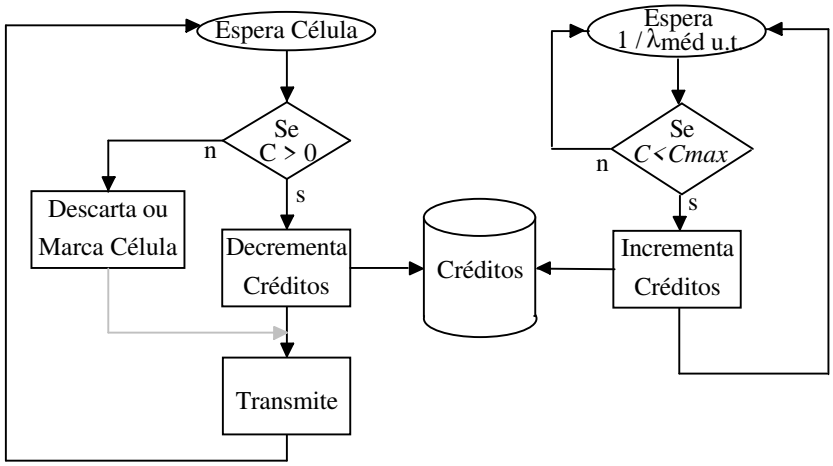



Figura V.1: Diagrama em blocos da implementação do balde furado.

Uma outra implementação deste mecanismo, proposta por Dittmann *et al* [DJM91], 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 possui as desvantagens de requerer marcas de tempo demasiadamente precisas e de executar operações de subtração não-unitárias. O pseudocódigo deste esquema é o seguinte:

A cada chegada de célula faça:

1. $\Delta t$ := 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 descartar célula;
   senao $C := C + 1$;

# V.3 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. Neste esquema (Figura V.2), 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.

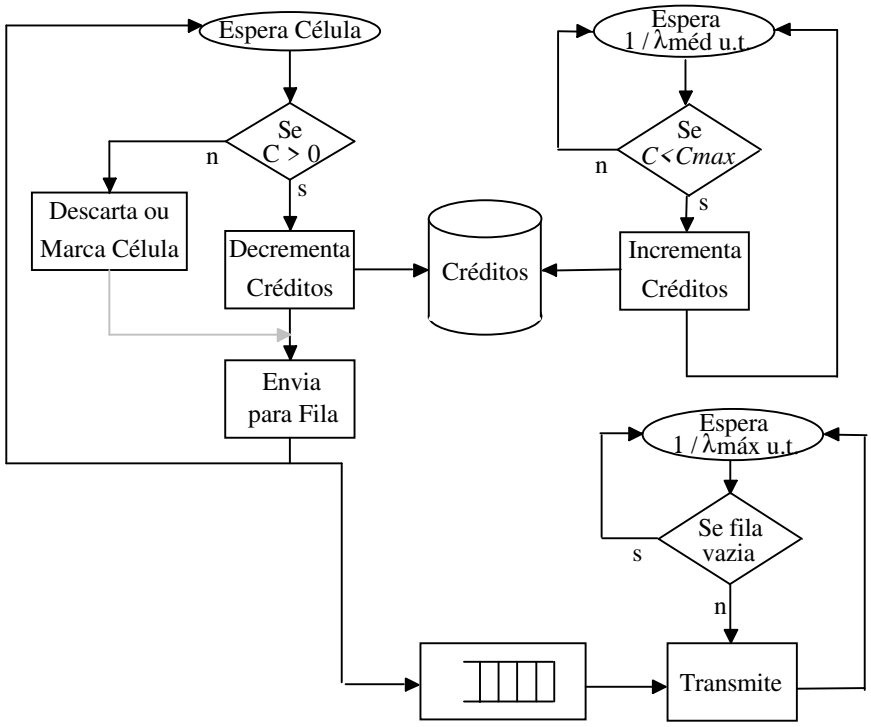



Figura V.2: Diagrama em blocos da implementação do balde furado com suavização do tráfego de saída.

## V.4 Janela Saltitante

O mecanismo de janela saltitante (*jumping window*) pode ser implementado por meio de um temporizador e um contador, de acordo com o seguinte procedimento. Ao concluir a fase de estabelecimento da conexão e de determinação dos parâmetros do mecanismo, o mecanismo janela saltitante 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 janela saltitante, 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 nesta implementação, optou-se por descartar a célula.

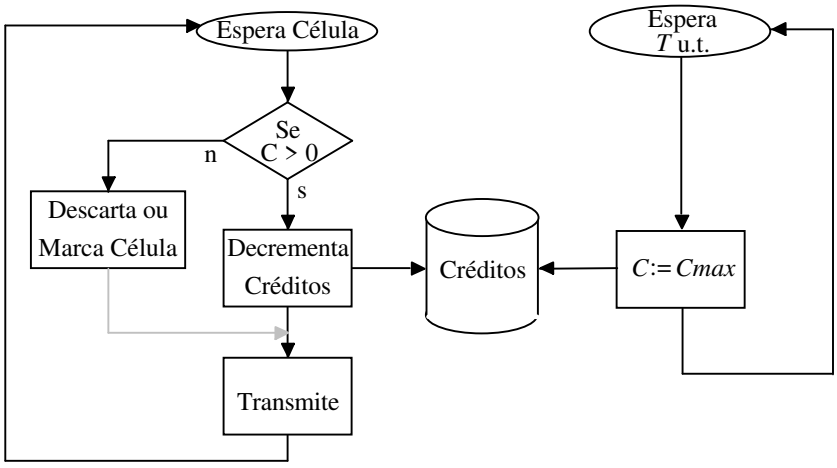



Figura V.3: Diagrama em blocos da implementação da janela saltitante.

## V.5 Janela Saltitante Sincronizada

O mecanismo janela saltitante sincronizada (*triggered jumping window*) 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.

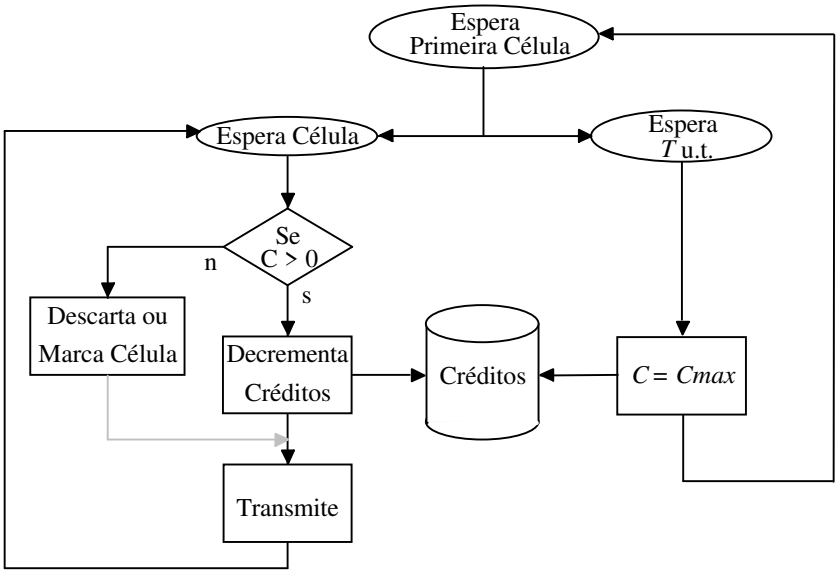



Figura V.4: Diagrama em blocos da implementação da janela saltitante sincronizada.

## V.6 Janela Deslizante Contínua

O mecanismo de janela deslizante contínua - JDC (*sliding window*) 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 igual a *T* unidades de tempo. No "estouro" de cada temporizador a variável correspondente ao 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.

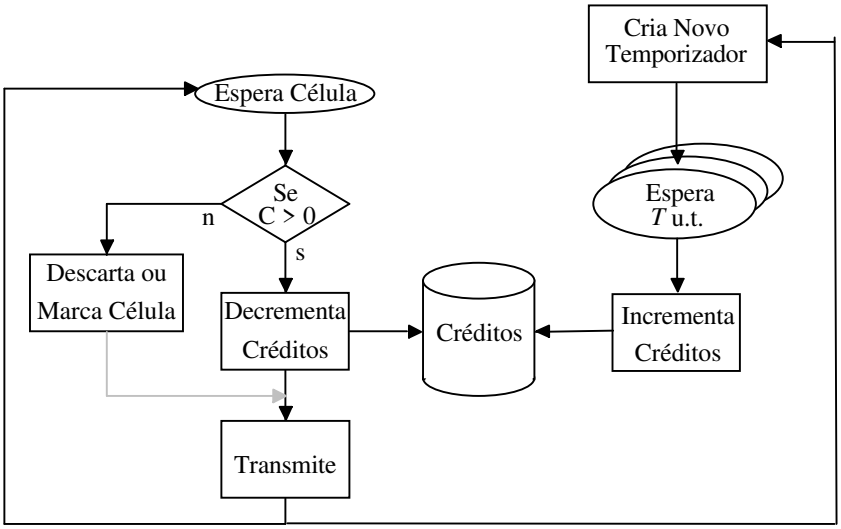



Figura V.5: Diagrama em blocos da implementação da janela deslizante contínua.

## V.7 Janela Deslizante Discretizada

O mecanismo janela deslizante discretizada - JDD (*discretized sliding window*) 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 (por exemplo, $t_{cel}$ = 2,7 μs 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 auto-disparável e um contador. Este esquema permite uma melhor utilização da memória e da CPU. O *bit* final do vetor de *bits* indica se chegou uma célula T unidades de tempo anteriores a atual. Este bit é somado ao número de créditos e deslocado a cada unidade de tempo $t_{cel}$. Cada célula não violadora entregue ao mecanismo seta o *bit* inicial do vetor e decrementa a variável crédito. Após T unidades de tempo este *bit* terá sido deslocado para a posição do *bit* final do vetor e será somado ao número de créditos.

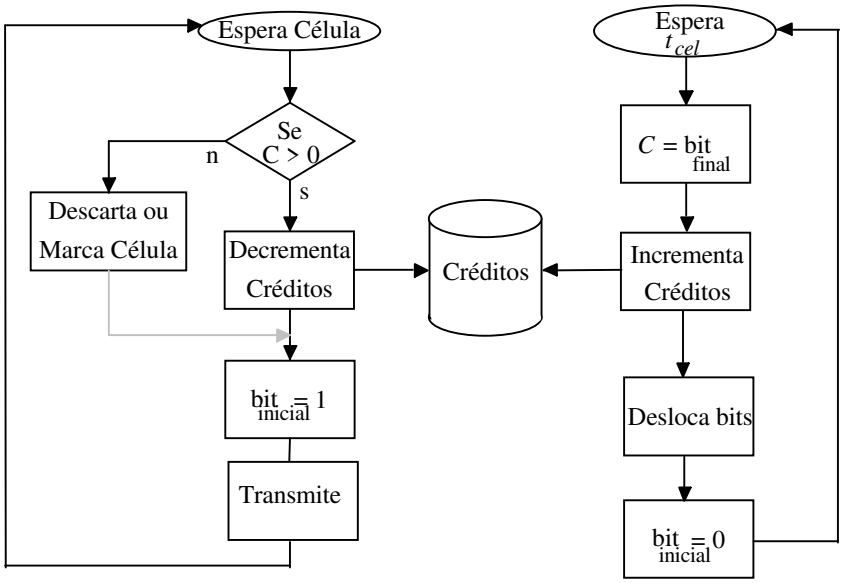



Figura V.6: Diagrama em blocos da implementação da janela deslizante discretizada.

## V.8 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 mecanismo janela deslizante discretizada. Quanto menor o número de partições, menor será a memória necessária, menor será a utilização da CPU e mais próximo estará do mecanismo janela saltitante.

O temporizador é responsável pela atualização do crédito a cada *T/P* unidades de tempo. Esta atualização corresponde ao incremento 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.

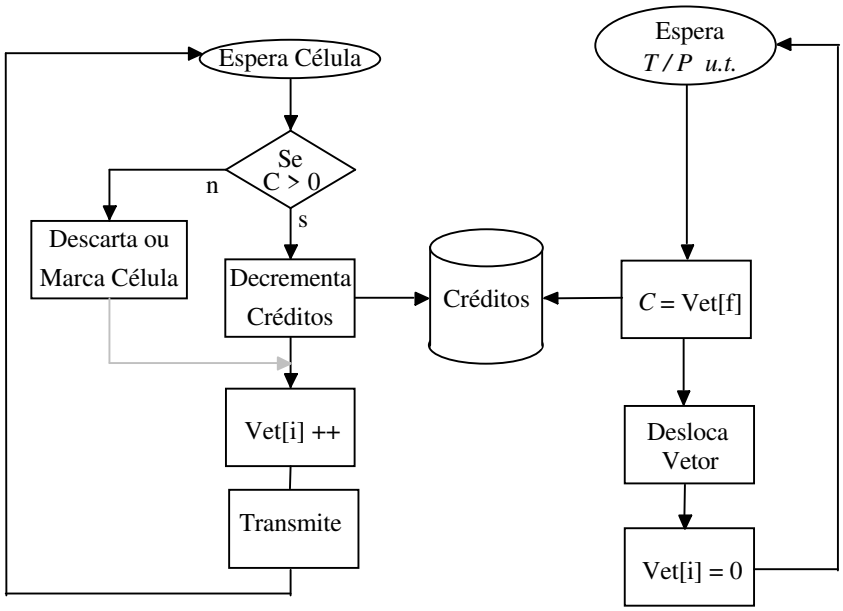



Figura V.7: Diagrama em blocos da implementação da janela móvel particionada.

## V.9 *Stop-and-Go*

O mecanismo *stop-and-go* pode ser implementado por meio de duas filas FIFO, de um contador de créditos por conexão e de um temporizador auto-disparável. São necessárias duas filas, uma para armazenar as células que serão transmitidas no próximo quadro de tempo (fila de espera) e outra para as células que serão transmitidas no quadro de tempo atual (fila de transmissão). O temporizador auto-disparável com período T é responsável por trocar as funções das duas filas. Quando este temporizador "estourar", a fila de transmissão estará obrigatoriamente vazia. A partir deste instante, esta fila passa a armazenar as células que estão chegando, enquanto que paralelamente são transmitidas as células que estavam armazenadas na outra fila. Observe que as filas possuem tamanho máximo $r$ limitado, determinado pela capacidade do canal de transmissão, de forma que possam ser transmitidas $r$ células em T unidades de tempo. O número máximo de células de uma conexão na fila de espera também é

limitado em $r_k$ células, conforme especificado na fase de estabelecimento da conexão. A cada célula da conexão $k$ submetida ao mecanismo, se o contador de créditos $C_k$ for positivo, este é decrementado e a célula é armazenada na fila de espera. Se o contador for nulo, optou-se por descartar a célula. No instante do "estouro" do temporizador, o contador de créditos $C_k$ de todas as conexões ativas são reinicializados com os respectivos valores de $r_k$. Este procedimento está ilustrado no fluxograma da Figura V.8.

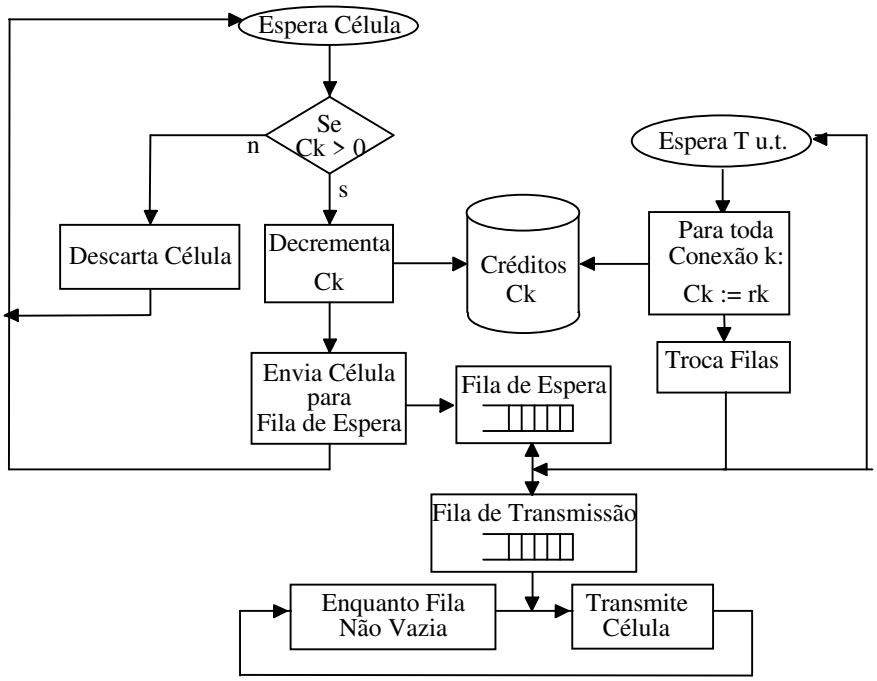



Figura V.8: Diagrama em blocos da implementação do *stop-and-go*.

## V.10 *Virtual Clock*

Para se implementar o mecanismo *virtual clock* utilizou-se uma fila indexada, de forma que se possa inserir elementos em qualquer posição da fila de acordo com o seu índice. Além desta fila, são necessários três contadores ($VC_i$, $auxVC_i$ e $AIR_i$) e uma constante ($Vtick_i$) por conexão. Este mecanismo não necessita de um temporizador especial, necessita apenas do valor do contador de tempo RT do módulo de gerenciamento de temporizações. O procedimento da implementação do mecanismo segue o próprio procedimento do algoritmo

que o descreve. Como está sendo considerado constante o tamanho dos pacotes (células de tamanho 53 octetos), o fator de atualização das variáveis, $Vtick_i$, é uma constante determinada no estabelecimento da conexão. A variável $auxVC_i$ é o índice de ordenação da fila de transmissão, enquanto que a variável $VC_i$ monitora a taxa média de transmissão. O procedimento de excução do *virtual clock* é apresentado a seguir.

- Ao se estabelecer a conexão *i*,
    1. $Vtick_i \leftarrow 1/AR_i$ (células/seg);
- Ao receber o primeiro pacote da conexão i,
    1. $VC_i \leftarrow RT$;
    2. $auxVC_i \leftarrow RT$;
    3. $AIR_i \leftarrow 1$;
- Ao receber os demais pacotes do fluxo i,
    1. $auxVC_i \leftarrow \max (RT, auxVC)$;
    2. $VC_i \leftarrow (VC_i + Vtick_i)$;
       $auxVC_i \leftarrow (auxVC_i + Vtick_i)$;
    3. Estampar a célula com o valor da variável $auxVC_i$ e enviá-la para a fila de transmissão.
    4. Incrementar AIR*i*;
- Se a fila de transmissão estiver cheia, descartar o último elemento da fila.
- Transmitir as células ordem crescente dos valores das estampas.
- Após receber um conjunto de $AIR_i$ (= $AR_i$ x $AI_i$) células da conexão *i*,
    1. Se ($VC_i$ - RT) > Tmax, interromper a conexão;
    2. Se ($VC_i$ < RT) então $VC_i \leftarrow RT$ e $auxVC_i \leftarrow RT$;

## V.11 *Hierarchical Round Robin*

O mecanismo *hierarchical round robin* pode ser implementado por meio uma fila FIFO por conexão, uma fila de identificadores de conexão (FID - Flow IDentifier). Esta fila só pode

conter um FID relativo a cada conexão. Além disso, são necessários contadores de créditos por conexão. O limite destes contadores é variável conforme o nível de prioridade da conexão. O procedimento de execução deste mecanismo é descrito pelo pseudocódigo a seguir.

- A cada chegada de célula,
    1. se o FID desta célula não estiver na fila de FIDs,
        inserir FID na fila de FIDs;
    2. armazene a célula na respectiva fila da conexão FID;

Paralelamente,

- para cada FID da fila de FIDs, (passo 0)
    1. transmitir célula da fila da conexão FID;
    2. se fila vazia
        servir o próximo elemento da fila de FIDs; (voltar ao passo 0)
    3. decrementar contador FID;
    4. se contador FID for nulo
        inserir FID na fila de FIDs;
        servir o próximo elemento da fila de FIDs; (voltar ao passo 0)
    5. servir o próximo elemento da fila da conexão FID; (voltar ao passo 1)

## V.12 *Jiiter Earliest-Due-Date*

O mecanismo *jitter-EDD* é composto de um regulador e de um escalonador de pacotes. O regulador possui a função de eliminar o *jitter*, atrasando o pacote de ET unidades de tempo. ET é o tempo de aptidão, de forma que após decorrido ET u.t. o pacote está apto a ser transmitido. O pacote é portanto colocado em uma fila de espera. O escalonador de pacotes é responsável por transmitir os pacotes segundo a disciplina de serviço *class-EDD*. O cálculo do tempo de espera ET é dependente do tempo de transmissão do pacote pelo nó anterior e portanto cada pacote $k$ possui um tempo de espera $ET_k$. Este mecanismo foi implementado

por Hui Zhang [Zha93] através de múltiplos temporizadores, um por pacote, para realizar o atraso de ET u.t. de cada pacote.

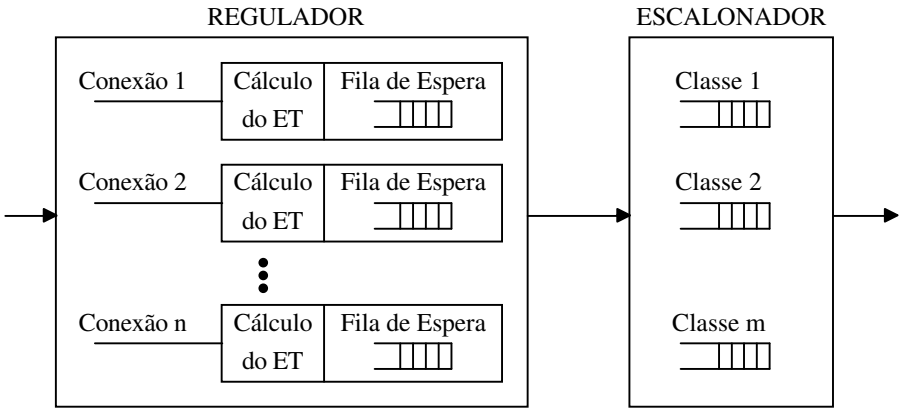



Figura V.9: Diagrama em blocos do mecanismo *jitter-EDD*.

Nesta seção, uma outra forma mais eficiente de implementar o mecanismo *jitter-EDD* é proposta. Alterando-se a forma de se calcular o ET, através da utilização da variável TRH - tempo restante para habilitação dos demais pacotes da fila de espera, é possível obter uma implementação com um único temporizador por conexão. Isto representa uma grande vantagem em termos de memória e de CPU utilizadas. Para se realizar o tempo de espera de cada pacote no regulador, o novo cálculo do tempo ET é feito, de forma que no tempo de chegada de cada pacote, após o cálculo original do tempo $ET_k$ do pacote $k$, decrementa-se desta variável o tempo restante para habilitação dos demais pacotes da fila de espera (TRH). O TRH é iniciado com zero, a cada chegada de pacote é incrementado do $ET_k$ e a cada unidade de tempo deve ser decrementado de 1 unidade. No estouro do temporizador que calcula o tempo de espera do primeiro pacote da fila de espera, este pacote é retirado da fila e enviado para o escalonador de pacotes. Um novo temporizador, de duração do ET do próximo elemento da fila de espera, é então iniciado.

Para implementar o mecanismo *jitter-EDD* foram necessários para cada conexão um temporizador para o tempo de espera, um temporizador para decrementar o TRH, um

contador para armazenar o TRH e uma fila FIFO de espera. Além disso, necessita-se de uma ou mais filas de transmissão multiplexadas por todas as conexões, servidas pelo escalonador de pacotes. A Figura V.9 ilustra o diagrama em blocos do mecanismo e a Figura V.10 o procedimento de execução do regulador do mesmo. Nesta figura, CT é o termo de correção enviado no cabeçalho do pacote pelo nó anterior.

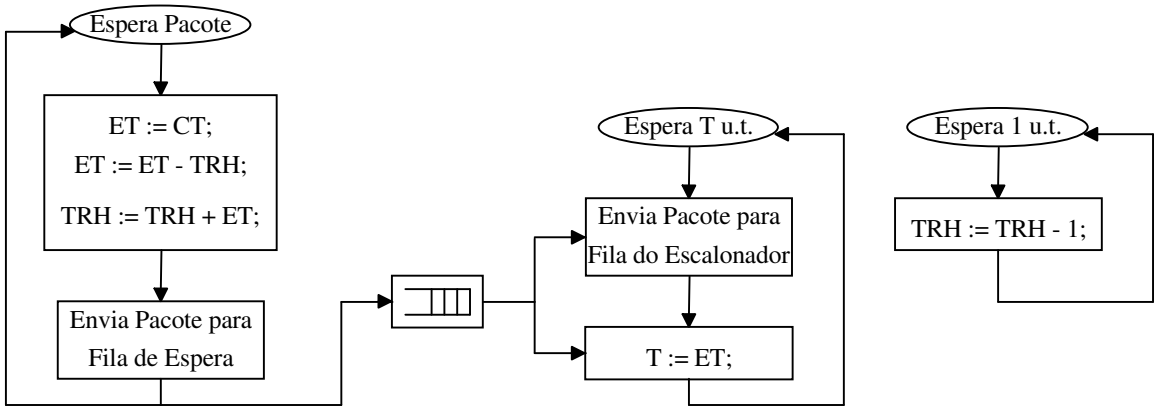
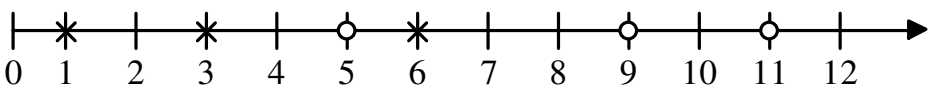



Figura V.10: Procedimento de execução do regulador do mecanismo *jitter-EDD*.

A Figura V.11 ilustra a evolução no tempo das variáveis TRH e $ET_k$, exemplificando o funcionamento do mecanismo. Nesta figura, x ilustra os tempos de chegada dos pacotes, e o os respectivos tempos a partir do qual estão aptos para serem transmitidos.




| 1 | 3 | 5 | 6 | 9 | 11 |
|---|---|---|---|---|---|
| TRH = 0; | TRH = 2; | TRH = 4; | TRH = 3; | TRH = 2; | TRH = 0; |
| $ET_1$ = 5; | $ET_2$ = 6; | Novo | $ET_3$ = 5; | Novo | |
| TRH = 5; | $ET_2$ = 4; | Temporizador: | $ET_3$ = 2; | Temporizador: | |
| T = 5; | TRH = 6; | T = $ET_2$ = 4; | TRH = 5; | T = $ET_3$ = 2; | |

Figura V.11: Evolução no tempo das variáveis TRH e $ET_k$.

## V.13 Resultados

Esta seção visa apresentar dados obtidos a partir da implementação em ambientes específicos dos mecanismos de controle de tráfego e policiamento balde furado, balde furado com limitação da taxa máxima de saída e mecanismos baseados em janela, enfocando a sua complexidade de implementação. São analisados a quantidade de memória necessária, o número de temporizadores, a precisão exigida para os temporizadores, o tempo médio de processamento total de cada célula (incluindo todo o processamento do protocolo do sistema de comunicação: alocação de estruturas de primitivas, preenchimento de cabeçalhos, retiradas e inserções em filas, etc.) e o tempo médio de processamento específico do mecanismo (onde são computados apenas os procedimentos referentes ao mecanismo: criação de temporizadores, testes lógicos, atualização de variáveis, etc.) para cada célula. Para o caso específico dos mecanismos baseados em janela, foi analisada a probabilidade de violação, verificando-se o comportamento do mecanismo janela móvel particionada, com relação aos mecanismos janela saltitante e janela deslizante.

A Tabela V.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, total 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 1.000 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 na Tabela V.1 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 janela saltitante e ao janela deslizante discretizada, da ordem de 176 μs e 150 μs.

Tabela V.1: Comparação entre os mecanismos de policiamento.

| 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 o número de temporizadores necessários para a implementação do mecanismo. Observa-se que o mecanismo janela deslizante contínua necessita do maior valor: $C$ temporizadores Deve-se ressaltar que quanto maior o número de temporizadores, maior será a utilização da CPU. O número de temporizadores do mecanismo balde furado com suavização do tráfego de saída não pode ser comparado com os demais, visto que este equivale a dois mecanismos de policiamento em cascata, necessitando portanto do dobro de temporizadores.

Quanto à precisão requerida pelos temporizadores, os temporizadores do mecanismo janela deslizante contínua exigem uma precisão altíssima, podendo ser disparados na chegada

de cada célula, e portanto, em qualquer instante de tempo. 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 um temporizador da ordem de $T$ unidades de tempo, e portanto sua precisão, em média, não será crítica. Uma das grandes contribuições do mecanismo janela móvel particionada é permitir a utilização de temporizadores com toda a faixa de precisão, desde a utilizada na janela saltitante até a utilizada na janela deslizante contínua, conforme o número de partições utilizado.

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

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

para memorização do tráfego. As demais regiões de memória estão ilustradas na Tabela V.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 $\{\text{P}\}$ *bits* para armazenar o número de partições. As duas variações da janela deslizante são comparáveis em relação a quantidade de memória necessária. A janela saltitante é a que necessita da menor quantidade de memória. Já a janela móvel particionada necessita de uma quantidade de memória variável de acordo com o número de partições, porém sempre inferior a quantidade necessária para a janela deslizante. Por exemplo, o caso da janela deslizante discretizada com $T = 60 \times t_{cel}$, onde $t_{cel}$ é o tempo necessário para a transmissão de uma célula, e $C = 30$. Neste caso ela necessita de 60 *bits* para memorizar o tráfego de uma janela. Se a janela móvel particionada com 20 partições for utilizada, em cada partição poderão chegar no máximo 3 células, representadas por 2 *bits*. Portanto o número total de *bits* para representar o vetor FIFO será de

um total de 40 bits. Para um JMP com 10 partições, 30 bits são necessários, e com 5 partições, 20 *bits*.

Teoricamente, o comportamento do mecanismo janela móvel particionada é, em termos de complexidade de implementação e de probabilidade de violação ($p_{viol}$), intermediário aos dos mecanismos janela saltitante e deslizante. Além da computação da $p_{viol}$ obtida na implementação, resultados de simulação foram analisados e comparados com a implementação. A descrição dos modelos de simulação estão descritas em [FAD95a, FAD95b]. Para a simulação dos mecanismos fez-se uso do software *Queueing Networks Analysis Package* - QNAP, o qual é baseado em redes de filas, utilizando um intervalo de confiança de 0.95.

Para a computação da probabilidade de violação dos mecanismos, tanto na simulação quanto na implementação, utilizou-se dois padrões de tráfego: um tráfego geométrico onde a média do intervalo entre as chegadas de células é igual a cinco unidades de tempo e um tráfego rajada/silêncio. Neste último, o tamanho da rajada obedeceu a uma distribuição geométrica com média 10, onde a taxa de transmissão da rajada é igual a 1 célula/unidade de tempo e o período de silêncio seguiu uma distribuição geométrica com média 40 unidades de tempo. Portanto, este tráfego apresenta nível de rajada igual a 5 e, como o tráfego anterior, taxa média igual a 1/5 células/unidade de tempo. Para todos os mecanismos foram arbitrados os parâmetros $T = 50$ e $C = 10$, representando uma taxa média policiada proporcional a 1/5 células/unidade de tempo.

A Figura V.12 mostra as probabilidades de violação do mecanismo janela móvel particionada em função do número de partições computadas, quando o tráfego utilizado foi o geométrico, e a Figura V.13, o rajada/silêncio. Os limites inferiores e superiores dos gráficos são as $p_{viol}$ obtidas para os mecanismos janela saltitante e deslizante, respectivamente.

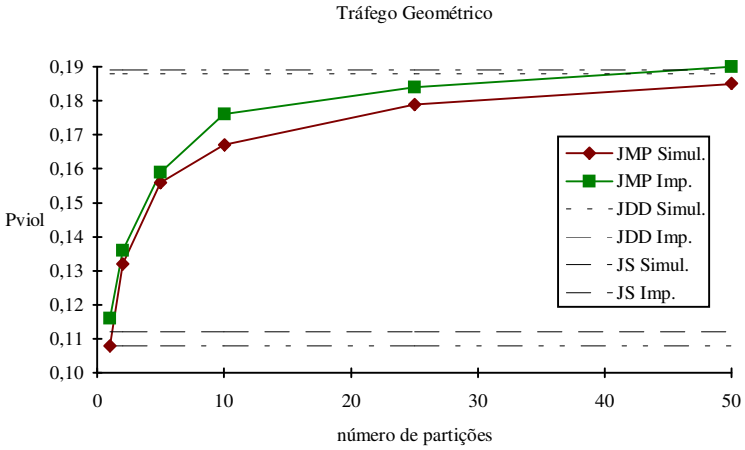



Figura V.12: Probabilidades de violação do mecanismo JMP em função do número de partições para o tráfego geométrico.

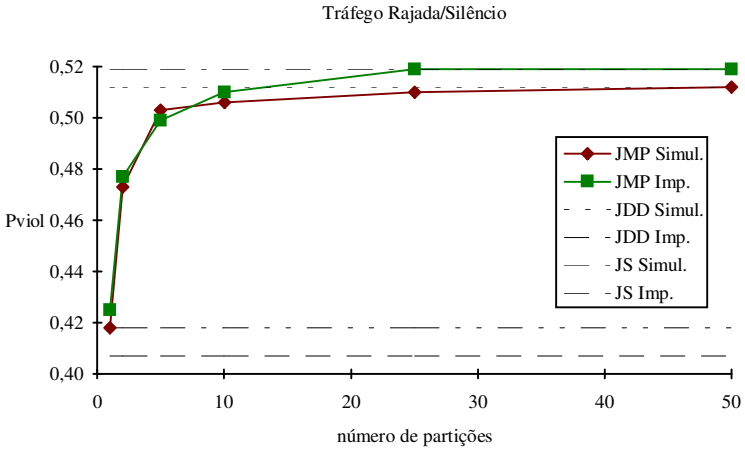



Figura V.13: Probabilidades de violação do mecanismo JMP em função do número de partições para o tráfego rajada/silêncio.

Em ambos os gráficos é possível constatar a semelhança entre os resultados obtidos pela simulação e pela implementação, sendo os resultados da simulação mais precisos que os da implementação, pois foram simuladas cem vezes mais amostras de tráfego que a quantidade utilizada na implementação. Pode-se observar que para o número de partições *P* igual a um, a probabilidade de violação do mecanismo janela móvel particionada é a mesma do mecanismo janela saltitante, e a medida que o número de partições aumenta, a probabilidade de violação

aproxima-se rapidamente da probabilidade de violação do mecanismo janela deslizante. No caso do tráfego rajada/silêncio a $p_{viol}$ aproxima-se mais rapidamente, ou seja, a partir de um número menor de partições, de seu valor final ($p_{viol}$ da janela deslizante) que no caso do tráfego geométrico. No entanto as probabilidades de violação no caso do tráfego rajada/silêncio para todos os mecanismos é bastante superior às $p_{viol}$ obtidas com o tráfego geométrico.

Os resultados de simulação e de implementação obtidos para padrões de tráfego geométrico e rajada/silêncio comprovam que mesmo com um número pequeno de partições ($P = 5$) obtém-se uma funcionalidade bastante próxima da janela deslizante, com uma baixa complexidade de implementação e uma economia considerável de memória e processamento.