Algoritmos de Controle de Tráfego para Garantia da Qualidade de Serviço em Aplicações de Tempo Real

Célio Vinicius N. de Albuquerque
e-mail: celio@coe.ufrj.br

Otto Carlos Muniz B. Duarte
e-mail: otto@coe.ufrj.br

Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro
COPPE/EE - Programa de Engenharia Elétrica
C.P. 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil

Resumo

Dentre os principais desafios das futuras redes de alto desempenho, está a garantia da qualidade de serviço (QoS) das aplicações de tempo real. A questão principal reside em prover uma enorme variedade de serviços com seus diferentes requisitos de desempenho, maximizando a possibilidade de admissão de novos usuários sem alterar as características das comunicações já existentes, otimizando a utilização dos recursos da rede e, obviamente, garantindo a qualidade de serviço negociada. Neste trabalho são analisados diversos algoritmos de controle de tráfego para garantia da qualidade de serviço de aplicações de tempo real, dentre os quais *Leaky-Bucket, Virtual Clock, Fair Queueing, Stop-and-Go, Hierarchical Round Robin, Delay-Earliest-Due-Date, Jitter-Earliest-Due-Date e Rate-Controlled Static-Priority*. As formas de caracterização do tráfego do fluxo dos algoritmos são identificadas, verificam-se quais parâmetros de QoS são garantidos e apresentam-se tabelas comparativas. São ressaltadas algumas questões ainda não resolvidas na garantia da qualidade de serviço de aplicações de tempo real.

Abstract

One of the challenges and open issues of the future high performance networks is to provide quality of service (QoS) guarantees to real time applications. The main problem is to support a broad range of traffic services, with their performance requirement's variety, maximizing the network's capability to support new real time channels, without jeopardizing the performance of the already established connections, improving the utilization of the network resources and, obviously, providing the guaranteed QoS. This paper analises several rate-based control algorithms. These are Leaky-Bucket, Virtual Clock, Fair Queueing, Stop-and-Go, Hierarchical Round Robin, Delay-Earliest-Due-Date, Jitter-Earliest-Due-Date and Rate-Controlled Static-Priority. Traffic Parameters used to describe the flow are identified, the QoS guarantees are verified and comparative tables are presented. Open issues in providing QoS guarantees are identified.

1 Introdução

Os recentes avanços na tecnologia dos semicondutores (VLSI) e o advento das fibras óticas nas redes de comunicação permitiram um aumento considerável da capacidade de transmissão e taxas de erros cada vez menores. Este acréscimo de desempenho estimulou a demanda por novos serviços com os mais diferentes, e por vezes até desconhecidos, requisitos de qualidade. A integração dos mais variados tipos de serviços resultou no interesse pelo desenvolvimento de uma rede única, universal e suficientemente flexível para prover todo tipo de serviço com garantias sobre a qualidade de serviço (QoS). A resposta tecnológica a este desafio foi o uso do modo de transferência assíncrono (*Asynchronous Transfer Mode - ATM*) que consiste basicamente na comutação de mini-pacotes de tamanho fixo (células) e funções simplificadas nos nós. A futura Rede Digital de Serviços Integrados em Banda Larga (RDSI-BL) [Pry91] usa esta técnica e deve prover suporte para aplicações distribuídas tais como: distribuição domiciliar de TV de alta definição (*High Definition TeleVision* - HDTV), vídeo-conferência (ou conferência multimídia), transferência de dados a alta velocidade, vídeo-telefonia, vídeo-biblioteca, educação domiciliar distribuída, visualização, realidade virtual, trabalho cooperativo (*Computer Supported Cooperative Work* - CSCW) e outras.

Dentre os novos serviços a serem providos pelas futuras redes de alto desempenho destacam-se, pelos seus fortes requisitos de qualidade de serviço, os conhecidos como serviços isócronos. A sua principal característica é a forte dependência temporal onde é permitida pouca ou nenhuma variação nos atrasos da comunicação, caso contrário a qualidade seria insatisfatória.

Estas novas classes de aplicações tem a propriedade de requererem um sistema distribuído capaz de prover garantias sobre a qualidade de comunicação entre as aplicações. Aplicações multimídia, com seus fortes requisitos de alta velocidade e serviços de alta qualidade, têm estimulado o desenvolvimento de novas arquiteturas e protocolos de redes de comunicação. Um dos problemas mais importantes a ser resolvido é como prover às aplicações uma qualidade de serviço (QoS) garantida. Isto não requer somente grandes bandas passantes, baixos atrasos e baixa taxa de erro (ou perda), mas também limites (superiores ou inferiores) garantidos para estes índices de desempenho.

Para caracterizar uma comunicação de tempo real normalmente faz-se uso de uma entidade abstrata denominada fluxo (*flow*). Um fluxo é uma cadeia de unidades de informação (pacotes, células, octetos, ...) com os mesmos requisitos de desempenho que segue uma rota simplex fixa entre a fonte e o destino.

A capacidade de garantir um desempenho específico torna-se particularmente importante em redes suportando aplicações de tempo real. Entende-se por tempo real, um serviço que possibilita a um usuário especificar os seus requisitos de desempenho e obter garantias quanto ao cumprimento destes requisitos [Fer90a]. A interação usuário-rede deve portanto ser bastante ampla e a sua interface bastante flexível de forma a possibilitar a especificação dos requisitos de qualquer tipo de fluxo. Uma questão importante na interação usuário-rede é o controle de admissão de conexão. A rede deve ser dotada de um mecanismo que seja capaz de responder se um novo fluxo pode ser estabelecido e seu QoS garantido, otimizando a utilização dos recursos da rede sem violar as garantias feitas aos fluxos já existentes. Para responder a esta questão a rede deve ser capaz de avaliar não apenas os parâmetros de QoS fim-a-fim do novo fluxo, mas também determinar o impacto da admissão deste novo fluxo no desempenho dos demais.

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]. 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 1 ilustra a mudança na taxa de transmissão do fluxo C resultante da multiplexação no nó N.

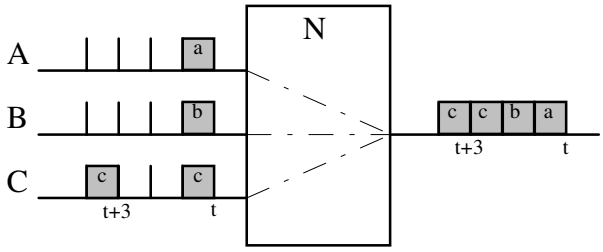

Figura 1: Mudança na Taxa de Transmissão devido à Multiplexação.

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 dos *jitters* de vazão e de atraso.

Com isso, novos modelos para redes são requeridos e propostos, onde tráfegos com diferentes requisitos de comunicação serão diferentemente gerenciados, de forma que aplicações com necessidades maiores serão privilegiadas e protegidas do tráfego de dados usual que flui nas redes de computadores atuais. No entanto, persiste a dúvida de quais os tipos de requisitos de comunicação uma aplicação de tempo real deve ter [Par92, Fer90a, Kur93]. Uma resposta precisa a esta questão é muito difícil devido ao grande número de aplicações que foram, têm sido e serão propostas. Enquanto algumas aplicações, p.ex., distribuição de vídeo, ainda que bastante exigentes em termos de banda passante, podem admitir alguma degradação na confiabilidade da entrega e nos atrasos na comunicação, outras, como vídeo-conferência e trabalho cooperativo (com caráter interativo) impõem fortes requisitos na qualidade de serviço geral. Existe, porém, um consenso de que a especificação dos requisitos de comunicação deve ser flexível o suficiente de forma a abranger ao máximo as necessidades das mais diversas aplicações de tempo real.

No que concerne ao controle de tráfego, a maioria das estratégias utilizadas nas redes de dados convencionais são baseadas em reconhecimento (*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 trabalho analisa e compara diversos algoritmos de controle de tráfego para garantia da qualidade de serviço orientados a controle de vazão. Os algoritmos aqui analisados são *Leaky-Bucket*, *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]. O objetivo da comparação dos algoritmos é explorar as suas diferenças, identificar o conjunto de parâmetros utilizados e o nível de garantia de QoS oferecido.

A organização deste trabalho é a seguinte. A seção 2 descreve as formas de caracterização do tráfego e os parâmetros de QoS utilizados pelos mecanismos acima. A seção 3 descreve e analisa cada algoritmo. A seção 4 apresenta uma tabela indicando os parâmetros usados por cada algoritmo e um quadro comparativo dos algoritmos. Finalmente, enfoca-se as questões ainda não resolvidas na garantia da qualidade de serviço de aplicações de tempo real.

2 Caracterização da Qualidade de Serviço

A qualidade do serviço de uma aplicação de tempo real é determinada pelos valores de alguns parâmetros de desempenho fim-a-fim. Tais parâmetros podem ser especificados em termos de três índices de qualidade e do nível de garantia da qualidade de serviço:

1. **Vazão do tráfego**: este índice está relacionado com a necessidade de caracterizar como o tráfego do fluxo será injetado na rede. Ele expressa a quantidade de dados que será enviada através da rede por unidade de tempo. Geralmente especifica as necessidades do tráfego da comunicação em termos da banda passante requerida;
2. **Atraso**: este índice de qualidade está relacionado com a sensibilidade das aplicações em relação ao atraso introduzido pela rede. Algumas aplicações são mais sensíveis que outras tanto em relação ao atraso quanto à variação no atraso (***jitter***). O termo *jitter* é comumente utilizado para expressar a variação no atraso, no entanto pode ser utilizado para expressar qualquer tipo de variação, como por exemplo *jitter* de vazão expressa variação na taxa de transmissão;
3. **Confiabilidade**: este índice está relacionado com a sensibilidade das aplicações em relação à perda de dados. Este índice de qualidade é fortemente dependente do mecanismo de armazenamento e reserva de recursos envolvido na transmissão da informação ao longo da rede. Existem duas fontes de perda de dados: devido a erros de transmissão (interferências ou falhas) e devido à transbordo de memória (*buffer overrun*). Os recentes trabalhos apresentados na literatura consideram basicamente a perda de células devido à transbordo em redes ATM.
4. **Nível de Garantia**: é esperado que a rede seja capaz de prover uma faixa de garantias de serviços: garantias fortes (perfeitas), suaves (imperfeitas) ou preditivas (probabilísticas). O nível de qualidade de serviço negociado e selecionado é fundamental para um bom desempenho da rede, pois ele influi nas probabilidades de sucesso na admissão de um determinado número de aplicações e na otimização da utilização dos recursos da rede.

Dentre as propostas de caracterização da qualidade de serviço citadas na literatura, a mais geral é a *flowspec* (*flow specification*) proposta em [Par92], onde são apresentados os parâmetros considerados pelo autor necessários e suficientes para especificar qualquer requisito de fluxo. A vazão do tráfego injetado na rede é caracterizada por quatro parâmetros: o comprimento máximo do pacote (*Maximum Transmission Unit* - MTU), a taxa de transmissão média (taxa de recebimento de créditos), a taxa de pico (taxa de transmissão máxima) e o tamanho de rajada (número de créditos máximo). A *flowspec* caracteriza a sensibilidade ao atraso através de dois parâmetros: o menor atraso perceptível e a variação de atraso máxima. A sensibilidade a perda de dados é caracterizada através de três parâmetros: a sensibilidade a perdas, a sensibilidade a perdas consecutivas e o intervalo de perdas. Além destes parâmetros, são ainda oferecidos seis níveis de qualidade do serviço.

A flexibilidade na especificação dos parâmetros e a possibilidade de suportar futuras modificações nos seus campos torna a *flowspec* uma ótima referência de estrutura de dados para a representação dos requisitos de QoS. No entanto, tal variedade de parâmetros não é utilizada em nenhum mecanismo de controle de tráfego conhecido na literatura.

De acordo com [Par92], existem dois problemas em permitir às aplicações especificarem um atraso máximo aceitável. Uma aplicação pode estar bastante satisfeita com um atraso de 100 ms entre o Brasil e o Japão, porém, bastante insatisfeita com um atraso de 100 ms numa comunicação interna a uma mesma cidade. Esta observação sugere que o atraso máximo é de fato variável e função do atraso considerado atingível. No entanto, o atraso atingível é fortemente determinado pela distância geográfica entre os pares comunicantes e este tipo de informação geográfica, em geral, não está disponível para a rede. E mais, com o advento dos sistemas de comunicação móveis, este tipo de informação se torna bastante difícil de se obter. Portanto, existem razões para acreditar que as aplicações devam ter sérias dificuldades em escolher um atraso máximo considerado razoável, tanto para o usuário quanto para a distância geográfica considerada.

O segundo problema com atrasos máximos é que eles tentam quantificar o desempenho que é aceitável para o usuário e uma aplicação, em geral, não sabe qual desempenho será aceitável para o seu usuário. Por exemplo, uma justificativa comum para especificar um máximo atraso aceitável é que considera-se humanamente difícil conversar com atrasos maiores que 100 ms. Certamente tal atraso pode tornar a comunicação menos agradável, mas é ainda possível se conversar mesmo com atrasos de vários segundos de duração, e dada a escolha entre a não conexão e um atraso longo, muitos usuários escolheriam o atraso longo.

3 Mecanismos de Controle de Tráfego

3.1 *Leaky-Bucket*

Um dos algoritmos mais citados na literatura para o controle da taxa de transmissão e policiamento do tráfego é o *Leaky-Bucket*. Este é um algoritmo proposto para ser implementado na controladora de comunicação das redes de alta velocidade. Várias versões deste algoritmo têm sido propostas, sendo que uma das mais completas 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.

Neste mecanismo (Figura 2) cada nó possui dois depósitos imaginários (*buckets*) para cada fluxo: um depósito de créditos (*bytes*) e um depósito de transmissão.

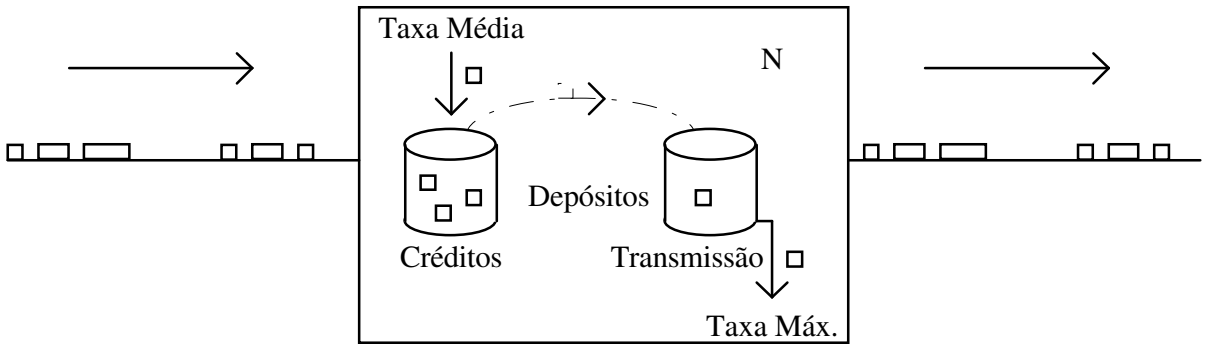

Figura 2: Estrutura do Algoritmo *Leaky-Bucket*

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 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 algoritmo 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.

3.2 *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). Note que 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 constante, 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 para AI razoável. 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 *Pi* no início do fluxo *i*, adiantando portanto os pacotes do fluxo *i*:

- Ao receber o primeiro pacote do fluxo *i*,
  $VC_i \leftarrow$ *RT - Pi;*
  $auxVC_i \leftarrow$ *RT - Pi;*

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 congestão se os fluxos transmitirem à taxa negociada. Porém, caso haja congestão, os pacotes do final da fila serão descartados.

3.3 *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 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 1: Comparação entre os algoritmos *Fair Queueing* e *Virtual Clock*.

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

Ordem Transmissão = auxVC	Ordem Transmissão = F^k	Vtick ⇔ 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*.

3.4 *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 susequentes. 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.

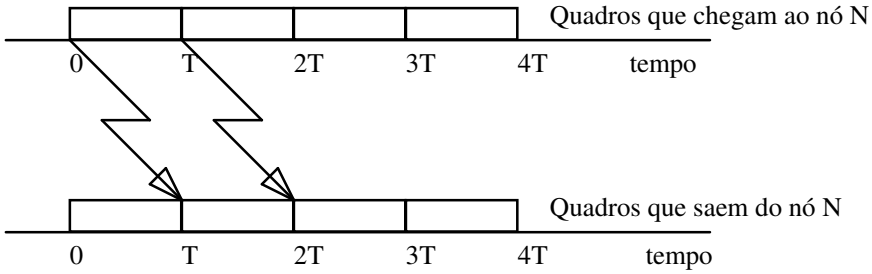

Figura 3: Disciplina de Serviço Stop-and-Go.

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 observe 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.

3.5 *Hierarchical Round Robin* (HRR)

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 4a. 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.

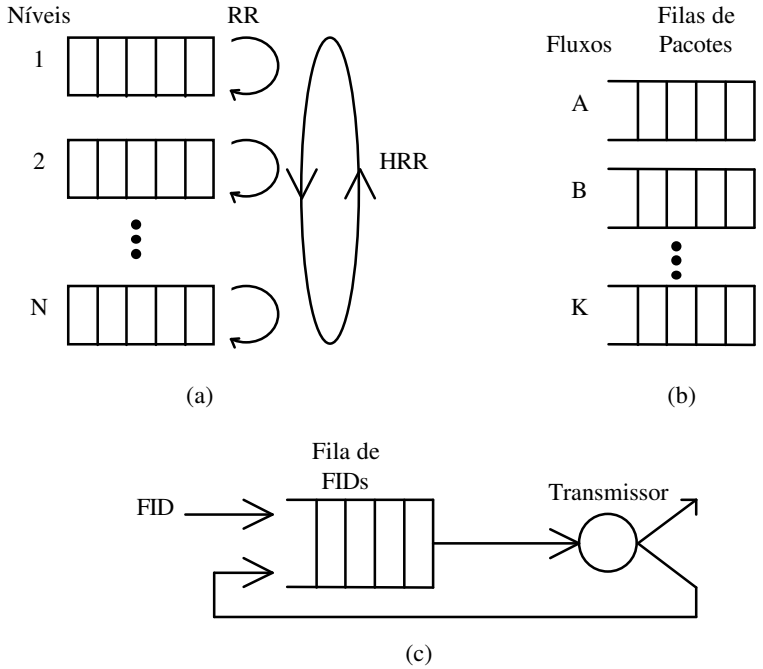

Figura 4: *Hierarchical Round Robin*.

Os pacotes que chegam ao nó são organizados em filas individuais por fluxo (Figura 4b). 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 4c), 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 4c.

3.6 *Delay-Earliest-Due-Date (Delay-EDD)*

Os algoritmos *Delay-EDD* [Fer90b] e *Jitter-EDD* [Fer91a] possuem três parâmetros de especificação de tráfego em comum: *Xmin* é o menor intervalo de tempo entre pacotes (determinando a taxa máxima), *Xave* 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 $Di \leq Dmáx$, para todo i, onde Di é o atraso fim-a-fim do pacote i, no caso de fluxos determinísticos e Prob ($Di \leq Dmax$) $\geq Zmin$, no caso de fluxos estatísticos.

Em [Fer90b] 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, i. e., 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:
  $DL_{ik} \leftarrow DL_{ik-1} + Xmin_i$
  $DL_{ik} \leftarrow max\ (AT + d_{in}, DL_{ik})$

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 (1) é o teste para garantir que o atraso fim-a-fim será respeitado. A equação (2) indica o valor de d_{in} a ser utilizado nos nós. Nestas equações *Di* é o atraso máximo fim-a-fim solicitado pela aplicação e *N* é o número total de nós entre a fonte e o destino.

$$D_i \geq \sum_{n=1}^{N} d_{in} \quad (1)$$

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

3.7 *Jitter-Earliest-Due-Date (Jitter-EDD)*

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: *Jmax* definido de forma análoga ao parâmetro *Dmax* 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 é 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 (*Wmin*) utilizado é estatístico, de forma que Prob (mensagem entregue correta) ≥ *Wmin*.

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 a *jitter* máximo permitido para o pacote *i* no nó $\underline{n}$.

- 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 5 ilustra como o *jitter* é eliminado ao longo dos nós.

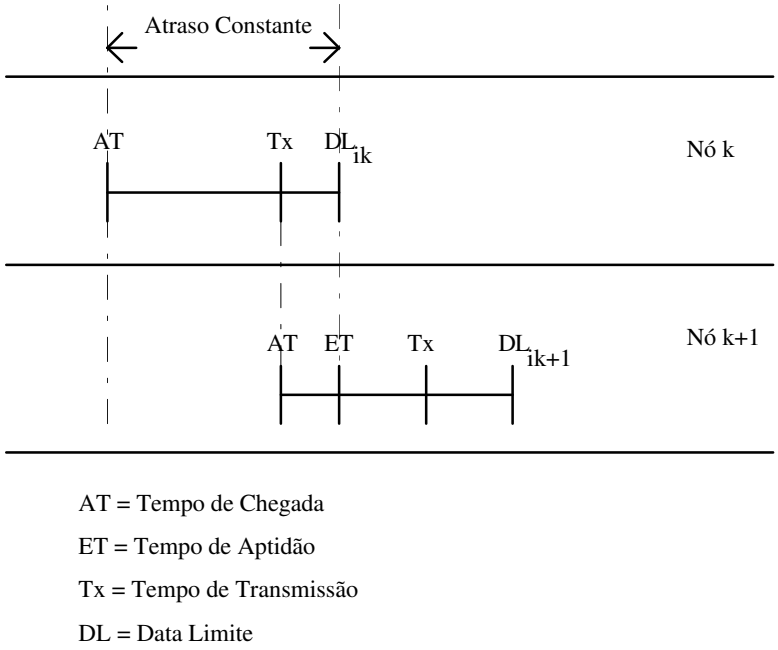

Figura 5: Mecanismo de Eliminação Distribuída de *Jitter* do *Jitter-EDD*.

3.8 *Rate-Controlled Static-Priority* (RCSP)

O algoritmo *Rate-Controlled Static-Priority* (RCSP) [Fer93] 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 de limites para o atraso contínuo, 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* (*Xmin, Xave, I, Dmax, Jmax, Wmin*).

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 (3).

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

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.

4 Considerações Finais

Este trabalho apresentou oito algoritmos de controle de tráfego para garantia da qualidade de serviço em aplicações de tempo real. O enfoque principal foi analisar os algoritmos, verificar que parâmetros de QoS são garantidos, identificando como cada um caracteriza o tráfego do fluxo. 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 2: Caracterização da Qualidade de Serviço.

	Flowspec	**LB**	**FQ**	**VC**	**SG**	**HRR**	**D-EDD**	**J-EDD**	**RCSP**
	Tam. max						Smax	Smax	Smax
Vazão	Taxa max.	Taxa max.					Xmin	Xmin	Xmin
	Taxa med.	Taxa med.		AR	r	AR	Xave	Xave	Xave
	Raj. max.	Raj. max.		AI	T	AI	I	I	I
							Dmax	Dmax	Dmax
Atraso							Zmin		
	D min								
Jitter	*Jitter* Max.							Jmax	Jmax
	Sensibilidade								
Perdas	Num Perdas						Wmin*	Wmin	Wmin
	Intervalo								

* *Delay-EDD* utilizando o mecanismo de reserva de *buffers* proposto em [Fer90c].

A Tabela 2 ilustra os parâmetros de QoS utilizados por cada mecanismo considerado neste trabalho 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 Tabela 3 resume as principais características dos algoritmos. Esta tabela não inclui o *Leaky-Bucket* pois este não define uma disciplina de serviço, garantindo apenas a vazão de um fluxo individual. 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).

Tabela 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

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 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. Esta tabela não visa 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.

Os algoritmos apresentados se limitam a garantias de desempenho determinísticas e esquemas de alocação de recursos estáticos. Os trabalhos de pesquisa atuais 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.

Referências

[Bro88] R. Brown, "Calendar Queues: A Fast 0(1) Priority Queue Implementation for the Simulation Event Set Problem", *Communications of the ACM*, vol. 31, no. 10, pp. 1220-1227, outubro de 1988.

[CSZ92] D. D. Clark, S. Schenker e L. Zhang, "Suporting Real Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", *Proc. ACM SIGCOMM 92*, pp. 14-26, Baltimore, agosto de 1992.

[Fer90a] D. Ferrari, "Client Requirements for Real-Time Communication Services", *IEEE Communications Magazine*, vol. 28, no. 11, pp. 65-72, novembro 1990.

[Fer90b] D. Ferrari e D. Verma, "A Scheme for Real Time Channel Establishment in Wide-Area Networks", *IEEE Journal on Selected areas in Communications*, vol. 8, no. 3, pp. 368-379, abril de 1990.

[Fer90c] D. Ferrari e D. Verma, "Buffer Space Allocation for Real-Time Channels in a Packet Switching Network", Technical Report TR-90-022, International Computer Science Institute, junho de 1990.

[Fer91] D.Verma, H. Zhang e D. Ferrari, "Delay Jitter Control for Real Time Communication in a Packet Switching Network", *Proceedings of TriComm 91*, pp. 35-46, Chapel Hill, abril 1991.

[Fer92] D. Ferrari, J. Ramaekers e G. Ventre, "Client-Networks Interactions in Quality of Service Communication Environments", *Proc. of HPN 92, 4th IFIP Conference on High Performance Networking*", Liege, Bélgica, dezembro 1992.

[Fer93] H. Zhang e D. Ferrari, "Rate-Controled Static-Priority Queueing", *Proceedings of IEEE INFOCOM*, 1993.

[Gol90] S. J. Golestani, "A Stop-and-Go Queueing Framework for Congestion Management", *SIGCOMM Symposium, Communications Architecture & Protocols*, pp. 8-18, Philadelphia Pennsylvania, setembro de 1990.

[Kes90a] A. Demers, S. Keshav e S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", *Journal of Internetworking Research and Experience*, pp. 3-26, outubro de 1990.

[Kes90b] C. Kalmanek, H. Kanakia e S. Keshav, "Rate Controlled Servers for Very High Speed Networks", *IEEE Global Telecommunications Conference*", San Diego, Califórnia, dezembro de 1990.

[Kes91a] S. Keshav, "On the Efficient Implementation of Fair Queueing", *Journal of Internetworking Research and Experience*, vol. 2, pp. 157-173, setembro de 1991.

[Kes91b] H. Zhang e S. Keshav, "Comparison of Rate-Based Service Disciplines", *Proceedings of ACM SIGCOMM*, pp. 113-121, setembro de 1991.

[Kur93] J. Kurose, "Open Issues and Challenges in Providing Quality of Service Guarantees in High Speed Networks", *ACM Computer Communication Review*, vol. 23, no. 1, pp. 6-15, janeiro 1993.

[Par92] C. Partridge, "A Proposed Flow Specification", Internet RFC-1363, setembro 1992.

[Pry91] M. de Prycker, "Asynchronous Tranfer Mode - Solution for Broadband ISDN", ed. Ellis Horwood, 1991.

[Zha91] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet-Switched Networks", *ACM Trans. on Computer Systems*, vol. 9, no. 2, maio 1991.