Capítulo III

Mecanismos para Garantia de Qualidade de Serviço

III.1 Introdução

Nos últimos anos muito se caminhou no desenvolvimento da RDSI-FL, porém vários problemas restam em aberto a procura de uma solução satisfatória. O controle de tráfego é sem dúvida um dos problemas mais complexos a ser resolvido.

A fim de garantir um bom desempenho da RDSI-FL é necessário um conjunto de mecanismos de controle de tráfego. Vários controles são previstos [ITU91] tais como: controle de admissão de conexão, controle de prioridade, controle de congestionamento e controle de parâmetros de utilização/rede.

O controle de parâmetros de utilização/rede se 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.

Uma outra questão é relativa a complexa interação entre os fluxos, que comumente resulta em mudanças no padrão do tráfego de alguns fluxos ao longo dos nós da rede [Kur93]. Para se obter uma maior eficiência da RDSI-FL, 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.

Mesmo que o usuário da rede emita um fluxo "bem comportado" respeitando o "contrato" estabelecido na conexão, o simples ato de multiplexar um fluxo com os demais fluxos pode aumentar a vazão além da taxa máxima de transmissão. A Figura III.1 ilustra a mudança na taxa de transmissão do fluxo *C* resultante da multiplexação no nó N.

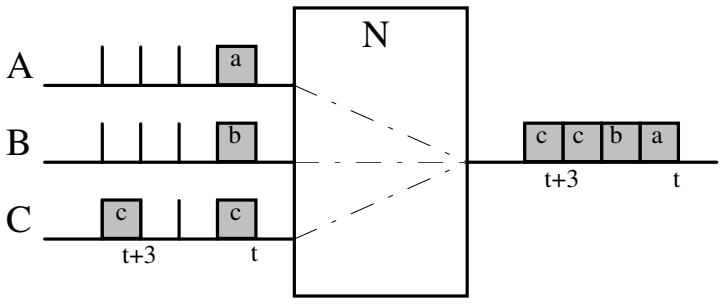


Figura III.1: Mudança na taxa de transmissão devido à multiplexação.

Neste exemplo fica claro que para prover garantias de QOS, os mecanismos devem, entre outras coisas, ser capazes de prover disciplinas de multiplexação que evitam tais mudanças nos parâmetros do tráfego, ou ser capazes de caracterizar tal variação visando uma futura eliminação das variações (*jitters*) de vazão e de atraso.

A RDSI-FL deve atender de forma eficiente ao maior número possível de usuários e garantir a veiculação do tráfego contratado. Parâmetros de qualidade de serviço, como vazão,

taxa de perda de células, atraso, variação do atraso, etc., devem portanto ser especificados, monitorados e garantidos pela rede.

No que concerne ao controle de tráfego, a maioria das estratégias utilizadas nas redes de dados convencionais são baseadas em reconhecimentos (*acknowledgement-based*) emitidos pelo destinatário. Esta abordagem não é apropriada para as redes de alta velocidade, uma vez que uma grande quantidade de informação já terá sido transmitida no momento que chegar o reconhecimento. O controle de tráfego orientado à vazão (*rate-based*) surge como alternativa a este problema. Além disso, as futuras redes de serviços integrados devem suportar os mais diversos tráfegos, dentre os quais os tráfegos de tempo real, que possibilitam às aplicações uma comunicação com garantias de desempenho severas. Estas garantias são dificilmente supridas pela tradicional disciplina *First-Come-First-Served* (FCFS) largamente utilizada nas redes convencionais. Para atender a este tipo de tráfego a utilização de disciplinas de serviço com prioridades é mais adequada.

Este capítulo apresenta, analisa e compara diversos mecanismos de controle de tráfego para garantia da qualidade de serviço orientados a controle de taxa de transmissão. A seção III.2 apresenta os principais mecanismos de policiamento citados na literatura e propõe o mecanismo baseado em janela denominado janela móvel particionada [FAD95a, FAD95b]. A seção III.3 apresenta um conjunto de mecanismos para garantia de QOS que possuem uma disciplina de serviço própria e cuja funcionalidade pode prover, além de policiamento, limites para o atraso (e *jitter*) fim-a-fim e eliminação de *jitter* de uma forma distribuída. Os mecanismos apresentados nesta seção são *Virtual Clock* [Zha91], *Fair Queueing* [Kes90a, Kes91a], *Stop-and-Go* [Gol90], *Hierarchical Round Robin* [Kes90b], *Delay-Earliest-Due-Date* [Fer90b], *Jitter-Earliest-Due-Date* [Fer91] e *Rate-Controlled Static-Priority* [Fer93].

III.2 Mecanismos de Policiamento[1]

Dado que uma chamada é admitida, a rede se protege de fontes abusivas através de um controle de parâmetros, denominado policiamento, garantindo que a fonte de tráfego não violará os parâmetros contratados, tais 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. Este trabalho considera o tráfego submetido a rede como sendo o tráfego não violador.

Vários mecanismos de policiamento têm sido propostos. Entre os mais citados na literatura tem-se o balde furado (*leaky bucket*) [BCT91, Rat91] e suas extensões [Ber90], o contador de pico [JMo94] e os mecanismos baseados em janela. Entre estes últimos destacam-se a janela saltitante, janela deslizante ou retangular, janela saltitante sincronizada, janela triangular e média móvel exponencialmente ponderada [DJM91, JMo94, Rat91].

Os mecanismos baseados em janela consistem em limitar, em um determinado valor C (crédito máximo), o número de células injetadas na rede durante um intervalo de tempo T. Este intervalo de tempo é denominado janela. Os mecanismos baseados em janela diferem entre si segundo a maneira pela qual as janelas evoluem ao longo do tempo e pela forma como o valor dos créditos é considerado.

---

[1] Parte do texto e figuras desta seção foi extraída do artigo [FAD95a].

III.2.1 Balde Furado

Um dos mecanismos para controle de taxa de transmissão e policiamento mais citados na literatura é o conhecido como Balde Furado - BF (*Leaky Bucket* - LB). Este é um algoritmo proposto para ser implementado na controladora de comunicação das redes de alta velocidade.

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 III.2). 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 da rajada máxima 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.

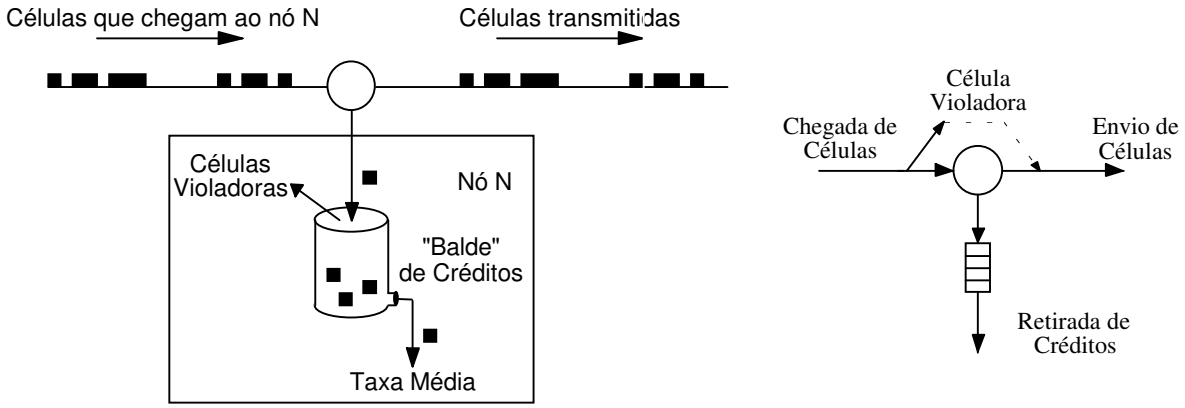


Figura III.2: Representações do mecanismo balde furado.

Várias versões deste algoritmo têm sido propostas, sendo que uma das mais citadas é a descrita em [Par92], que utiliza como parâmetros de QOS a taxa de transmissão média, a taxa de transmissão máxima e o tamanho da rajada máxima. Nesta versão (Figura III.3) cada nó

possui dois depósitos imaginários (baldes ou *buckets*) para cada fluxo: um depósito de créditos e um depósito de transmissão.

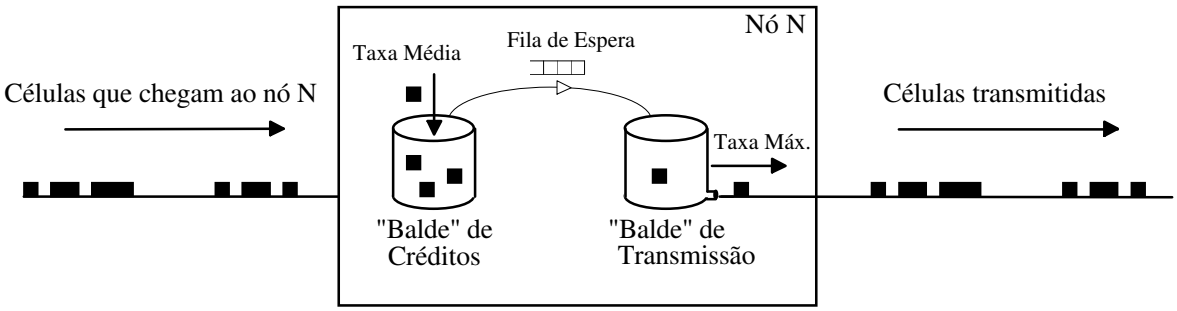


Figura III.3: Versão do mecanismo balde furado descrita em [Par92].

O depósito de créditos é preenchido à taxa de transmissão média e o número máximo de créditos (tamanho da rajada) limita quantos créditos um fluxo pode armazenar. Quando o depósito de créditos está cheio, os novos créditos depositados são descartados. Para enviar um pacote, o transmissor deve retirar do depósito de créditos um número de créditos igual ao tamanho do pacote. Se não houver créditos suficientes o transmissor deve esperar na fila até que uma quantidade suficiente de créditos esteja armazenada no depósito. Quando o pacote for enviado, um número de créditos igual ao tamanho do pacote é colocado num depósito vazio (depósito de transmissão) que drena créditos à taxa de transmissão máxima. Nenhum pacote poderá ser enviado até que este depósito esteja novamente vazio. Isto garante que a taxa de transmissão nunca será superior à taxa máxima negociada.

Este algoritmo é bastante eficiente como um mecanismo de controle de fluxo, no entanto não define uma disciplina de serviço, ou seja, não considera a interação entre fluxos, não controlando a ordem de transmissão de pacotes de diferentes fluxos nem discriminando os pacotes de acordo com seus diferentes parâmetros de QOS.

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 III.4(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.

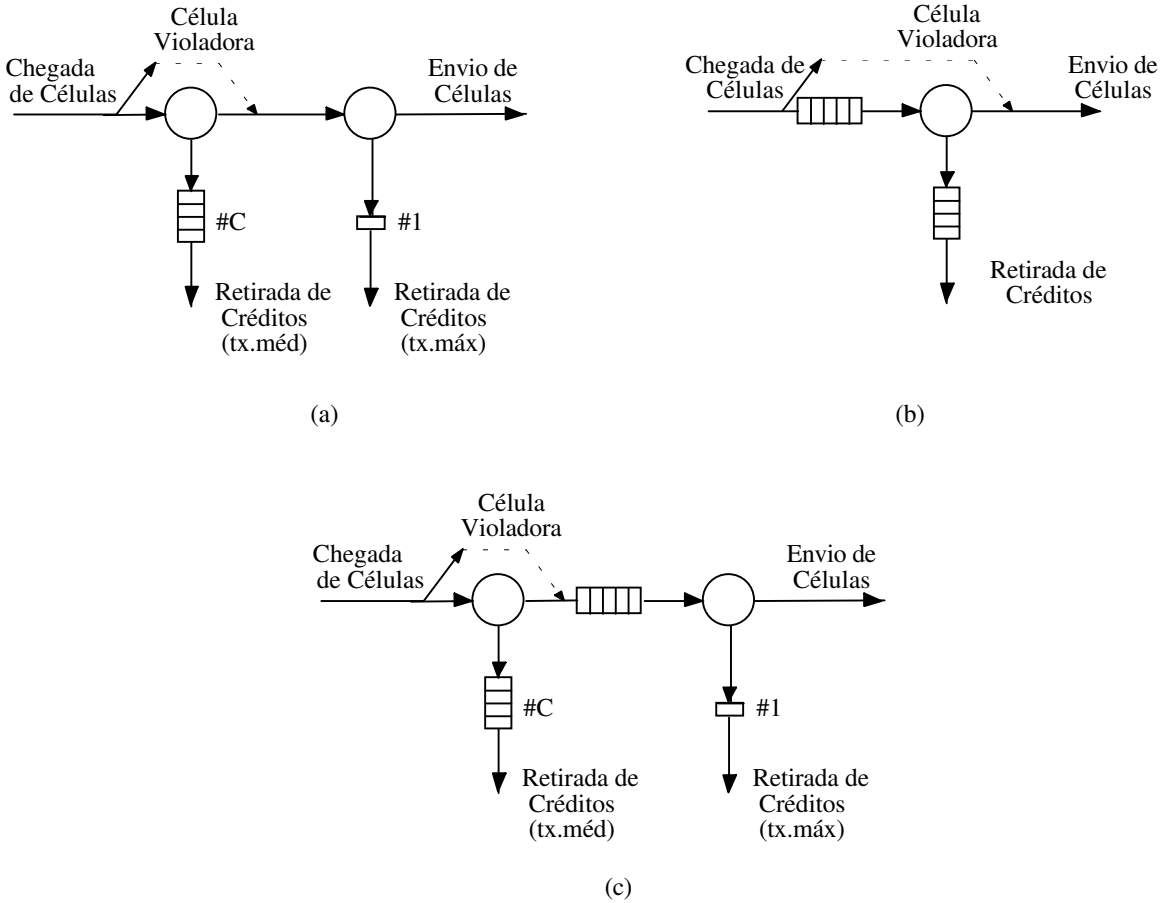


Figura III.4: Variações do mecanismo balde furado.

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 III.4(b) apresenta o BF com suavização do tráfego na entrada e a Figura III.4(c) com suavização na saída. As células consideradas violadoras são armazenadas até que possam ser transmitidas sem penalização.

III.2.2 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.

Este mecanismo é bastante simples de ser implementado através de um temporizador e de um contador. As janelas justapostas deste mecanismo são obtidas pelo temporizador na sua configuração auto-disparável, ou seja, que torna a disparar no momento que estoura. O contador deve ser inicializado com o valor *C* do crédito máximo e decrementado de uma unidade a cada célula que é injetada na rede (Figura III.5).

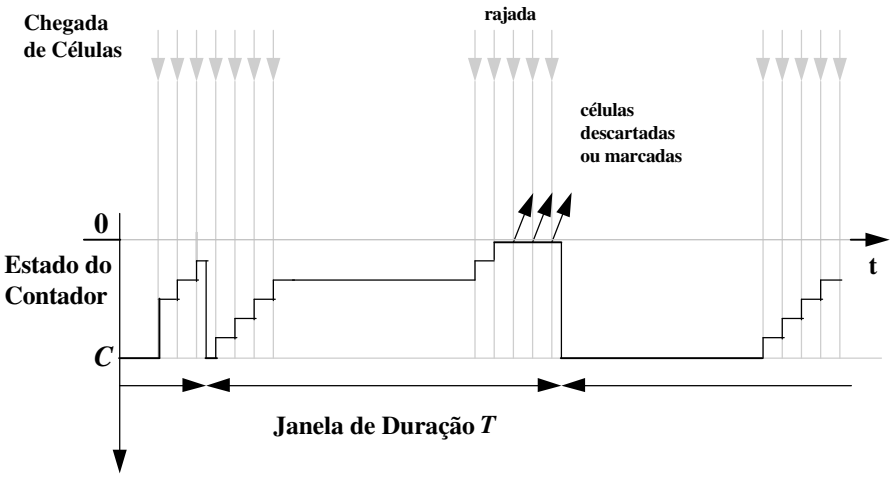


Figura III.5: O mecanismo janela saltitante.

Um padrão de tráfego bastante utilizado na análise dos mecanismos de policiamento é o denominado tráfego de pior caso. Define-se como tráfego de pior caso, o tráfego que passa pelo mecanismo de policiamento, explorando totalmente a taxa policiada sem sofrer qualquer tipo de penalização e que possui características que dificultem sua veiculação pela rede. A condição mais severa para a rede é durante as rajadas pois dependendo do estado da rede as células poderão ser armazenadas (provocando atrasos) e até mesmo perdidas por transbordo de memória. Assim, pode-se concluir que o tráfego de pior caso é do tipo rajada/silêncio onde o comprimento da rajada é o maior possível.

Se a fonte estiver sincronizada com a janela do mecanismo janela saltitante, ela pode gerar duas rajadas concatenadas, de C células, ao fim de um intervalo e no início do intervalo seguinte. Um período de silêncio de $2T$ - ($2C \times t_{cel}$) será necessário, onde t_{cel} denota o intervalo entre chegadas de células, durante a rajada. Portanto, o ciclo de ocupação para o mecanismo janela saltitante tem duração de $2T$ (Figura III.6).

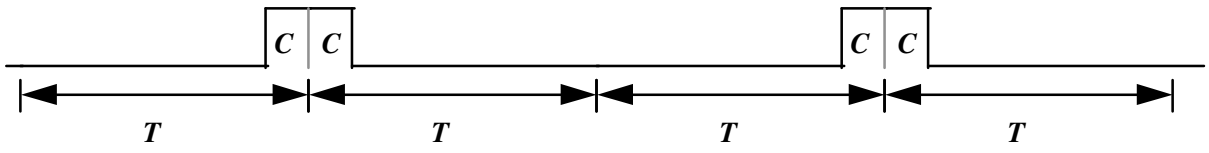


Figura III.6: Tráfego de pior caso para o mecanismo janela saltitante.

III.2.3 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 (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.

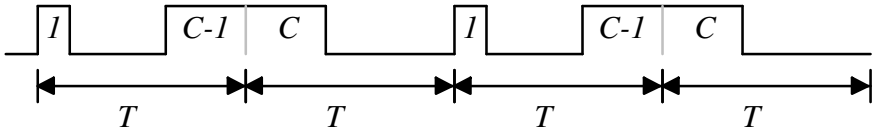


Figura III.7: Tráfego de pior caso para o mecanismo janela saltitante sincronizada.

Neste mecanismo o tráfego de pior caso permite a transmissão de uma rajada de 2*C* - 1 células e o seu ciclo de ocupação tem duração de 2*T* conforme ilustrado na Figura III.7.

III.2.4 Janela Deslizante

Dentre os mecanismos de policiamento, o que apresenta concomitantemente vantagens referentes à memorização do tráfego da fonte, ao controle da taxa média, além da limitação do comprimento da rajada, é a janela deslizante. Porém este mecanismo tem sido alvo de críticas referentes à sua complexidade de implementação [Rat91, JMo94]. Já Dittmann *et al* [DJM91] apresenta uma proposta de implementação para a janela deslizante, onde defende um nível de complexidade não tão crítico.

III.2.4.1 Janela Deslizante 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*. A Figura III.8 ilustra o comportamento deste mecanismo. A superposição das janelas produz um efeito de "memorização" do tráfego.

Uma outra forma de vizualizar este mecanismo é ao invés de se considerar várias janelas, considerar uma única janela que desliza ao longo do tempo memorizando os instantes de chegada das células. Esta memorização é realizada através de um contador que é decrementado (memoriza) no instante da chegada de uma célula e incrementado (apaga da memória) *T* unidades de tempo após.

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.

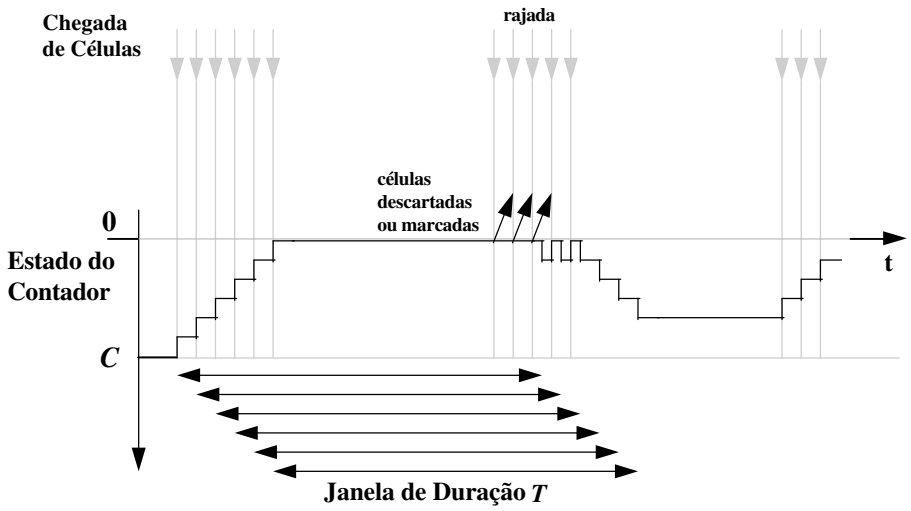


Figura III.8: O mecanismo janela deslizante contínua.

Pelo comportamento deste mecanismo, o número de células submetido a rede durante qualquer intervalo de tempo de duração *T* está limitado ao crédito máximo *C*. Portanto, o mecanismo janela deslizante apresenta a vantagem de limitar a taxa policiada do tráfego durante qualquer intervalo de duração *T* ao valor *C*/*T*. Neste mecanismo o tráfego de pior caso é dado por uma rajada de comprimento *C*, seguida por um período de silêncio cuja duração é $T - (C \times t_{cel})$. como ilustrado na Figura III.9.

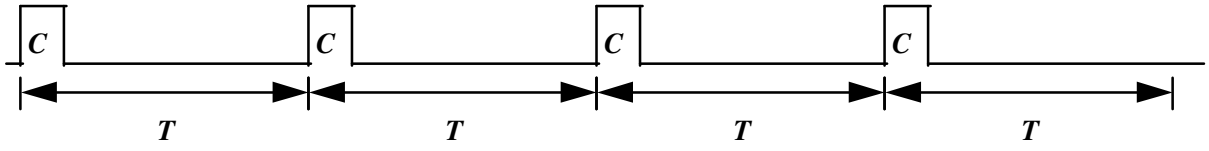


Figura III.9: Tráfego de pior caso para o mecanismo janela deslizante.

III.2.4.2 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. Este mecanismo surgiu como uma alternativa para diminuir a complexidade de implementação relativa a memorização do tráfego. Uma das vantagens deste esquema é relativa ao número de temporizadores necessários a sua implementação, que será explicada no capítulo V. 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.

III.2.5 Janela Móvel Particionada

Este seção propõe e analisa o mecanismo de janela móvel particionada [FAD95a, FAD95b], um mecanismo de policiamento baseado em janela que requer uma complexidade de implementação inferior ao mecanismo de janela deslizante e que é tão restritivo quanto este em relação à duração das rajadas. Este mecanismo também apresenta a vantagem de ser extremamente flexível. De acordo com as exigências do contrato entre usuário e rede, a configuração da janela móvel particionada mapeia seu algoritmo desde a janela saltitante até a janela deslizante.

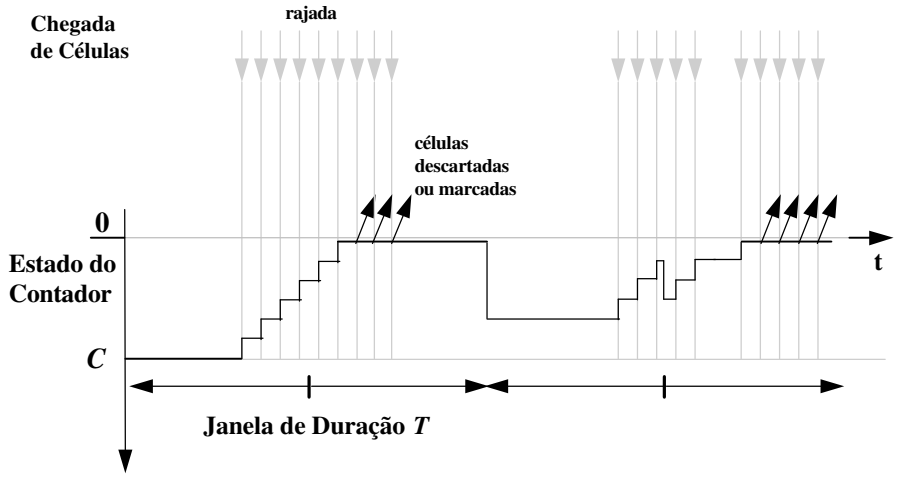
O mecansimo Janela Móvel Particionada - JMP busca um compromisso entre a simplicidade da janela saltitante e a eficiência da janela deslizante em relação a memorização do tráfego, policiamento acurado da taxa média e restrição a comprimento de rajadas.

Um dos principais fatores que leva o mecanismo de janela deslizante a demandar uma grande complexidade de implementação é o armazenamento do instante da chegada de uma célula. Por outro lado, as janelas justapostas do mecanismo de janela saltitante podem ser simplesmente implementadas por um temporizador auto-disparável.

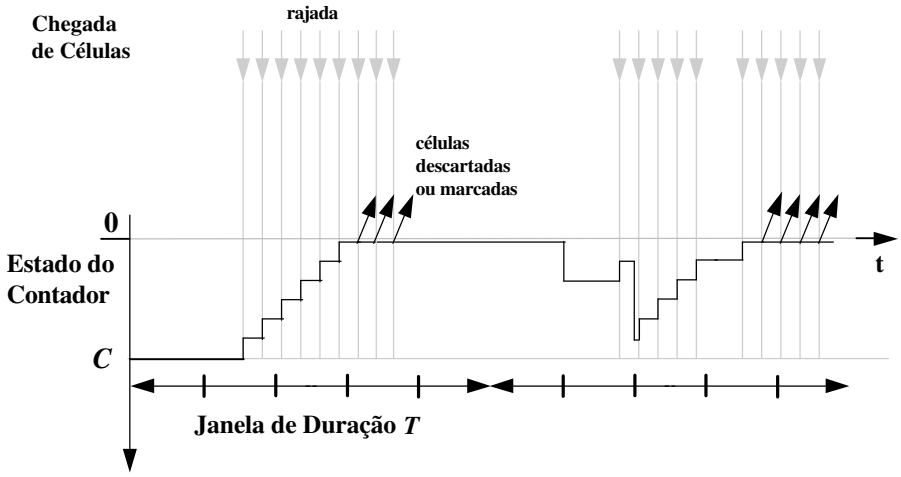
Para se buscar a eficiência em relação a restrição do comprimento máximo da rajada é necessário "memorizar" informações ou em outras palavras superpor janelas.

Neste mecanismo a janela de duração *T* é subdividida em *P* períodos de tempo iguais, denominados partições. Analogamente à janela saltitante e à janela deslizante, a cada chegada de célula um contador, estando abaixo do limite de créditos *C*, é decrementado. *T* unidades de tempo após o início de uma partição p_i, o contador é incrementado do número de células que chegaram durante esta partição p_i. Assim, o mecanismo janela móvel particionada é baseado na idéia de "janelas superpostas que saltam, de partição em partição" ao invés de deslizarem. A Figura III.10 ilustra o comportamento da janela móvel particionada para 2 e 5 partições.

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.



(a)



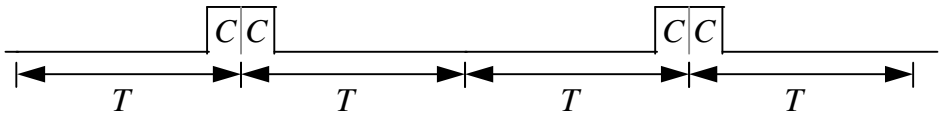
(b)

Figura III.10: O mecanismo janela móvel particionada para a) 2 partições e b) 5 partições.

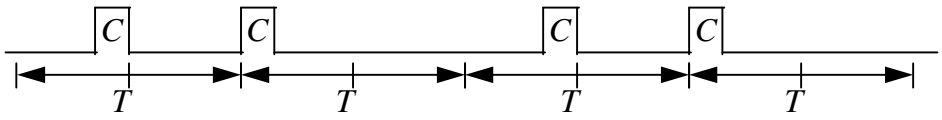
O tráfego de pior caso é gerado por rajadas de comprimento *C*. As rajadas são sincronizadas de forma alternada com as partições. O final de uma rajada sincroniza com o final de uma partição, enquanto que a rajada posterior começa no mesmo instante do início da partição que ocupa. Este padrão se repete periodicamente. Quando a duração da rajada é menor que a duração da partição, o período de silêncio mínimo entre as rajadas tem duração igual a P - 1 partições. Como cada partição dura T/P unidades de tempo, a duração do período mínimo de silêncio, nesta situação , pode ser expressa como:

$$\frac{(P-1)\times T}{P}.$$

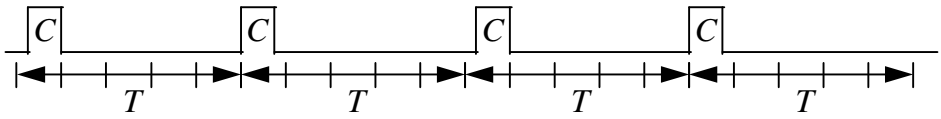
Assim como na janela deslizante este mecanismo limita o comprimento da rajada a *C* células enquanto na janela saltitante o comprimento máximo da rajada é de 2*C* células. No entanto, o mecanismo não consegue mais manter a característica da janela deslizante onde em qualquer período de tempo igual a *T* apenas *C* células serão submetidas a rede. Como na janela saltitante pode-se chegar até a 2*C* células em período de tempo *T* mas elas estarão espaçadas de um período de tempo no mínimo igual a uma partição que no pior caso será de *T*/2 para duas partições. Pode-se constatar que desde a sua configuração mais simples, a partir de duas partições, o mecanismo janela móvel particionada já é tão restritivo quanto o mecanismo janela deslizante em termos de comprimento da rajada, limitando-a a *C* células.



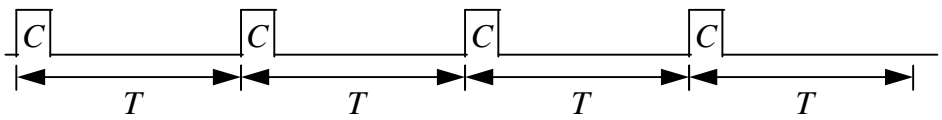
(a) janela saltitante



(b) janela móvel particionada com 2 partições



(c) janela móvel particionada com 5 partições



(d) janela deslizante

Figura III.11: Tráfego de pior caso para mecanismo janela móvel particionada

Para um aumento no número de partições o intervalo mínimo entre as rajadas também aumenta. Já para *P* tendendo a infinito, a duração das partições tende a zero e cada célula de uma rajada chega em uma partição diferente. Nesta situação, o padrão do tráfego de pior caso da janela móvel particionada tende ao padrão do tráfego de pior caso da janela deslizante.

O gerenciador de qualidade de serviço é responsável por parametrizar o número de partições deste mecanismo, de acordo com os requisitos de QOS estabelecidos.

III.3 Mecanismos para Garantia de QOS

Esta seção apresenta um conjunto de mecanismos para garantia de qualidade de serviço que definem sua própria disciplina de serviço, de forma a oferecer garantias quanto à vazão, proteção entre fluxos, e no caso de alguns mecanismos, limites para o atraso e *jitter* fim-a-fim.

III.3.1 *Virtual Clock*

O *Virtual Clock* [Zha91] é um algoritmo de controle de tráfego baseado no TDM (*Time Division Multiplexing*) que objetiva controlar a taxa de transmissão do fluxo de dados, monitorar o fluxo de dados estatísticos, provendo garantias quanto à vazão e proteção entre os fluxos, e resultando em baixos atrasos nas filas (porém sem garantias quanto ao atraso fim-a-fim). Um sistema TDM elimina completamente a interferência entre os fluxos já que cada fluxo transmite apenas em intervalos de tempo específicos. Entretanto, a capacidade do canal é desperdiçada quando em um dado intervalo de tempo, o fluxo correspondente não possui dados prontos para serem transmitidos.

No algoritmo do *Virtual Clock*, a cada fluxo está associada uma variável VC que evolui da média do tempo entre pacotes (Vtick) com a chegada de cada pacote. Se a fonte envia os pacotes de acordo com a taxa especificada, seu VC acompanhará o relógio de tempo real

(RT). O *Virtual Clock* simplesmente ordena a transmissão dos pacotes de acordo com a variável auxVC. Esta variável corresponde ao tempo no qual o pacote deveria ser transmitido se estivesse num sistema TDM. Este mecanismo de ordenação provê a proteção entre os fluxos, de forma que o fluxo mais "mal-comportado" receberá o pior serviço. Como o único quesito a ser garantido pelo *Virtual Clock* é a vazão, os únicos parâmetros de descrição do seu fluxo são a Taxa de Transmissão Média (AR) e o Intervalo para Monitoração Médio (AI). O intervalo médio AI pode ser associado ao parâmetro Tamanho Máximo da Rajada da *flowspec*. O funcionamento do *Virtual Clock* é descrito a seguir.

- Ao receber o primeiro pacote do fluxo *i*,

  $VC_i \leftarrow RT$;

  $auxVC_i \leftarrow RT$;

  onde RT é o valor do relógio de tempo real e VC, a variável relógio virtual.
- Ao receber os demais pacotes do fluxo *i*,
  1. $auxVC_i \leftarrow \max (RT, auxVC)$;
  2. $VC_i \leftarrow (VC_i + Vtick_i)$;

     $auxVC_i \leftarrow (auxVC_i + Vtick_i)$;

     Se todos os pacotes tiverem um tamanho fixo, então $Vtick_i = 1/AR_i$ (pacotes/seg);

     Caso contrário o valor de $Vtick_i$ deve ser computado individualmente por pacote;
  3. Estampar o pacote com o valor da variável $auxVC_i$.
- Se não houver *buffers* disponíveis, descartar o último pacote da fila.
- Transmitir os pacotes na ordem crescente dos valores das estampas.
- Após receber um conjunto de AIR_i (= AR_i x AI_i) pacotes do fluxo *i*,

  Se (VC_i - RT) > Tmax, ações de controle devem ser tomadas, pois a fonte está transmitindo a uma taxa superior à negociada;

  Se (VC_i < RT) então $VC_i \leftarrow RT$ e $auxVC_i \leftarrow RT$.

Do ponto de vista da rede o *Virtual Clock* funciona como um monitor, um medidor de fluxo de acordo com a chegada de pacotes. A diferença (VC-RT) mostra o quão próximo está a taxa de transmissão real da negociada. Nesta monitoração no entanto, não está claro como a aplicação deve escolher um valor razoável para AI. Se AI for mínimo (AI = 1/AR), os pacotes deverão ser transmitidos exatamente na taxa negociada, por outro lado se AI for muito grande, a rajada permitida também o será, possibilitando a perda excessiva de pacotes por falta de *buffers*. Um outro ponto obscuro é a escolha do valor *Tmax*, a partir do qual considera-se que o fluxo está transmitindo a uma taxa excessivamente superior à negociada.

Um mecanismo de prioridade pode ser facilmente implementado no *Virtual Clock*, simplesmente diminuindo a variável *VC* e *auxVC* de uma certa quantidade P_i no início do fluxo *i*, adiantando portanto os pacotes do fluxo *i*:

- Ao receber o primeiro pacote do fluxo *i*,

  $VC_i \leftarrow \quad RT - P_i;$

  $auxVC_i \leftarrow \quad RT - P_i;$

A ordenação da transmissão dos pacotes de acordo com a variável *auxVC* assegura que mesmo que um fluxo "mal comportado" utilize recursos disponíveis, este não comprometerá o serviço dos demais fluxos, pois quanto mais rápido chegam os pacotes do fluxo *i*, mais rápido cresce *auxVCi*, e portanto mais para o final da fila de transmissão ficam os pacotes deste fluxo. O mecanismo de controle de reserva de recursos garante que não haverá congestionamento se os fluxos transmitirem à taxa negociada. Porém, caso haja congestão, os pacotes do final da fila serão descartados.

III.3.2 *Fair Queueing*

A idéia básica do *Fair Queueing* [Kes90a, Kes91a] é bastante simples. Se N canais compartilham um único canal de saída, então cada canal tem direito a 1/N da banda passante,

no entanto, se algum canal utilizar menos, o restante é distribuído igualmente entre os demais. Isto é obtido através de um mecanismo de transmissão do tipo Round Robin (RR) entre os canais. *Fair Queueing* foi descrito em um contexto de orientação à datagrama e portanto nenhuma especificação de tráfego é considerada. A ordenação da transmissão dos pacotes pelo *Fair Queueing* é equivalente a feita pelo *Virtual Clock*, conforme ilustrado na Tabela III.1. A diferença básica entre o *Fair Queueing* e o *Virtual Clock* é que este último é baseado em reserva de recursos independentes para cada fluxo enquanto o *Fair Queueing* compartilha igualmente os recursos disponíveis.

Tabela III.1: Comparação entre os algoritmos *Fair Queueing* e *Virtual Clock*.

Virtual Clock	***Fair Queueing***	Equivalências
auxVC ← max (RT, auxVC)	$F^k \leftarrow \max (R^k, F^k)$	$RT \Leftrightarrow R^k$
auxVC ← auxVC + Vtick	$F^k \leftarrow F^k + P/n$	$auxVC \Leftrightarrow F^k$
Ordem Transmissão = auxVC	Ordem Transmissão = F^k	$Vtick \Leftrightarrow P/n$

Apesar de não considerar nenhum parâmetro de QOS, o *Fair Queueing* de maneira análoga ao *Virtual Clock*, oferece proteção entre os fluxos em relação à banda passante utilizada e maximiza o intercalamento entre os pacotes dos diferentes fluxos. No entanto, considera que todos os fluxos possuem as mesmas necessidades, fornecendo a cada um a mesma parcela da banda passante do canal, já que a cada pacote que chega, o incremento da variável F^k é constante (P/n) e dependente do número total de fluxos. Enquanto que no *Virtual Clock* o incremento $Vtick_i$ é específico do fluxo *i* e dependente apenas do tamanho do pacote e da taxa de transmissão máxima do fluxo *i*.

III.3.3 *Stop-and-Go*

A estratégia adotada pelo *Stop-and-Go* [Gol90] introduz a noção de quadros de tempo (*time frames*) e é composta por um mecanismo de transmissão de pacotes por conexão no nó fonte e por uma disciplina de serviço nos nós subsequentes. A idéia básica é inserir um padrão "suave" de tráfego e preservá-lo ao longo da rede. Os parâmetros de descrição do tráfego são representados pelo par (*r, T*), que define a "suavidade" do tráfego, onde durante um quadro de tempo de duração T, não serão transmitidos mais de *rT* bits. Consequentemente, se os pacotes forem de tamanho fixo e igual a Γ , então o número de pacotes em *T* é limitado em rT/Γ . Este mecanismo de transmissão de pacotes pela fonte garante a "suavidade" na entrada da rede. Para um determinado fluxo *k*, a transmissão de pacotes deve ser limitada em (r_k, *T*). Caso um fluxo viole o limite r_kT em um período *T*, seus pacotes excedentes serão descartados. Supondo que a fonte respeite os parâmetros (r_k, *T*), então eles também serão respeitados nos nós subsequentes, devido a disciplina de serviço utilizada.

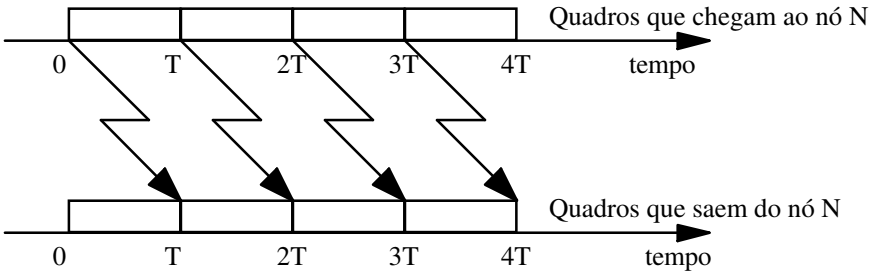


Figura III.12: Disciplina de Serviço Stop-and-Go.

Em cada quadro de tempo *T* são transmitidos os pacotes que chegaram ao nó *N* no quadro anterior. Desta forma o *Stop-and-Go* mantém o padrão do tráfego injetado na rede ao longo dos nós susequentes. Não é especificado nenhum limite máximo para o atraso fim-a-fim, mas pode-se observar que este é limitado em $\tau + HT \leq atraso \leq \tau + 2HT$, onde τ é a soma dos atrasos de processamento nos nós mais o atraso de propagação fim-a-fim e *H* é o número de nós ao longo do caminho. Como os limites no atraso e no *jitter* estão fortemente

dependentes do tamanho do quadro de tempo, múltiplos tamanhos de quadro (T) são oferecidos. Em cada nó é alocado um *buffer* de tamanho $2C_lT$, suficiente para eliminar qualquer possibilidade de transbordo de memória, onde C_l é a capacidade do canal l. Este mecanismo oferece, portanto, garantias quanto à vazão, atraso e *jitter*, além de ser livre de perdas.

III.3.4 *Hierarchical Round Robin*

O *Hierarchical Round Robin* (HRR) [Kes90b] possui vários níveis de serviço, onde cada nível funciona como um *Round Robin* (RR) secundário, com prioridades diferenciadas, conforme ilustrado na Figura III.13(a). O objetivo do HRR é que os níveis tenham prioridades específicas e que cada nível compartilhe de uma banda passante constante, reservada por nível. De maneira similar ao *Stop-and-Go* o tempo é dividido em quadros, resultando em um limite no atraso e no *jitter* fim-a-fim. Os parâmetros de QOS utilizados são herdados do *Virtual Clock* e são portanto a taxa média AR em um intervalo de tempo AI.

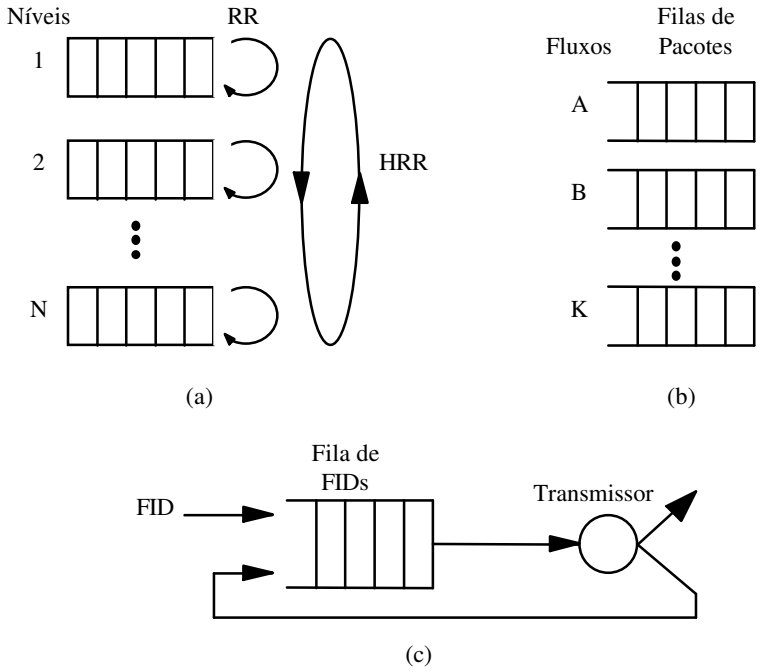


Figura III.13: *Hierarchical Round Robin*.

Os pacotes que chegam ao nó são organizados em filas individuais por fluxo (Figura III.13(b). Quando um pacote chega ao nó, este é colocado em sua respectiva fila, e o seu identificador de Fluxo (FID) é inserido no final de uma fila de FIDs (Figura III.13(c)), caso esta já não o contenha. Um *bit* de flag por fluxo assegura que existe somente um FID por fluxo na fila de FIDs. O primeiro FID indica qual o próximo fluxo a ter pacotes transmitidos. O número de pacotes a ser transmitido (quantum) pode ser diferente de acordo com o FID, e determina o nível de prioridade dado ao fluxo. Se durante a fase de transmissão a fila de pacotes do fluxo correspondente ficar vazia, o *bit* de flag deste fluxo é desativado e o transmissor passa a servir o próximo FID da fila. Caso o transmissor termine de enviar o quantum deste fluxo e ainda houver pacotes na fila de pacotes correspondente, então o FID deste fluxo é re-inserido no final da fila de FIDs, conforme mostrado na Figura III.13(c).

III.3.5 *Delay Earliest-Due-Date*

Os algoritmos *Delay-EDD* [FeV90a] e *Jitter-EDD* [VZF91] possuem três parâmetros de especificação de tráfego em comum: X_{min} é o menor intervalo de tempo entre pacotes (determinando a taxa máxima), X_{ave} é o intervalo de tempo entre pacotes médio em um intervalo I (determinando respectivamente a taxa média e o tamanho da rajada). O *Delay-EDD* visa oferecer garantias quanto à vazão e principalmente visa oferecer limites (estatísticos ou determinísticos) para o atraso fim-a-fim. Para tanto, faz uso de mais dois parâmetros: D_{max} e Z_{min}, tais que $D_i \leq D_{máx}$, para todo i, onde D_i é o atraso fim-a-fim do pacote i, no caso de fluxos determinísticos e

$$\text{Prob}\ (D_i \leq D_{max}) \geq Z_{min},$$

no caso de fluxos estatísticos.

Em [FeV90a] o *Delay-EDD* mantém 3 filas: uma para fluxos determinísticos, outra para estatísticos e outra para os demais fluxos. A disciplina de serviço adotada é uma variação

do mecanismo *EDD* (*Earliest-Due-Date*), onde em caso de conflito é dado uma prioridade maior aos canais determinísticos que aos estatísticos. No entanto, esta disciplina de serviço, conhecida como *class-EDD*, pode ser facilmente generalizada e expandida para *n* tipos de canais, com prioridades decrescentes. A ordem dos pacotes nas filas é crescente de acordo com as suas datas-limites (*deadlines* - *DL*). Na fila determinística as datas-limites devem ser alinhadas, ou seja, apropriadamente reduzidas para evitar conflito entre dois pacotes determinísticos.

O funcionamento do *class-EDD* é simples: compara-se a data-limite do primeiro pacote da fila de prioridade *k* (DL_k) com o tempo inicial (*TI* = data-limite - tempo de serviço) do primeiro pacote da fila de prioridade *k+1*. Se a data-limite DL_k for maior que o tempo inicial TI_{k+1} então deve-se transmitir o pacote de maior prioridade (*k+1*), caso contrário pode-se aplicar o algoritmo às filas de menor prioridade sem risco de ultrapassar a data-limite do pacote de maior prioridade. O pseudo-código deste mecanismo é descrito a seguir.

- Variáveis:

  D1 é o primeiro pacote da fila determinística;

  E1 é o primeiro pacote da fila estatística e;

  O1 é o primeiro pacote da fila sem prioridade.

- Procedimento:

  Se $(TI_{D1} < DL_{E1})$ então enviar o pacote *D1*.

  Senão

  Se $(TI_{E1} < DL_{O1})$ então enviar o pacote *E1*.

  Senão enviar o pacote *O1*.

A implementação deste mecanismo é bastante simples, no entanto a complexidade do *Delay-EDD* está na atribuição das datas-limites aos pacotes. Esta atribuição parte do princípio que se conhece o tempo máximo que um pacote do fluxo *i* pode permanecer em um nó *n* (d_{in}). O cálculo da data-limite leva em consideração a taxa de transmissão máxima, de forma que a

diferença entre as datas-limites de dois pacotes consecutivos do mesmo fluxo não seja superior ao intervalo mínimo entre pacotes. O pseudo-código deste cálculo é descrito a seguir.

- Variáveis:

  DL_{ik} é a data-limite do *k*-ésimo pacote do fluxo *i*;

  $Xmin_i$ é o intervalo mínimo entre pacotes do fluxo *i*;

  AT é o tempo de chegada do pacote ao nó *n*.

- Procedimento:

  $DL_{ik} \leftarrow \quad DL_{ik-1} + Xmin_i$

  $DL_{ik} \leftarrow \quad max\ (AT + d_{in},\ DL_{ik})$

O tempo máximo que um pacote do fluxo *i* pode permanecer em um nó *n* (d_{in}) é obtido durante a fase de controle de admissão de fluxos. Nesta fase de estabelecimento do fluxo, a mensagem de estabelecimento segue ao longo dos nós onde são feitos testes para verificar a capacidade de processamento dos nós, estimativas dos atrasos sofridos pelo pacote em cada nó e testes para evitar a saturação do escalonador de pacotes. A estimativa do atraso (d_{in}) é transmitida para os nós subseqüentes até o destino, onde são realizados os testes finais e onde é calculado o valor de d_{in} a ser utilizado nos nós. Estes valores são transmitidos de volta à origem, e em cada nó a variável d_{in} é atualizada. A equação (III.1) é o teste para garantir que o atraso fim-a-fim será respeitado. A equação (III.2) indica o valor de d_{in} a ser utilizado nos nós. Nestas equações D_i é o atraso máximo fim-a-fim solicitado pela aplicação e *N* é o número total de nós entre a fonte e o destino.

$$D_i \geq \sum_{n=1}^{N} d_{in} \qquad \text{(III.1)}$$

$$d_{in} = \frac{1}{N}[D_i - \sum_{n=1}^{N} d_{in}] + d_{in} \qquad \text{(III.2)}$$

III.3.6 *Jiiter Earliest-Due-Date*

O *Jitter-EDD* é uma evolução do *Delay-EDD* e portanto, visa oferecer garantias quanto à vazão e ao atraso fim-a-fim, mas principalmente visa oferecer limites determinísticos para o *jitter* fim-a-fim. Para tanto, faz uso do parâmetro: $J_{máx}$ definido de forma análoga ao parâmetro $D_{máx}$ do *Delay-EDD* e. corresponde exatamente ao parâmetro Variação de Atraso Máxima da *flowspec*.

A forma mais simples de eliminação de *jitter* é através da reserva de recursos (*buffers*) no sistema receptor. A quantidade de *buffer* requerida para remover a distorção no sistema receptor é fortemente determinada pela variação do atraso na transmissão fim-a-fim. Conhecendo-se o atraso médio *D*, o *jitter* máximo *J* e a taxa de transmissão máxima *R*, então o sistema receptor necessitará de um *buffer* de tamanho *2JR*. Para entender esta equação considere dois pacotes, A e B, enviados com T unidades de tempo de intervalo. No pior caso, A chegará depois de um atraso de *D-J* unidades de tempo (atraso mínimo) e B chegará depois de um atraso *D+J* unidades de tempo (atraso máximo). O receptor não pode entregar B até que ele chegue, o que no máximo ocorrerá em *T+2J* unidades de tempo depois de A. Para assegurar que A seja entregue *T* unidades de tempo antes de B, A deve permanecer no *buffer* por *2J* unidades de tempo.

Na maioria dos casos a quantidade *buffer* necessária para a eliminação do *jitter* no sistema receptor pode ser muito grande. O *Jitter-EDD* consiste de um mecanismo distribuído de eliminação de *jitter*, através de reserva de recursos nos nós intermediários, visando obter atrasos constantes em cada nó. Além da limitação no *jitter*, o *Jitter-EDD*, diferentemente do *Delay-EDD*, oferece garantias quanto à perda de pacotes por transbordo de memória. O mecanismo de reserva de *buffers* descrito em [Fer90c] foi originalmente proposto para em conjunto com o *Delay-EDD* oferecer garantias quanto à sensibilidade a perdas. No *Jitter-EDD* foi proposta uma pequena variação neste mecanismo para a alocação de *buffers*

adicionais para eliminação de *jitter*. O parâmetro de sensibilidade a perdas (W_{min}) utilizado é estatístico, de forma que

$$\text{Prob (mensagem entregue correta)} \geq W_{min}.$$

Neste mecanismo, cada nó executa duas funções: o módulo regulador é responsável por reconstruir o padrão original do fluxo e o módulo escalonador por assegurar que este padrão não seja muito distorcido. Para tanto introduz-se a noção de tempo de aptidão (*ET*). *ET* é o tempo a partir do qual o pacote está apto a ser transmitido e consiste do tempo de chegada (*AT*) do pacote ao nó mais o termo de correção para eliminação do *jitter*. Este termo de correção é parte do cabeçalho do pacote e é a diferença entre a data-limite e tempo real de transmissão do pacote no nó anterior. A disciplina de serviço utilizada é a mesma do *Delay-EDD* (*class-EDD*), no entanto o cálculo das datas-limites é diferente e os pacotes só são colocados em sua fila de transmissão após ter decorrido o seu tempo de aptidão. O pseudocódigo do cálculo das datas-limites é descrito a seguir.

- Variáveis:

  ET é o tempo de aptidão, a partir do qual o pacote está apto a ser transmitido;

  AT é o tempo de chegada do pacote ao nó *n*;

  CT é o termo de correção (parte do cabeçalho do pacote;

  DL_{ik} é a data-limite do *k*-ésimo pacote do fluxo *i*;

  $Xmin_i$ é o intervalo mínimo entre pacotes do fluxo *i*;

  j_{in} é o *jitter* máximo permitido para o pacote *i* no nó *n*.

- Procedimento:

  $ET \leftarrow AT + CT;$

  $DL_{ik} \leftarrow DL_{ik-1} + Xmin_i;$

  $DL_{ik} \leftarrow max\ (ET + j_{in},\ DL_{ik}).$

O termo j_{in} é calculado de maneira semelhante ao termo d_{in}, que no *Jitter-EDD* não é mais necessário já que neste mecanismo a garantia do atraso fim-a-fim é conseqüência da garantia do *jitter* fim-a-fim. Observe que o *jitter* máximo no último nó deve ser igual ao *jitter* máximo fim-a-fim. A Figura III.14 ilustra como o *jitter* é eliminado ao longo dos nós.

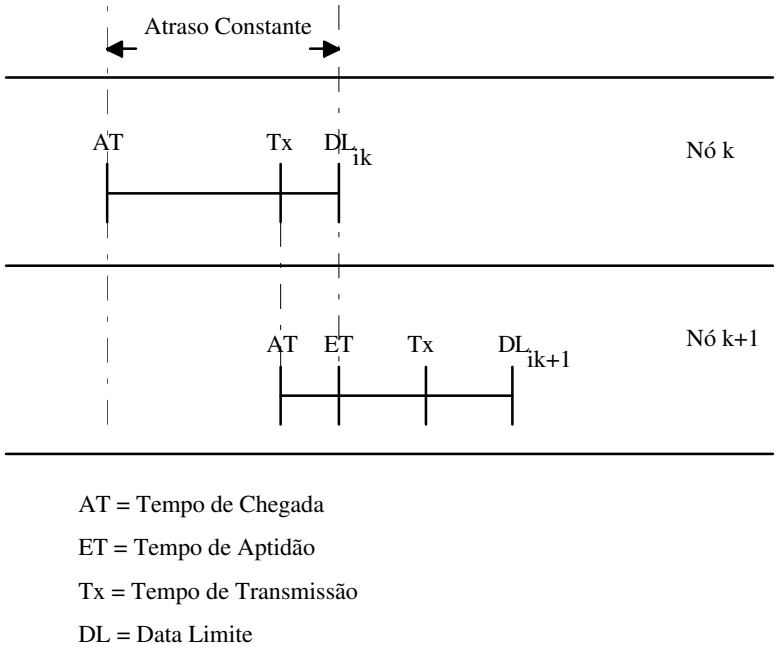


Figura III.14: Mecanismo de eliminação distribuída de *jitter* do *Jitter-EDD*.

III.3.7 *Rate Controlled Static Priority*

O algoritmo *Rate-Controlled Static-Priority* (RCSP) [ZFe93] visa prover garantias de vazão, atraso, *jitter* e confiabilidade (livre de erros). Sua arquitetura é bastante similar ao *Jitter-EDD* difere, no entanto, no fato de que o *Jitter-EDD* oferece um espectro contínuo de limites para o atraso, enquanto o RCSP oferece apenas um número fixo de limites para o atraso. A principal vantagem em relação ao *Jitter-EDD* é a simplicidade do processamento. Os parâmetros de QOS utilizados são herdados do *Jitter-EDD* (X_{min}, X_{ave}, I, D_{max}, J_{max}, W_{min}).

Cada nó é composto de um módulo regulador e um módulo escalonador. São oferecidos dois tipos de reguladores, um para a eliminação do *jitter* de vazão e outro para a eliminação do *jitter* de atraso. Cada fluxo utiliza um único módulo regulador. Estes reguladores utilizam a mesma noção de tempo de aptidão (*ET*), de forma que os pacotes só são inseridos na fila do escalonador após o tempo *ET*. O cálculo do tempo de aptidão para o regulador de *jitter* de atraso é descrito a seguir.

- Variáveis:

  ET é o tempo de aptidão;

  AT 'o tempo de chegada;

  d_j é o atraso máximo no escalonador do nó j;

  $\pi_{j,j-1}$ é o atraso de propagação entre o nó $j-1$ e o nó j.

- Procedimento:

  $ET_0^k = AT_0^k$;

  $ET_j^k = ET_{j-1}^k + d_{j-1} + \pi_{j-1,j}$.

O principal resultado deste cálculo é a reconstrução total do padrão do tráfego original na saída do regulador de cada nó, conforme a equação III.3.

$$ET_j^{k+1} - ET_j^k = AT_0^{k+1} - AT_0^k; \qquad \text{(III.3)}$$

O módulo escalonador é composto de *m* filas de tráfego tempo real com prioridades decrescentes e de uma fila sem prioridade. A disciplina de serviço executada pelo escalonador seleciona os pacotes em uma ordem FCFS (*first come first served*) da fila de maior prioridade não vazia, de forma que os pacotes da fila sem prioridade só são enviados quando não há nenhum pacote nas filas de tráfego de tempo real a ser transmitido. Este algoritmo só oferece garantias para os parâmetros de QOS se forem respeitadas as condições para o controle de admissão de fluxos que limita o número de fluxos em cada nível de prioridade.

III.3.4 Comparação dos Mecanismos

A Tabela III.2 ilustra os parâmetros de QOS utilizados por cada mecanismo onde os parâmetros da *fowspec* são utilizados como referência. Os parâmetros da mesma linha são equivalentes, no entanto foi mantida a notação mencionada na descrição original dos algoritmos. A *flowspec* foi usada como base por ter sido um dos primeiros trabalhos na área e por ser largamente usada na comunidade científica como referência de estrutura de dados para a representação dos requisitos de QOS. A sua principal vantagem é a flexibilidade na especificação dos parâmetros e a possibilidade de suportar futuras modificações nos seus campos.

Tabela III.2: Caracterização da Qualidade de Serviço.

	Flowspec	**FQ**	**VC**	**SG**	**HRR**	**D-EDD**	**J-EDD**	**RCSP**
Vazão	Tam. max					S_{max}	S_{max}	S_{max}
	Taxa max.					X_{min}	X_{min}	X_{min}
	Taxa med.		AR	r	AR	X_{ave}	X_{ave}	X_{ave}
	Raj. max.		AI	T	AI	I	I	I
Atraso						D_{max}	D_{max}	D_{max}
						Z_{min}		
	D min							
Jitter	*Jitter* Max.						J_{max}	J_{max}
Perdas	Sensibilidade							
	Num Perdas					W_{min}*	W_{min}	W_{min}
	Intervalo							

* *Delay-EDD* utilizando o mecanismo de reserva de *buffers* proposto em [FeV90b].

A Tabela III.3 resume as principais características dos algoritmos. Todos os algoritmos considerados neste trabalho alocam recursos estaticamente durante a fase de controle de admissão do fluxo de forma a torná-los livre de perdas (caso a fonte respeite a taxa máxima de transmissão negociada).

A eficiência de um algoritmo de controle de tráfego pode ser medida considerando a utilização dos recursos da rede, tais como: capacidade do canal, memória, carga de processamento etc. Na Tabela III.3 o termo eficiência é utilizado apenas no sentido do uso do canal de comunicação, de forma que o escalonador nunca fique inativo se houver pacotes a transmitir.

Tabela III.3: Características de Garantia de Qualidade de Serviço.

Garantias	**VC**	**FQ**	**SG**	**HRR**	**D-EDD**	**J-EDD**	**RCSP**
Vazão	Sim	Não	Sim	Sim	Sim	Sim	Sim
Atraso	Não	Não	Sim	Sim	Sim	Sim	Sim
Jitter	Não	Não	Sim	Não	Não	Sim	Sim
Perdas	Livre	Livre	Livre	Livre	Livre	Livre	Livre
Eficiência	Sim	Sim	Não	Não	Sim	Não	Sim

As tabelas apresentadas nesta seção não visam indicar o melhor algoritmo de controle de tráfego, uma vez que sua escolha depende de uma série de fatores, tais como, requisitos de desempenho, configuração da rede, complexidade de implementação, etc. Vale ressaltar que os mecanismos *Delay-EDD*, *Jitter-EDD* e *RCSP* permitem uma melhor especificação do tráfego e dos requisitos de QOS. Os mecanismos *Stop-and-Go*, *Jitter-EDD* e *RCSP* garantem limites superiores e inferiores para o atraso fim-a-fim, limitando portanto o seu *jitter*. O atraso e o *jitter* máximos garantidos pelo mecanismo *Stop-and-Go* são função do número de nós intermediários entre a fonte e o destino. O mecanismo *RCSP* limita o valor máximo do *jitter*

em um conjunto limitado de valores possíveis. O mecanismo *Jitter-EDD* permite a especificação do *jitter* máximo desejado e provê garantias para o mesmo, se houverem recursos disponíveis nos nós intermediários. No entanto, só o mecanismo *Rate-Controlled Static Priority* limita o *jitter* sem que o transmissor fique inativo se houver pacotes a transmitir.

Os algoritmos apresentados se limitam a garantias de desempenho determinísticas e esquemas de alocação de recursos estáticos. Alguns recentes trabalhos de pesquisa buscam algoritmos de controle melhor adaptados a tráfegos com alto nível de rajada, onde o conceito de garantia de QOS é estatísticamente definido, permitindo uma melhor utilização dos recursos da rede. A alocação dinâmica de recursos é outro complexo tópico de grande interesse científico.