Análise de Desempenho de Algoritmos de Controle de Admissão de Conexões para Redes Multimídia

por

Magnos Martinello

UFRJ

Tese submetida para a obtenção do título de

Mestre em Ciências em Engenharia de Sistemas e Computação

ao Programa de Pós-Graduação de Engenharia de Sistemas e Computação

da COPPE/UFRJ

por

Magnos Martinello

Agosto 2000

Análise de Desempenho de Algoritmos de Controle de Admissão de Conexões para Redes Multimídia

Magnos Martinello

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Edmundo de Souza e Silva, Ph.D.

Prof. Paulo Henrique de Aguiar Rodrigues, Ph.D.

Prof. José Ferreira de Rezende, Ph.D.

Profa. Morganna Carmem Diniz, Dr.

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2000

MARTINELLO, MAGNOS

Análise de Desempenho de Algoritmos de Controle de Admissão de Conexões para Redes Multimídia [Rio de Janeiro] 2000

XIV, 80 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2000)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Redes Multimídia
2. Controle de Admissão
3. Gerador de Tráfego

I. COPPE/UFRJ II. Título (Série)

Aos meus familiares com muito carinho.

Agradecimentos

Agradeço à minha família, principalmente aos meus pais e aos meus irmãos pelo apoio e incentivo durante toda a minha formação.

Agradeço à minha namorada Roberta Lima Gomes, que compreendeu a minha dedicação e me apoiou em todas as horas.

Agredeço a colaboração e dedicação do Prof. Edmundo de Souza e Silva, durante todo o trabalho. Também agradeço a constante colaboração e incentivo da Professora Rosa Maria Meri Leão.

Agradeço aos meus amigos que me acompanharam diretamente ou indiretamente, em especial aos amigos Douglas, Sidney, Raniery, Carlos, Daniel, Flavio, Drika, Kelvin, Ana Paula, Claudio Prata, Igor, Elcio, Paulo André, Cesar, entre outros.

Ao pessoal da praia que sempre trouxeram momentos divertidos e alegres.

Esta Tese foi resultado de parte do trabalho realizado no projeto ALMADEM pertencente ao programa PROTEM-III financiado pelo CNPq e pelo programa PRONEX. Agradecimentos também a CAPES como entidade que financiou a bolsa de mestrado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

Análise de Desempenho de Algoritmos de Controle de Admissão de Conexões para Redes Multimídia

Magnos Martinello

Agosto/2000

Orientador: Edmundo de Souza e Silva
Programa: Engenharia de Sistemas e Computação

As redes multimídia são capazes de suportar uma ampla variedade de conexões com diferentes requisitos de banda e características de tráfego, e garantem qualidade de serviço (QoS) adequada a cada aplicação. O controle de admissão de conexões(CAC) é usado para decidir se uma nova conexão deve ou não ser aceita em um tronco de rede, sem prejuízo de QoS das conexões existentes. Visto que os parâmetros UPC (Usage Parameter Control) são os únicos parâmetros padronizados de caracterização do tráfego, o desenvolvimento de esquemas eficientes de CAC baseados nos parâmetros UPC são importantes para prover a QoS desejada. O objetivo deste trabalho é analisar diferentes algoritmos de CAC propostos na literatura e compará-los baseado em medições feitas em um comutador real. Para isso, selecionamos e implementamos dois algoritmos de CAC, um para tráfego não-regulado, proposto em [17] e outro para tráfego regulado pelo leaky bucket, proposto em [31]. Comparamos esses algoritmos usando o comutador ATM da Fore-Systems ASX-1000, disponível em nosso laboratório, e analisamos o comportamento deste algoritmo em relação a parâmetros que podem ser escolhidos pela aplicação. Para a realização de testes, desenvolvemos um gerador de tráfego com suporte IP e ATM-Nativo. O gerador foi incorporado ao ambiente Tangram-II [7, 35] de modelagem e análise.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Performance Analysis of Connection Admission Control Algorithms to Mutimidea Networks

Magnos Martinello

August/2000

Advisor: Edmundo de Souza e Silva
Department: Computer and System Engineering

Multimidia networks are capable of supporting a wide range of connections with different bandwidth requirements and traffic characteristics, and provide Quality-of-Service (QoS) guarantees to each application. Connection Admission Control (CAC) is used to decide whether an incoming connection should be accepted or rejected in a network node, keeping (QoS) of aplications. Since the parameters UPC(Usage Parameter Control) are the only standardized parameters to traffic characterization, developing efficient CAC schemes based on UPC parameters are important to provide the required QoS. The objective of this work is analyzing different CAC algorithms proposed in the literature and compare their effectiveness from measurements performed using an ATM switch. We selected and implemented two CAC algorithms, one for a non-regulated traffic, proposed by [17], other for a leaky-bucket regulated traffic, proposed by [31]. We compared those algorithms using the Fore-Systems ASX-1000 switch, available in our laboratory, and analyzed the algorithm behavior with respect to parameters that can be chosen by the application. To enable test executions, we developed a traffic generator supporting IP and native-ATM. The traffic generator is currently part of the Tangram-II modeling environment [7, 35].

Palavras-chave

1. Redes Multimídia

2. Controle de Admissão

3. Modelos Markovianos

4. Geração de Tráfego

Glossário

QoS :	Qualidade de Serviço (*Quality of Service*);
ATM :	Modo de Transferência Assíncrono (*Asynchronous Transfer Mode*);
VC :	Circuito Virtual(*Virtual circuit*);
CBR :	Taxa Constante de Bits (*Constant Bit Rate*);
UBR :	Taxa Não Definida de Bits (*Unknown Bit Rate*);
ABR :	Taxa Disponível de Bits (*Available Bit Rate*);
VBR-nrt :	Taxa Variável de Bits – não tempo real (*Variable Bit Rate – non real time*);
VBR-rt :	Taxa Variável de Bits – tempo real (*Variable Bit Rate – real time*);
CAC :	Controle de Admissão de Conexões (*Connection Admission Control*);
UPC :	Parâmetros de Controle para Uso (*Usage Parameter Control*);
SWITCH :	Equipamento de Comutação de pacotes ou células (*Switch*)

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

ANÁLISE de desempenho em redes de alta velocidade, através de modelos matemáticos, permite a compreensão em fase de projeto de problemas envolvidos no desenvolvimento de um equipamento, antes mesmo da sua construção e colocação em uso operacional. Por este motivo, é fundamental o desenvolvimento de modelos que possibilitem uma análise aproximada do comportamento do sistema.

As redes de alta velocidade viabilizam a provisão de diversos tipos de serviços que podem ser oferecidos de maneira integrada. Entre estes serviços podem ser citados vídeo sob demanda , teleconferência multimídia, transmissão de televisão de alta resolução, telefonia, trabalhos cooperativos em grupo, etc.

As redes ATM padronizadas pelo ITU(International Telecommunications Union), utilizam um paradigma de comunicação orientada a conexão, no qual os dados são segmentados em pacotes de tamanhos fixos chamados células. A interface de acesso UNI [26] permite o uso de uma variedade de serviços com diferentes requisitos de QoS. Vários padrões foram propostos pelo ATM Forum, com intuito de agilizar a implantação de redes ATM interoperáveis. Os padrões definidos pelo ATM Forum tem sido retificados de forma quase integral pelo ITU-T [18].

As classes de serviço padronizadas no ATM Forum são CBR(Constant bit rate), VBR(Variable bit rate), ABR(Available bit rate) e UBR(Unspecified bit rate), [36], além de diferentes camadas de adaptação AAL(s) [8]. Também foram padronizados pelo ATM Forum os chamados descritores de tráfego UPC. Os descritores são parâmetros que visam descrever o comportamento estatístico do tráfego que vai ser gerado para a rede, de forma a estabelecer um contrato entre rede e usuário. Os descritores UPC padronizados são: PCR(Peak cell rate) ou taxa de pico, SCR(Sustainable cell rate) ou taxa média, MBS(Maximum burst size) ou tamanho máximo da rajada e CDVT(Cell delay variation tolerance) ou tolerância na geração de células.

Devido a variabilidade estatística dos diversos tráfegos e da existência de requisitos de QoS bem distintos, tarefas como o controle de congestionamento são extremamente importantes para proteger a rede e o usuário, além de otimizar os recursos. O controle de congestionamento pode ser classificado como controle *preventivo* e controle *reativo*. No esquema de controle preventivo, o procedimento de verificação é executado no estabelecimento da conexão. No esquema de controle reativo, confia-se em informações de retorno para controlar o nível de congestionamento. Cada abordagem tem vantagens e desvantagens. Em redes ATM, a combinação destas duas abordagens é usada para prover controle de congestionamento efetivo. Classes de tráfego CBR e VBR usam esquemas preventivos e classes ABR baseiam-se em um esquema reativo.

Quando uma nova requisição de conexão é recebida na rede, o procedimento de admissão de chamada é executado para decidir se a conexão deve ser aceita ou rejeitada. A chamada é aceita se a rede possui recursos suficientes para prover os requisitos de QoS da conexão requisitada [27], sem afetar a QoS das conexões já existentes. Em resumo, o objetivo é maximizar a utilização dos recursos disponíveis, sem comprometer a QoS já oferecida aos usuários. O algoritmo que toma a decisão de aceitar ou não uma nova conexão mediante as condições da rede e necessidades do usuário, é chamado de *controle de admissão de conexões* (CAC). O CAC é um esquema de controle de congestionamento preventivo, que tenta prevenir a queda de desempenho.

Para exemplificar, consideremos um switch ATM não-bloqueante com filas nas portas de saída da Figura 1.1 . Neste exemplo, temos células que chegam através de várias portas de entrada e que devem ser encaminhadas para a porta de saída 1. Neste tipo de arquitetura, o ponto de congestionamento ocorre nas portas de saída, pois as células são armazenadas nas filas de saída, antes de serem transmitidas. Portanto, o algoritmo de CAC leva em consideração o buffer de cada porta de saída. Se nós isolarmos uma porta de saída (com seu respectivo buffer) do resto do switch, nós obtemos o modelo de fila mostrado na figura 1.2. Este tipo de modelo de fila representa um número de fontes acessando um buffer de capacidade finita servido a uma taxa constante.

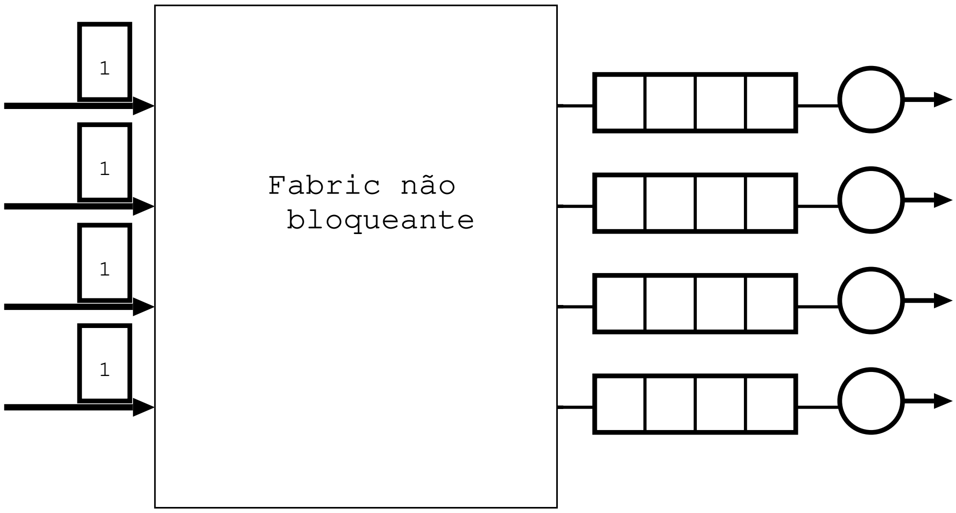


Figura 1.1: Switch ATM com bufferização na saída

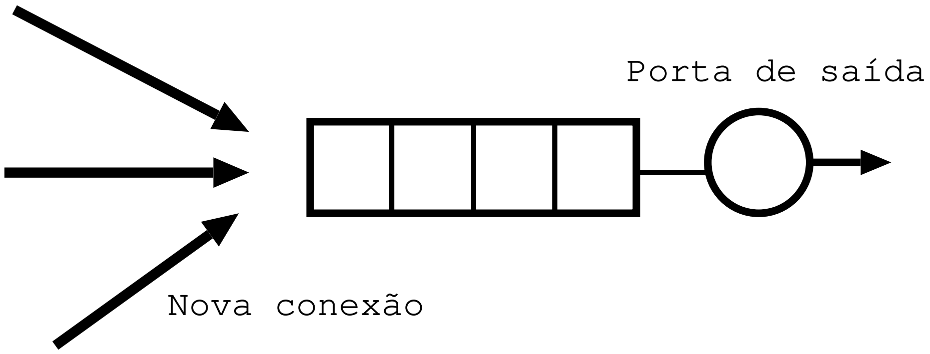


Figura 1.2: Conexões passantes em uma porta de saída

É importante notar que o algoritmo de CAC depende da arquitetura do comutador ATM. No exemplo da Figura 1.1, o algoritmo atua em cada porta de saída, visto que nesta arquitetura não existem perdas na comutação, mas pode ocorrer perda de células por falta de espaço no buffer de saída. Se os buffer(s) estiverem localizados na entrada do comutador ATM, o algoritmo deve atuar de outra forma, pois células endereçadas para diferentes saídas podem estar no mesmo buffer.

Podemos enumerar alguns requisitos básicos para um algoritmo de controle de admissão:

- *Eficiência de banda e QoS* - Melhorias na eficiência de banda podem significar serviços mais baratos. Uma das funções do controle de admissão é manter alta eficiência no compartilhamento da banda sem prejudicar a QoS negociada com o usuário.
- *Complexidade de Implementação* - Visto que o controle de admissão é um procedimento de controle de tráfego em tempo-real, é necessário reduzir o volume de processamento. O algoritmo de CAC deve ser simples o suficiente para ser executado em tempo-real.
- *Escalabilidade* - Se a complexidade no tempo de execução do algoritmo de CAC cresce em função do número de conexões, provavelmente este algoritmo não deve tomar a decisão em tempo suficiente, tornando-se inviável para decisões em tempo-real.
- *Dependência do modelo de tráfego* - Dependência do modelo de tráfego deve ser reduzida, isto é, o algoritmo deve ser robusto independentemente das características do tráfego.

Várias abordagens de algoritmos de controle de admissão foram propostas na literatura. Algumas abordagens obtêm expressões baseadas em modelos de tráfego markovianos [17, 15], outras baseiam-se em descritores UPC padronizados [31, 23, 13], entre outras. Neste trabalho nosso maior interesse são algoritmos de CAC com boa aplicabilidade. Selecionamos 2 algoritmos através dos seguintes critérios: facilidade de implementação e baseados em parâmetros UPC. Onde facilidade de implementação significa que os parâmetros utilizados no algoritmo de CAC devem ser facilmente estimados.

O objetivo deste trabalho é analisar algoritmos de CAC propostos na literatura e compará-los através de testes realizados sobre um comutador real. Para isso, selecionamos e implementamos dois algoritmos de CAC, um para tráfego não-regulado, proposto em [17] e outro para tráfego regulado pelo leaky bucket, proposto em [31]. Comparamos esses algoritmos usando o comutador ATM da Fore-Systems ASX-1000, disponível em nosso laboratório, e analisamos o comportamento deste algoritmo em relação a parâmetros que podem ser escolhidos pela aplicação. Para a realização de testes, desenvolvemos um gerador de tráfego com suporte IP e ATM-Nativo.

Este trabalho está organizado da seguinte forma: No capítulo 2 apresentamos alguns dos principais algoritmos de controle de admissão. No capítulo 3 discutimos as principais funcionalidades das ferramentas desenvolvidas. Estas ferramentas são de uso geral e fazem parte do conjunto de ferramentas Tangram-II [7, 35]. No capítulo 4 apresentamos um estudo do comportamento do comutador ATM em relação a parâmetros configuráveis chamados de overbooking. No capítulo 5 apresentamos um conjunto de medições realizadas no comutador ATM. Finalmente no capítulo 6, são feitas considerações finais sobre o trabalho sugerindo temas para investigações futuras. No apêndice A, apresentamos um exemplo passo a passo de como encontrar expressões de *capacidade efetiva* baseado em [36]. A *capacidade efetiva* é uma expressão para alocação de capacidade. Esta capacidade deve garantir a QoS de um conjunto de conexões em um enlace ATM.

Capítulo 2

Algoritmos de Controle de Admissão de Conexões

MUITOS métodos de controle de admissão (CAC) têm sido propostos na literatura. Alguns autores dividem os métodos de CAC em métodos baseados em descritores de tráfego e métodos baseados em medições [22, 24, 16]. Outros preferem dividir os métodos de CAC classificando-os pelas diferentes aproximações da probabilidade de perda e capacidade efetiva [33]. Existem diversos algoritmos de CAC baseados em modelos de tráfego auto-similares [39] e para diferentes políticas de escalonamento [34]. Algumas propostas são baseadas em modelos de fluido [19] e outras estimam a capacidade efetiva agregada a partir de filtros [14], entre outros.

A maioria destes algoritmos não têm sido implementados, porque os parâmetros necessários para alguns dos métodos não estão padronizados, ou os cálculos são muito complexos para serem incluídos em um equipamento real.

Por outro lado, sabemos que os descritores de tráfego padronizados pelo ITU-T, UPC(Usage Parameter Control) (pico, média, mbs) nem sempre capturam as características estatísticas relevantes de um tráfego multimídia. Entretanto, o enfoque que resolvemos adotar está baseado em parâmetros UPC, pois nosso objetivo é analisar algoritmos que usam os parâmetros padronizados e compará-los utilizando medições em um equipamento de comutação ATM real. Desta forma, selecionamos dois entre os principais algoritmos de controle de admissão com boa aplicabilidade.

Neste capítulo abordamos alguns dos principais algoritmos de controle de admissão propostos na literatura. Na seção 2.1, apresentamos uma proposta de estimativa estatística para a *capacidade efetiva* baseada em um modelo de fluido [17]. Na seção 2.2, apresentamos um método de controle de admissão para tráfegos regulados pelo leaky bucket, baseado em aproximações da probabilidade de perda [31].

2.1 Capacidade Efetiva Baseada em Modelos de Fluido

O modelo utilizado para esta abordagem é um modelo de fluido onde a taxa de bits gerada por unidade de tempo é representada como um fluxo contínuo de bits com intensidade variando de acordo com o estado de uma cadeia de Markov. Esta cadeia de Markov de tempo contínuo é obtida pela superposição de fontes associadas a cada conexão.

No caso de N fontes on-off(s) idênticas (cada uma com uma taxa de pico R) a intensidade do fluxo agregado de chegada é jR, quando j fontes estão ativas. O fluxo agregado é fornecido a um buffer que é esvaziado a um taxa constante c.

O objetivo é determinar o menor valor de taxa de serviço $\tilde{C}$, para um dado tamanho de buffer x, tal que a *probabilidade de buffer overflow* seja menor que ϵ. Onde $\tilde{C}$ é a *capacidade efetiva* das conexões multiplexadas.

Para encontrar uma expressão para a *capacidade efetiva* é necessário achar a distribuição do conteúdo do *buffer* em função das características das conexões e da taxa de serviço. A partir desta expressão, o valor da menor taxa de serviço que garanta a *probabilidade de perda* para o tamanho existente do *buffer* é obtida. Esta taxa de serviço chamada de *capacidade efetiva* é a capacidade que deve ser alocada para as conexões. A distribuição do conteúdo do buffer pode ser derivada utilizando-se métodos existentes para fontes idênticas [30], e outros métodos para cadeias de Markov mais genéricas [29, 21]. Embora estes métodos forneçam as informações necessárias para determinar a *capacidade efetiva*, a complexidade computacional não é compatível com as necessidades de decisões em *tempo-real*. Portanto, aproximações devem ser utilizadas.

Considere o modelo de uma única fonte Markoviana on-off, ilustrada na Figura 2.1.

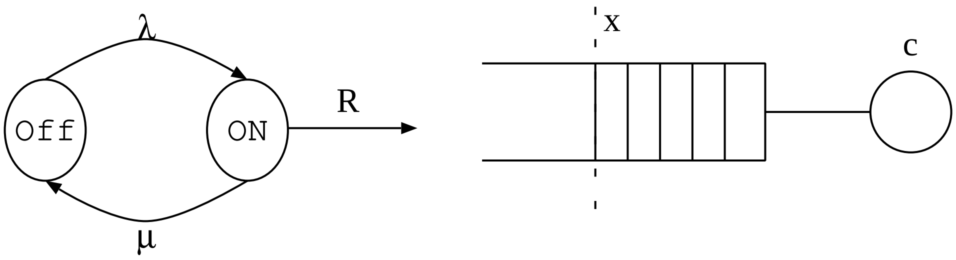


Figura 2.1: Modelo de fonte on-off usado no cálculo da *Capacidade Efetiva*

Seja a fonte on-off da Figura 2.1 caracterizada pelos seguintes parâmetros:

- taxa de pico R (células/s);
- utilização ρ;
- tamanho médio do burst b (células).

Então as seguintes relações são válidas:

$$\mu = \frac{1}{b} \quad \lambda = \frac{\rho}{b(1-\rho)} \tag{2.1}$$

de forma a obter o casamento do modelo com os parâmetros fornecidos acima.

O sistema de equações que iremos apresentar é um caso particular do trabalho fundamental de Anick et al [30], que utiliza a teoria de análise espectral para estudar um sistema composto por N fontes *on-off* idênticas e independentes, com um servidor de capacidade constante C e *buffer* infinito.

Seja $P_i(t, x)$, a probabilidade de que no tempo t, i fontes se encontrem no estado *on* e o conteúdo do *buffer* não exceda x. O desenvolvimento que se segue é para uma única fonte e desta forma temos:

$$P_0(t + \Delta t, x) = P_0(t, x + C\Delta t)(1 - \lambda\Delta t) + P_1(t, x - R\Delta t + C\Delta t)(\mu\Delta t) \qquad (2.2)$$

e

$$P_1(t + \Delta t, x) = P_0(t, x + C\Delta t)(\lambda\Delta t) + P_1(t, x - R\Delta t + C\Delta t)(1 - \mu\Delta t) \qquad (2.3)$$

Subtraindo ambos os lados da equação (2.2) por $P_0(t, x)$, dividindo por Δt e fazendo $\Delta t \to 0$, temos:

$$\tfrac{dP_0(t,x)}{dt} - C\tfrac{dP_0(t,x)}{dx} = -\lambda P_0(t, x) + \mu P_1(t, x) \qquad (2.4)$$

Subtraindo ambos os lados da equação (2.3) por $P_1(t, x)$, dividindo por Δt e fazendo $\Delta t \to 0$, temos:

$$\tfrac{dP_1(t,x)}{dt} - (C - R)\tfrac{dP_1(t,x)}{dx} = \lambda P_0(t, x) - \mu P_1(t, x) \qquad (2.5)$$

As medidas de interesse, são as probabilidades em estado estacionário. Logo:

$$\frac{dP_i(t, x)}{dt} = 0 \qquad (2.6)$$

Seja a *probabilidade em estado estacionário* da fonte estar ativa (*on*, i=1) ou inativa (*off*, i=0) e o conteúdo do buffer ser $\leq x$:

$$F_i(x) \triangleq lim_{t\to\infty} P_i(t, x) \quad , i = 0, 1 \qquad (2.7)$$

Desta forma, as equações (2.4) e (2.5) podem ser escritas, em notação matricial como:

$$\mathbf{D}\tfrac{d\mathbf{F}(x)}{dx} = \mathbf{QF}(x) \quad , x \geq 0 \qquad (2.8)$$

Onde,

$$D = \begin{bmatrix} \text{c} & 0 \\ 0 & \text{(R - c)} \end{bmatrix}$$

$$Q = \begin{bmatrix} -\lambda & \mu \\ \lambda & -\mu \end{bmatrix}$$

e λ é a taxa de transição de off para on e μ é a taxa de transição de on para off.

Seja $G(x)$ a probabilidade do conteúdo do buffer sobrepassar x, *probabilidade de overflow* .

$$G(x) \triangleq P(buffer > x) = 1 - F(x) \quad , x \geq 0 \tag{2.9}$$

A solução para a equação (2.8) é dada por (2.10), onde z_i e Φ_i são os autovalores e autovetores respectivamente de $\mathbf{D^{-1}Q}$, e os coeficientes a_i dependem dos autovetores e condições iniciais.

$$F(x) = \textstyle\sum_i a_i \Phi_i e^{z_i x} \quad , x \geq 0 \tag{2.10}$$

O maior autovalor negativo dos z_i e o correspondente autovetor determinam a expressão para a *probabilidade de overflow* ϵ. Para um tamanho de buffer finito x, é mostrado em [17] que:

$$\epsilon = \beta e^{-\frac{x(c-\rho R)}{cb(1-\rho)(R-c)}} \tag{2.11}$$

onde

$$\beta = \tfrac{(c-\rho R)+(\epsilon\rho(R-c))}{(1-\rho)c} \tag{2.12}$$

A *capacidade efetiva* $\tilde{c}$ é obtida isolando c em (2.11). Mesmo para o caso de uma única fonte, nenhuma expressão explícita é obtida para a *capacidade efetiva*, e (2.11) deve ser resolvida numericamente. Uma simplificação natural é observar o termo β. Pode ser mostrado que o termo é próximo a 1 (de fato, sempre menor). Se β for aproximado para 1, provê um limite superior explícito para $\tilde{c}$. A *capacidade efetiva* associada a uma única conexão é dada por:

$$\tilde{c} = \frac{\alpha b(1-\rho)R - x + \sqrt{[\alpha b(1-\rho)R - x]^2 + 4x\alpha b\rho(1-\rho)R}}{2\alpha b(1-\rho)} \tag{2.13}$$

onde $\alpha = \ln(1/\epsilon)$. Aplicando os limites quando $\rho = 1$ e $b = \infty$ resulta no valor esperado de $\tilde{c} = R$.

Uma maneira para encontrar a distribuição $F(x)$ é considerar uma aproximação assintótica. O sistema para uma aproximação assintótica tem um buffer "infinito", onde a *probabilidade de buffer overflow* é equivalente a probabilidade do conteúdo do buffer exceder o tamanho atual do buffer. Para grandes buffers, podemos obter uma aproximação considerando apenas a contribuição do termo correspondente ao maior autovalor negativo, ou autovalor dominante [30] (os outros termos ficam desprezíveis quando o tamanho do buffer aumenta). Dentro destas condições, a *probabilidade de buffer overflow* $G(x)$ tem a seguinte forma, [30]:

$$G(x) = \beta e^{z_o x} \tag{2.14}$$

onde z_o é o maior autovalor negativo, e β é um termo constante independente de x.

No caso de N fontes on-off(s) idênticas, temos:

$$G(x) = (\frac{N\rho R}{C})^N [\prod_{i=1}^{N-[C/R]-1} \frac{z_i}{z_i + z_o}] e^{-\frac{N(C-N\rho R)}{b(1-\rho)(NR-C)C}x} \tag{2.15}$$

onde z_i são todos os autovalores negativos, z_o é o maior autovalor negativo, C é a capacidade alocada para as N conexões.

Pela equação (2.15), pode-se notar que a expressão para β é complexa. Uma aproximação consistente com o caso de uma única fonte, é aproximar β para 1. Embora o valor real de β infelizmente nem sempre é próximo a 1, ainda assim, esta aproximação é razoável [30]. Entretanto, isto resulta em uma super-estimativa da *capacidade efetiva*.

Fazendo $\beta = 1$ em (2.14) e utilizando a *probabilidade de buffer overflow* ϵ , temos:

$$\epsilon = e^{z_o x} = z_o = -\frac{\alpha}{x} \tag{2.16}$$

onde $\alpha = \ln(1/\epsilon)$.

No caso de fontes on-off(s) (não necessariamente idênticas) existe uma relação entre o autovalor dominante z_o e a capacidade alocada, dada por:

$$C = \frac{\sum_{i=1}^{N}[(\lambda_i+\mu_i+R_i z_o)-\sqrt{(\lambda_i+\mu_i+R_i z_o)^2-4\lambda_i R_i z_o}]}{2z_o} \tag{2.17}$$

onde $1/\lambda_i$ e $1/\mu i$ são os tamanhos médios dos períodos de silêncio e rajada respectivamente da fonte i.

O valor da *capacidade efetiva* $\tilde{C}$ de N fontes multiplexadas, combinadas com a equação (2.16) e (2.17), é dada por:

$$\tilde{C} = \sum_{i=1}^{N} \tilde{c}_i \tag{2.18}$$

onde cada $\tilde{c}_i$ é determinado pela equação (2.13) e as relações (2.1) também foram usadas.

Uma das maiores vantagens de (2.18) é a simplicidade computacional. A linearidade de (2.18) certamente simplifica a alocação da banda para as conexões, mas também identifica claramente a limitação da expressão. A simplificação de $\beta = 1$ ignora o efeito da multiplexação estatística. Em particular, a menos que as *capacidades efetivas* das conexões individuais sejam próximas às taxas médias, a soma delas é tipicamente uma super-estimativa da *capacidade efetiva*.

Sendo assim, uma outra aproximação é proposta para melhorar a qualidade da expressão para levar em consideração a multiplexação estatística.

Neste caso, a distribuição estacionária da taxa de bits é aproximada pela distribuição Gaussiana.

A Figura 2.2 mostra a região de sobrecarga da distribuição gaussiana.

Esta aproximação é dada por:

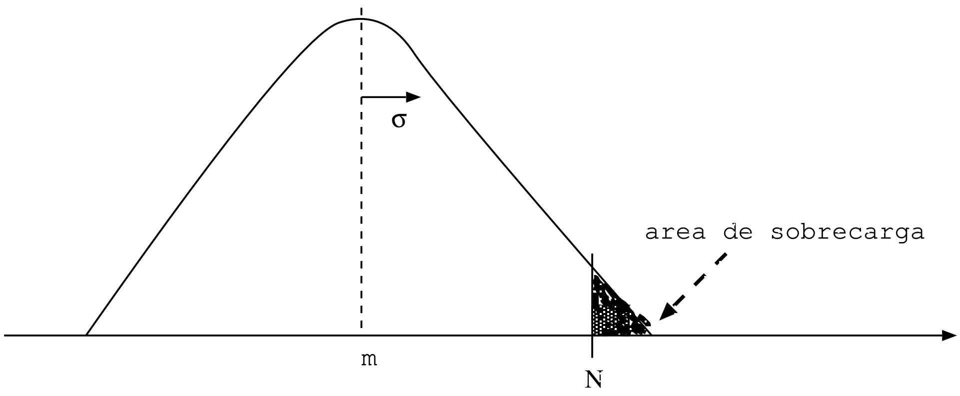


Figura 2.2: Cauda da distribuição gaussiana

$$\tilde{C} = [m + \alpha'\sigma] \tag{2.19}$$

onde $\alpha' = \sqrt{-2\ln(\epsilon) - \ln(2\pi)}$, m é a taxa média agregada $m = \sum_{i=1}^{N} \mathbf{m_i}$ e σ é o desvio padrão da taxa agregada $\sigma^2 = \sum_{i=1}^{N} \sigma_i^2$.

A combinação da aproximação de fluxo (2.13) com a aproximação estacionária (2.19) em uma única expressão, fornece a *capacidade efetiva* de um conjunto de conexões. Como ambas as aproximações super-estimam a *capacidade efetiva* (aproximação de fluxo ignora o efeito "smoothing" da multiplexação estatística, e a aproximação estacionária não leva em conta o tamanho do buffer) e não são totalmente precisas para diferentes características das conexões, a *capacidade efetiva* será o mínimo das duas abordagens.

$$\tilde{C} = min[m + \alpha'\sigma \quad , \sum_{i=1}^{N} \tilde{c}_i] \tag{2.20}$$

onde N é o número de conexões.

Esta abordagem tem boa aplicabilidade, devido às fontes serem facilmente mapeadas sobre os descritores de tráfego padronizados.

O algoritmo desta abordagem, funciona da seguinte forma:

Para cada nova fonte de tráfego descrita pelos parâmetros pico R, taxa de off-on

λ e taxa de on-off μ, calcule:

- A *Capacidade efetiva* $\tilde{c}_i$ da nova fonte i, para um dado tamanho de buffer x e QoS ϵ, usando a equação (2.13). Adicione $\tilde{c}_i$ à nova *capacidade efetiva agregada* $\tilde{C}_e$;

- A média m_i e o desvio padrão σ_i^2 da conexão i. Adicione m_i em m e σ_i^2 em σ^2. Compute usando a equação (2.19), a nova *capacidade efetiva agregada gaussiana* $\tilde{C}_g$;

- A nova *capacidade efetiva* $\tilde{C} = min[\tilde{C}_e, \tilde{C}_g]$. Se $\tilde{C} \leq C$ a nova conexão é admitida, caso contrário é rejeitada;

2.2 Algoritmo de Controle de Admissão para Tráfegos Heterogêneos Regulados

Nesta abordagem, as fontes de tráfego são submetidas ao mecanismo leaky bucket [6] de moldagem ou regulagem de tráfego.

Desta forma é possível determinar precisamente os limites superior e médio da taxa de bits gerada por uma fonte regulada, além do tamanho máximo da rajada. Sabendo que a faixa de variação de uma fonte regulada é menor que uma fonte comum, é possível tomar decisões mais precisas sobre a alocação de recursos.

O objetivo deste algoritmo de admissão é obter um esquema que utilize os parâmetros padronizados (sabendo que não são os melhores descritores para todos os casos) para estimar uma probabilidade de perda agregada.

Considere o modelo de N fontes reguladas pelo leaky bucket indicado na Figura 2.3. A curva de chegada de cada conexão j é limitada por $\alpha(t) = min(R_j t, m_j t + b_j)$, onde R_j é a taxa de pico (células/s), m_j é a taxa média (células/s) e b_j (células) é o tamanho máximo da rajada (número máximo de fichas no leaky-bucket).

No sistema ilustrado pela Figura 2.3, podemos obter expressões para o tamanho

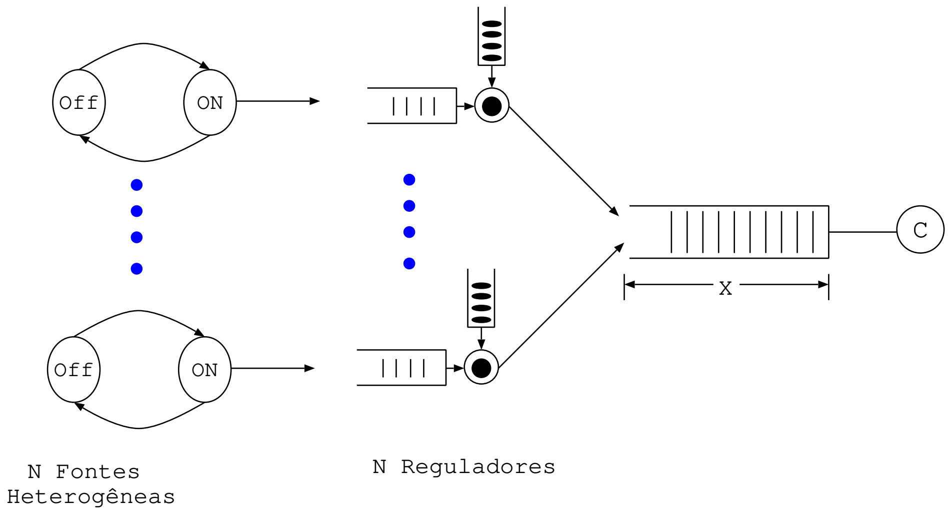


Figura 2.3: Modelo de tráfego on-off(s) regulados acessando um *buffer* de tamanho X com capacidade de transmissão C

médio de um período ativo T_{ONj} e tamanho médio de um período inativo T_{OFFj} de cada conexão j. Observando o processo de saída de cada regulador, assumindo que as fases de cada conexão são estatisticamente independentes, temos [31]:

$$T_{ONj} = \frac{b_j}{R_j - m_j} \tag{2.21}$$

$$T_{OFFj} = \frac{b_j}{m_j} \tag{2.22}$$

A idéia do CAC proposto por [31] consiste em alocar uma fração c da banda total disponível C e uma fração x do buffer total disponível X, de tal forma que as alocações sejam proporcionais aos recursos do nodo. Se todas as fontes são conhecidas e são heterogêneas, encontrar valores para a banda que produzam a menor quantidade de buffer estimada, sem exceder as restrições de banda total, envolve uma otimização complicada. Uma otimização do gerenciamento dos recursos, ocorre quando a alocação de recursos compartilhados da rede é equivalentemente dividida. Para cada VC uma fração de banda e buffer é reservada, sendo que a fração de buffer reservada depende do tamanho total do buffer X, a capacidade reservada para o VC depende do tamanho total da capacidade total de transmissão C no enlace ATM.

A equação (2.23) mostra que a fração de capacidade reservada deve ser equivalente a fração do buffer reservado. Esta expressão representa a primeira equação de reta linear da Figura 2.4 e também pode ser interpretada como $x/c = X/C$ significando que a fração de reserva de recursos de um VC deve obedecer a fração de recursos totais de um nodo ATM.

Desta forma, em [31] adotou-se:

$$\frac{c}{C} = \frac{x}{X} \tag{2.23}$$

Na primeira parte da proposta, expressões são obtidas para o caso onde os requisitos de QoS são de *multiplexação com perda zero* .

Seja c_{0j} a *capacidade efetiva com perda zero da conexão* j e seja x_{0j} o *buffer efetivo com perda zero da conexão* j. Supondo que todas as N conexões são idênticas, desejamos obter expressões para c_0 e x_0 que levam em consideração a fração de recurso do nodo X/C. Conforme a fração de recurso X/C aumenta, as expressões podem ser mais otimistas.

A Figura 2.4 mostra a coordenada (c_0,x_0), como a interseção de duas retas lineares.

A condição de existência da interseção entre as duas retas, e como consequência a existência do ponto (c_0,x_0) é se e somente se:

$$\frac{C}{X} > \frac{m}{b} \tag{2.24}$$

Note na Figura 2.4 que x decresce linearmente com o aumento de c para $m \leq c \leq R$, e os pontos extremos de (c,x) são (m,b) e $(R,0)$.

A expressão encontrada para *buffer* x, representando a segunda equação linear segundo [31] é:

$$x = b - \frac{b(c-m)}{R-m} \tag{2.25}$$

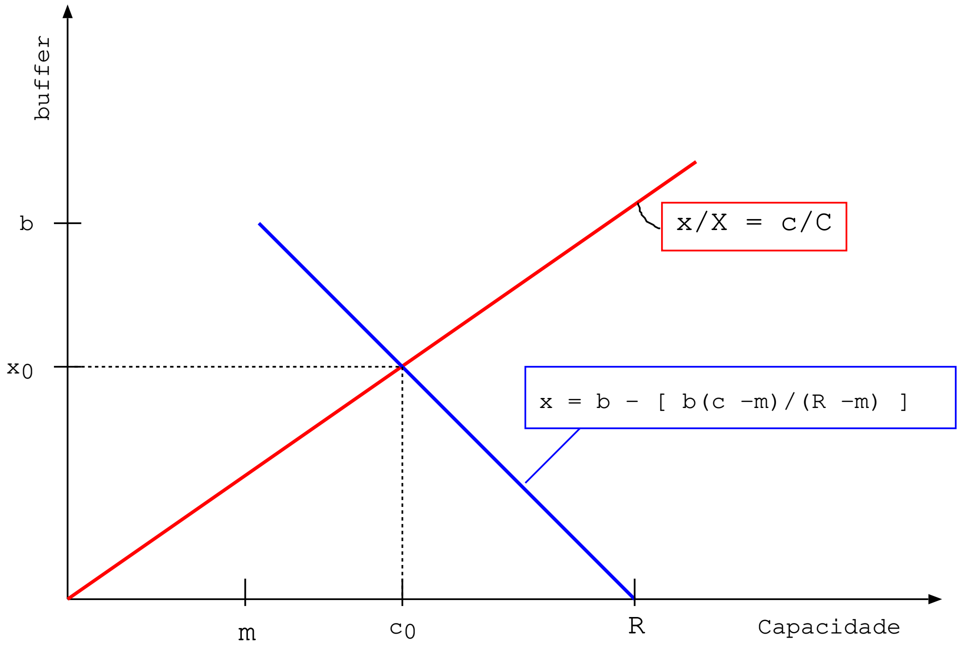


Figura 2.4: Obtendo a alocação da *banda* e *buffer* (c_0, x_0) respectivamente

Pela interseção entre as duas equações de retas lineares (2.23) e (2.25), obtêm-se:

$$c_0 = \begin{cases} \frac{R}{1+\frac{X/C}{b}(R-m)} & \text{, se } m < b\frac{C}{X} \\ m & \text{, caso contrário} \end{cases} \tag{2.26}$$

Para o caso da alocação do *buffer* temos as seguintes opções:

Se a margem de violação da equação (2.24) é pequena, a sugestão é $x_0 = mX/C$, ou $x_0 = b$ caso contrário.

A segunda parte da proposta analisa o sistema *multiplexando estatisticamente* e permite pequenas *probabilidades de perda.*

Para prover uma estimativa para a *probabilidade de perda*, o limite de Chernoff (Chernoff's bound) utilizando refinamentos assintóticos da teoria dos grandes desvios é aplicado ao sistema. Por definição:

$$P_{loss} \leq e^{-F_K(s^*)} \tag{2.27}$$

onde,

$$F_K(s) = [sC - \sum_{j=1}^{N} K_j \ln[1 - w_j + w_j e^{sc_{0,j}}]] \tag{2.28}$$

O parâmetro K_j é o número de fontes que podem ser admitidas, s^* é o ponto onde a função F_K é maximizada. O parâmetro w_j é a fração do tempo que o sistema está ocupado, dado por $w_j = m/c_{0j}$.

O parâmetro s^* é obtido resolvendo a seguinte equação, [31]:

$$C = \sum_{j=1}^{N} \frac{K_j w_j c_{0,j} e^{sc_{0,j}}}{1 - w_j + w_j e^{sc_{0,j}}} \tag{2.29}$$

No caso de um único tipo de tráfego, ou seja, $N = 1$, temos as seguintes expressões com $a = (C/c_{0j})/K$

$$s^* = \frac{1}{c_{0j}} \ln[\frac{a}{1-a}.\frac{1-w}{w}] \tag{2.30}$$

$$F_K(s^*) = K[a \ln(\frac{a}{w}) + (1-a) \ln(\frac{1-a}{1-w})] \tag{2.31}$$

A expressão acima pode ser usada para obter K_{max}, que é o valor de K no qual

$$F_K(s^*) = \ln(1/\epsilon) \tag{2.32}$$

onde ϵ é a probabilidade de perda desejada.

A expressão para o tipo de tráfego j da *capacidade efetiva estatisticamente multiplexada* $\tilde{c}_j$ é:

$$\tilde{c}_j = \frac{\ln[1 - \frac{w_j}{c_{0,j}} + \frac{w_j}{c_{0,j}} e^{s^* c_{0,j}}]}{s^* + (\ln(\epsilon))/C} \tag{2.33}$$

Um refinamento na expressão de limite superior para probabilidade de perda P_{loss}, a partir de aproximações de teoria dos grandes desvios, é ainda apresentado na abordagem. A aproximação refinada é:

$$P_{loss} = \frac{e^{-F_K(s^*)}}{s^*\sigma(s^*)\sqrt{2\pi}} \tag{2.34}$$

onde,

$$\sigma(s^*) = \sqrt{\sum_{j=1}^{N} \frac{(1-w_j)w_j c_{0,j}^2 exp(s^* c_{0,j})}{(1-w_j+w_j exp(s^* c_{0,j}))^2}} \tag{2.35}$$

Analisando a qualidade das expressões para um único tipo de tráfego, temos o seguinte:

Considerando o cálculo de K_{max}, o maior valor de K para o qual $F_k(s^*) \leq \ln(1/\epsilon)$ indica o número total de fontes admitidas. Note que:

Se $\ln(1/\epsilon) < C/c_0 \ln(1/w)$, no máximo de $F_k(s^*)$, então $K_{max} > C/c_0$. Desta forma o ganho com a multiplexação estatística $g = c_0/\tilde{c} > 1$, visto que $\tilde{c} = C/K_{max}$.

Por outro lado, se $\ln(1/\epsilon) \geq C/c_0 \ln(1/w)$, então não existe solução para $F_k(s^*)$. Entretanto, sabemos que, $K = C/c_0$ e $P_{loss} = 0$, assim $K_{max} = C/c_0$. Neste caso $g = 1$.

Para o caso de vários tipos de tráfego:

Seja a *banda crítica* C_c a menor quantidade de capacidade alocada, tal que o ganho com a multiplexação estatística $g > 1$. Isto é válido para todo C_c tal que $C > C_c$. A prova para esta proposição pode ser encontrada em [31].

A expressão para C_c é:

$$\mathbf{C_c} = \frac{c_0 \ln(1/\epsilon)}{\ln(1/w)} \tag{2.36}$$

Neste procedimento de alocação de recurso, note que se c_0 e w dependem da capacidade de recurso do nodo X e C somente através de X/C, (2.26). Isto também é verdade para C_c.

No caso de um modelo com dois tipos de tráfego distintos, outra propriedade importante da definição de *banda crítica* é:

- Se $C < min(C_{c,1}, C_{c,2})$ podemos classificar os tipos de tráfego como NS-VBR

(nonstatistically-multiplexable VBR).

- Se $C > max(C_{c,1}, C_{c,2})$ podemos classificar os tipos de tráfego como S-VBR (statistically-multiplexable VBR).

- Se $C_{c,1} < C < C_{c,2}$ tipo 1 é S-VBR e tipo 2 é NS-VBR.

O *algoritmo de controle de admissão* desta abordagem, para um único tipo de tráfego $N = 1$, QoS ϵ, buffer do nodo definido como X e capacidade de transmissão do nodo definida como C, funciona da seguinte forma:

Se não existir solução para $F_K(s^*)$, isto é, se $\ln(1/\epsilon) \geq C/c_0 \ln(1/w)$, a *capacidade efetiva com perda zero* c_0, é calculada usando a equação (2.26) e o número de fontes admitidas $K = C/c_0$ onde $P_{loss} = 0$.

Caso exista solução, para cada novo tipo de conexão, descrita pelos parâmetros pico R_j, média m_j e tamanho máximo da rajada b_j, compute:

- A *capacidade efetiva com perda zero* c_0, usando a equação (2.26).

- Se existe solução para $F_K(s^*)$, isto é, se $\ln(1/\epsilon) < C/c_0 \ln(1/w)$, determinar o número de fontes K_{max} admitidas para a QoS ϵ. Enquanto a função $F_K(s^*)$ é menor que $\ln(1/\epsilon)$ o número de conexões admitidas K pode ser aumentado. A função $F_K(s^*)$ é obtida pela equação (2.31) e depende do parâmetro a. O parâmetro a varia em função do número de fontes K, implicando por consequência na variação de $F_k(s^*)$ a cada nova fonte adicionada. O ponto s^* é calculado usando a equação (2.30), a cada iteração de K, visto que também está relacionado com o parâmetro a.

- A *banda crítica* $C_{c,j}$ usando (2.36).

- A aproximação para a *probabilidade de perda* P_{loss} usando a equação (2.34) e (2.35), a partir de s^* anteriormente calculado.

- A *capacidade efetiva multiplexada estatisticamente* $\tilde{c}_j$ usando a equação (2.33) a partir de s^* anteriormente calculado.

Capítulo 3

Ferramentas Desenvolvidas

NESTE capítulo apresentamos as principais funcionalidades das ferramentas desenvolvidas para prover um ambiente de testes, além da validação dos algoritmos de admissão implementados.

Na seção 3.1, apresentamos as características principais do gerador de tráfego, a especificação de modelos de tráfego e modos de transmissão do gerador. Na seção 3.2, apresentamos o módulo de admissão, incluindo os algoritmos de admissão implementados. Este módulo simula o funcionamento de algoritmos de CAC de acordo com parâmetros fornecidos pelo usuário.

3.1 Gerador de Tráfego

Em um ambiente de rede de alto desempenho, existe a necessidade de determinar o desempenho de equipamentos isolados, como, por exemplo, o processo de perda de células em um comutador, ou a latência do comutador, ou até determinar o desempenho de serviços de rede.

Estamos particularmente interessados em medir perda de células, bem como gerar tráfego seguindo a descrição de um modelo de tráfego com melhor nível de precisão.

O *Gerador de Tráfego* desenvolvido é uma ferramenta flexível capaz de gerar pacotes para a rede em tamanho e intervalos distribuídos conforme especificação do usuário. O *Gerador de tráfego* foi implementado em C/C++ e o usuário tem a opção de enviar dados sobre IP ou sobre ATM nativo através da camada de adaptação AAL5.

No caso do gerador UDP/IP, fazemos uso das API(s) *socket BSD* padrão [37], sem maiores requisitos para a utilização. O gerador UDP/IP está disponível para os sistemas operacionais Solaris e Linux.

No caso específico do ATM nativo, a API de programação utilizada para o Sistema Operacional Solaris foi a API/XTI [38] disponibilizada pela FORE Systems Inc. Para o Linux, foi utilizada a API desenvolvida por Werner Almesberger, disponível em [1, 2], sendo esta API independente de fabricantes.

A ferramenta desenvolvida possui ajuda através da linha de comando, além da interface em Tangram-II [7, 35] descrita pela Figura 3.1, facilitando a especificação dos parâmetros necessários.

3.1.1 Formas de Enviar o Tráfego

Um problema crítico de implementação refere-se à geração precisa de pacotes nos instantes de transmissão. Outro fator importante é que o modelo de tráfego não é necessariamente determinístico, trazendo intervalos de tamanhos distintos para a aplicação.

Uma das alternativas de implementação com alto grau de precisão é a espera ocupada. Esta solução é interessante para sessões de pequena duração e quando o número de geradores de tráfego é único em cada estação [25]. Nesta abordagem o processador deve ser totalmente alocado ao processo de geração. Para sessões de maior duração, em um ambiente onde há necessidade de um número maior de geradores com vários processos concorrentes na mesma estação, esta solução torna-se inviável.

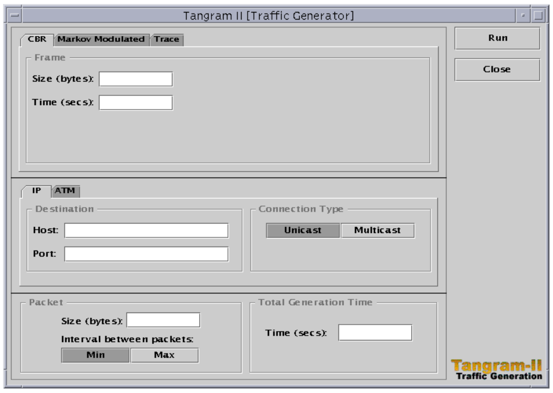
Outra alternativa considerada é a interface de sinais, padrão do sistema UNIX, disponibilizada na especificação POSIX.4. Um sinal associado a um processo faz com que o processo seja acordado em intervalos configuráveis. Neste caso, os intervalos entre cada geração não são constantes, e a restauração dinamica de sinais tem um custo adicional. Mesmo que a granularidade dos sinais esteja pouco abaixo do *usleep()* (10ms), variar dinâmicamente os tempos de um sinal de acordo com cada intervalo aleatório do modelo de tráfego, juntamente com o custo de restauração de um novo sinal, acaba inviabilizando esta solução.

Uma solução imediata é colocar o processo para dormir através da função *usleep()*. A precisão da função *usleep()* está diretamente ligada ao sistema operacional. A resolução da função *usleep()* em Linux para a plataforma i386 é próxima a 20ms. Entretanto, podemos ter uma melhoria na implementação. Toda vez que um processo envia pacotes, este torna-se ocioso, sendo que ao final de uma escrita bloqueante, buscamos o tempo do relógio do sistema e comparamos com o resíduo do intervalo, se o resíduo é positivo, realizamos uma chamada de *usleep()* colocando o processo para dormir durante o resíduo de tempo, caso contrário incrementamos o contador de intervalo sem dormir.

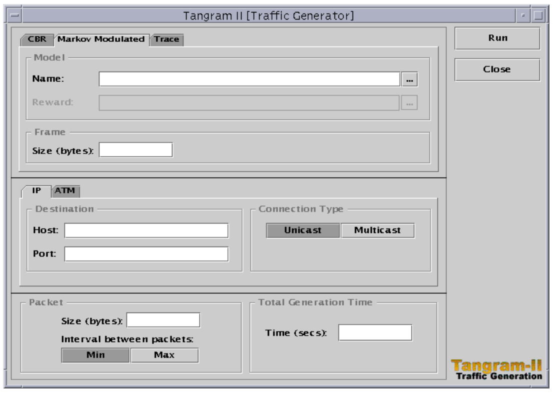
Para geração de taxas não tão altas, cerca de 20Mbps, sessões de curta ou longa duração, qualquer tipo de fontes com intervalos aleatórios ou determinísticos e até 2 geradores por estação, esta solução é viável. Para geração de taxas altas, sugerimos um kernel do linux com suporte para escalonamento de eventos em tempo real, onde a ordem de resolução de cada evento é microsegundos [3].

A partir da interface da Figura 3.1, ou pela linha de comando, o usuário pode especificar o tamanho dos pacotes L (opção *Packet - Size*), o tempo total de geração T (opção *Total Generation Time*), o intervalo entre gerações consecutivas de quadros I (depende do modelo de tráfego) e o número de bytes por quadro D (depende do modelo de tráfego). O número de pacotes por quadro N, depende do tamanho do quadro e do pacote $N = \frac{D}{L}$.

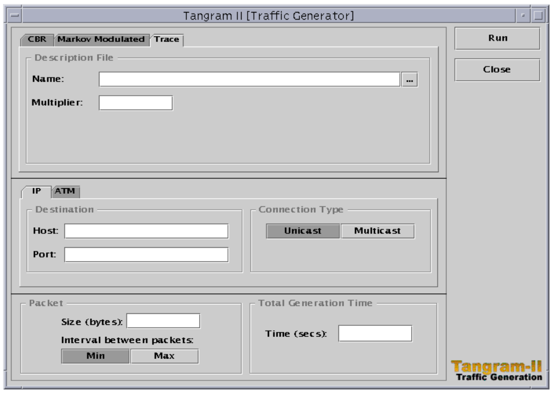
Nas telas da Figura 3.1 o usuário pode escolher o modelo de tráfego na parte superior e a tipo da rede (IP ou ATM). Nas opções de rede, para o UDP/IP, o usuário deve especificar o nome do host e uma porta de destino com uma opção de enviar para um único destino *Unicast* ou enviar para um grupo *Multicast*. Na opção ATM, o usuário deve especificar *VPi, VCi* e a *Traffic Class - UBR ou CBR*, sendo para CBR obrigatório o valor da taxa de pico em (opção *Peak rate* na Tela 4)



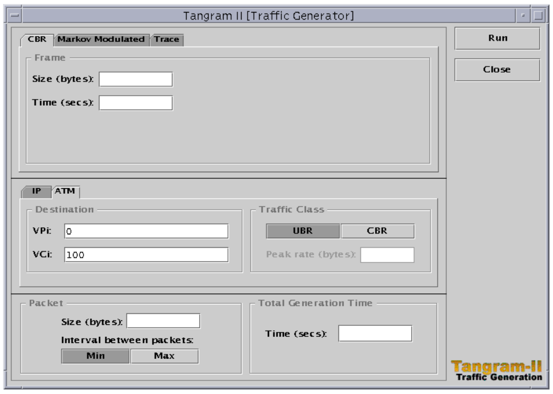
Tela 1



Tela 2



Tela 3



Tela 4

Figura 3.1: Interface do Tangram-II para a geração de tráfego

Em particular para o gerador ATM nativo, é possível estabelecer uma conexão com QoS. As placas das estações ATM em nosso laboratório, suportam VC(s) CBR(s) com o descritor de taxa de pico R e VC UBR sem descritor. O descritor de PCR(Peak Cell Rate) R, representa a taxa total em células ATM incluindo cabe-çalhos. O gerador ATM utiliza AAL5 [9] como camada de adaptação. O número de bytes por quadro D especificado pelo usuário é o PDU (Payload Data Unit) de cada frame AAL5.

Pacotes de um quadro podem ser transmitidos de 2 maneiras: gerados no início do intervalo ou espalhados de forma uniforme no intervalo. Estas 2 maneiras estão nas 4 telas da Figura 3.1 no canto inferior esquerdo de cada tela. Chamamos esta opção de *Interval between packets*, onde *min* significa que os pacotes serão gerados sequencialmente como indicado na Figura 3.2 e *max* significa que os pacotes serão transmitidos uniformemente como indicado na Figura 3.3.

A Figura 3.2 indica que os pacotes são transmitidos sequencialmente, sem o processo de geração dormir entre cada envio consecutivo. Apenas ao final do último pacote transmitido, buscamos o tempo do relógio do sistema e comparamos com o resíduo do intervalo, se o resíduo for positivo, realizamos uma chamada de *usleep()* colocando o processo para dormir durante o resíduo de tempo, caso contrário incrementamos o contador de intervalo sem dormir. Neste modo de transmissão, somente uma chamada de *usleep()* é realizada por quadro.

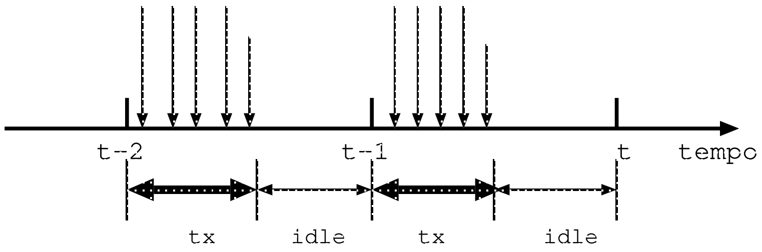


Figura 3.2: Modo de geração no início do intervalo

A Figura 3.3 indica que os pacotes são transmitidos uniformemente no intervalo, onde o processo de geração pode dormir entre cada envio consecutivo. Neste modo de transmissão, a cada envio de pacote uma chamada de *usleep()* é realizada. O número de chamadas por quadro dependem do número de pacotes por quadro N, anteriormente discutido.

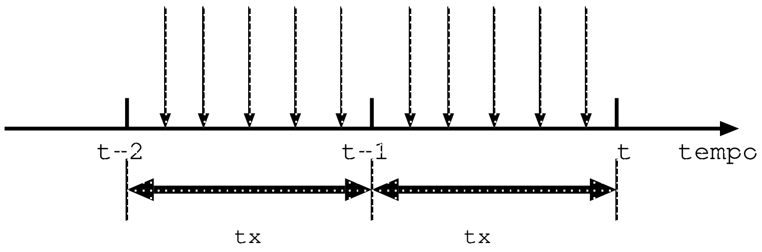


Figura 3.3: Modo de geração espalhando o tráfego uniformemente no intervalo

Atualmente a ferramenta encontra-se disponível para uso apenas sobre PVC's (Permanent Virtual Circuit), sendo toda a execução das chamadas de escrita feitas em modo síncrono (bloqueantes). Em testes realizados com chamadas não bloqueantes, verificamos que quando os buffers de escrita no S.O estão cheios, as chamadas de escritas retornam erro impedindo novas transmissões, (enquanto os buffers não se esvaziarem), implicando em perda de sincronismo com o tempo real. As chamadas bloqueantes aguardam pelo esvaziamento de forma transparente, facilitando o sincronismo com o tempo real.

3.1.2 Modelos de Tráfegos Suportados pelo Gerador

Na descrição de modelos de tráfego, provemos 3 "tipos" de tráfego. Todos podem ser especificados através da interface indicada na Figura 3.1. Os 3 modelos suportados são:

- Tráfego CBR(Contant Bit Rate) - Neste tipo de tráfego, o gerador envia quadros em intervalos fixos. O usuário deve especificar apenas a quantidade de bytes (opção *Frame - size*) a ser enviada a cada intervalo e o tamanho do intervalo (opção *Frame - Time*).

- Tráfego modelado por um cadeia de Markov - Neste opção, o tráfego a ser gerado, é modelado por uma cadeia de Markov de tempo contínuo e espaço de estados finito com recompensas. Desta forma, a recompensa r_j associada ao estado j é igual a taxa de transmissão em j. A especificação da cadeia pode ser feita usando-se a ferramenta de modelagem Tangram-II [7, 35]. O gerador discretiza a cadeia, obtendo o parâmetro Δ_j da seguinte forma:

$$\Delta_j = \sum_k \lambda_{k,j} \tag{3.1}$$

  , onde $\lambda_{k,j}$ é a k-ésima taxa de transição no estado j.

  As probabilidades de saída de cada estado j são calculadas como:

$$Prob_{j,k} = \sum_k \frac{\lambda_{k,j}}{\Delta_j} \tag{3.2}$$

, onde $Prob_{j,k}$ é a k-ésima probabilidade de saída do estado j.

Supondo o sistema estar no estado j, usando a taxa Δ_j calculada pela equação 3.1, geramos a variável aleatória I_j exponencialmente distribuída, e enviamos $r_j * I_j$ bytes no tempo I_j. Também sorteamos uma variável aleatória uniforme[0..1] e através das probabilidades da equação 3.2 determinamos a transição ou permanência em um estado j.

Para a especificação deste modelo, o usuário deve associar um arquivo contendo a matriz de transição de estados do modelo (Tela 2, opção *Model - Name*) e também deve associar um arquivo com a matriz de recompensas (Tela 2, opção *Model - Reward*). Os 2 arquivos podem ser gerados pelo ambiente de modelagem do Tangram-II.

- Tráfego tipo "Trace" - Neste caso o usuário fornece um arquivo (Tela 3, opção *Description File - Name*) contendo: instante de tempo e a quantidade de bytes a ser enviada no intervalo. O arquivo de trace pode ser obtido a partir do simulador do Tangram-II que pode gerar tráfego de diversas formas incluindo tráfegos com característica de dependência de longa duração FARIMA e FBM [28, 4]. Por outro lado, o arquivo de trace pode conter taxas obtidas de um sequência real, dando assim uma enorme flexibilidade ao usuário.

3.2 Módulo de Admissão

O módulo de admissão, foi implementado para criar um ambiente de testes, visando comparar diferentes algoritmos de CAC. Este módulo simula o funcionamento de algoritmos de CAC de acordo com parâmetros fornecidos pelo usuário.

Implementamos 2 algoritmos de CAC que permitem o usuário descrever as características do enlace ATM, através dos parâmetros de *Capacidade de Transmissão*, *Probabilidade de Perda Desejada* e *Tamanho do Buffer*. A partir da especificação do enlace ATM cada algoritmo calcula a *Capacidade efetiva* e o *número de fontes admitidas* para fontes especificadas através de descritores de tráfego. Cada método possui um conjunto de descritores de tráfego não necessariamente idênticos.

Em uma das formas de funcionamento do módulo de admissão, o usuário especifica as características do enlace ATM, selecionando um dos métodos discutidos no capítulo 2 e obtém a *capacidade efetiva* além do número de fontes que podem ser admitidas neste enlace. Desta forma, sabe-se quantas fontes devem ser geradas e qual o modelo de tráfego (a partir dos descritores), de forma a se obter a QoS desejada. Usando a ferramenta de geração de tráfego é possível simular estas fontes. Para coletar as medidas de interesse, por exemplo a razão de perda de células, utilizamos a MIB(Management Information Base) do comutador e acessamos estas informações através do protocolo de gerência SNMP(Simple Network Management Protocol).

O módulo de admissão está integrado à ferramenta de uso geral Tangram-II [7, 35]. A Figura 3.4 mostra a interface do módulo de admissão. Na parte superior de cada tela encontram-se os parâmetros de descrição do enlace ATM, sendo: *Capacidade de Transmissão*, *Probabilide de Perda Desejada* e *Tamanho do Buffer*. Na opção de *Regulated-Traffic Algorithm* o usuário deve entrar com os descritores pico, média e tamanho máximo da rajada (ou número de fichas do leaky-bucket). Na opção de *Non-Regulated Traffic Algorithm* o usuário deve entrar com os descritores pico, taxa de transição de on-off e off-on. Os algoritmos devolvem o número de fontes admitidas no enlace ATM e a *capacidade efetiva* de cada fonte.

O módulo de admissão também pode funcionar como um servidor que aguarda requisições de clientes em uma porta configurável. O módulo de geração de tráfego implementa um cliente que se comunica com o servidor através de um protocolo simples mostrado na Figura 3.5. Na implementação do módulo de admissão fazemos uso das API(s) *socket BSD* padrão [37], sem maiores requisitos para a utilização. A ferramenta de admissão está disponível para os sistemas operacionais Solaris e Linux.

A Figura 3.5 mostra um cliente (conexão i) enviando um pedido para o módulo

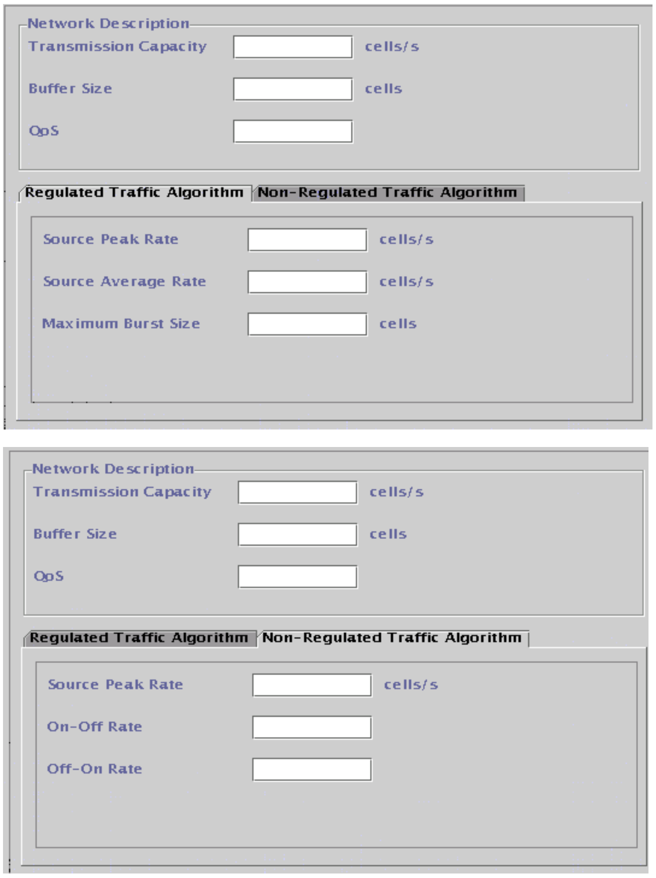


Figura 3.4: Interfaces do Tangram-II do módulo de admissão

de admissão. Atualmente, o módulo implementa os dois algoritmos de CAC descritos anteriormente nas seções 2.1 e 2.2. Cabe a este módulo admitir ou não o novo cliente. Caso o cliente seja autorizado, no caso de um aplicativo gerador de tráfego, o módulo envia um identificador e o processo de geração de tráfego na rede é iniciado.

O módulo de admissão armazena uma lista dos clientes já admitidos que ainda estão ativos conforme a Figura 3.5. Quando o cliente deseja encerrar a sessão, este envia um pacote de "disconnect" liberando o identificador na lista e, consequentemente a banda anteriormente alocada.

É possível através desta extensão aproveitar o módulo, para integrar outros aplicativos no futuro (por exemplo um servidor de voz ou de vídeo). O novo aplicativo deve incluir um arquivo de cabeçalho contendo a definição das estruturas de dados

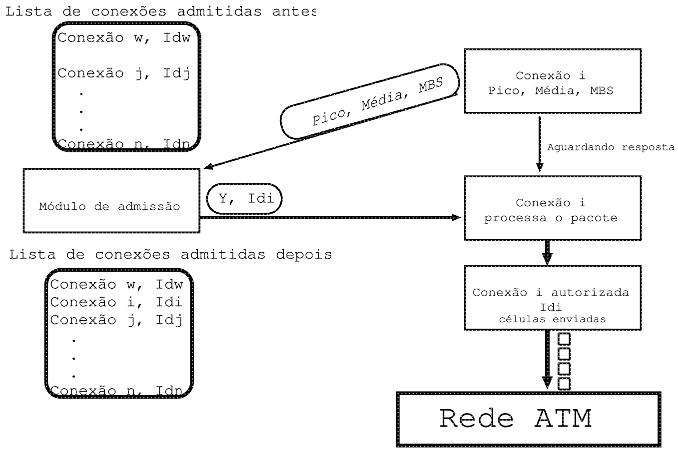


Figura 3.5: Comunicação com o Módulo de Admissão

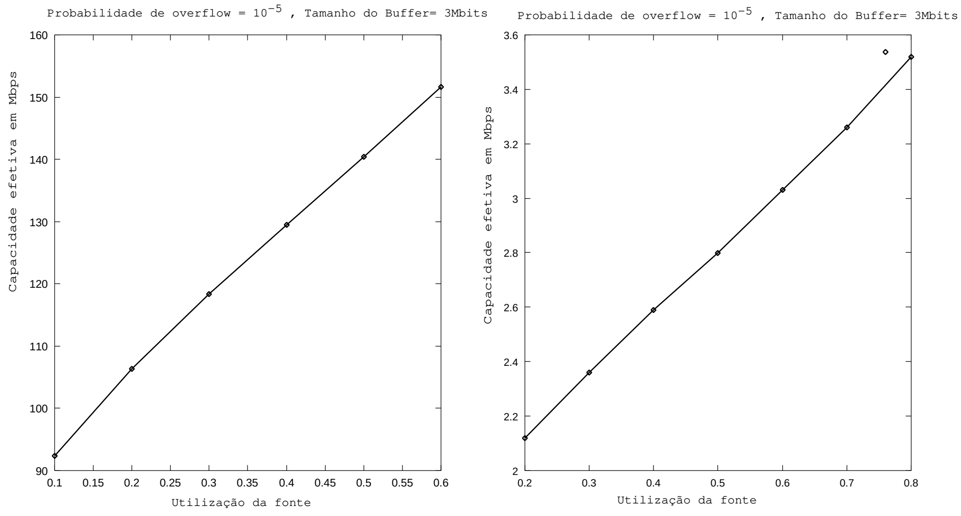
e funções do protocolo de comunicação mostrado na Figura 3.5.

A título de ilustração, mostraremos alguns gráficos com resultados coletados a partir dos algoritmos implementados no Módulo de Admissão. O objetivo é validar os resultados dos algoritmos implementados com os gráficos indicados nos artigos originais.

A Figura 3.6 mostra a *capacidade efetiva* em função da utilização definida como $\frac{\lambda}{\lambda+\mu}$, onde λ é a taxa de off-on e μ é a taxa de on-off, para garantir uma probabilidade de buffer overflow $\epsilon = 10^{-5}$, com um tamanho de buffer de 3 Mbits.

Os valores calculados para a *capacidade efetiva* na Figura 3.6 foram obtidos a partir do algoritmo da seção 2.1 implementado no Módulo de Admissão. O gráfico A foi retirado a partir do valor calculado da *capacidade efetiva*, e é equivalente à Figura 3 do artigo [17]. O gráfico B da Figura 3.6 mostra a *capacidade efetiva* em função da utilização de 5 fontes homogêneas multiplexadas. O gráfico B também foi retirado a partir do valor calculado da *capacidade efetiva*, e é equivalente à Figura 4 do artigo [17].

As Figuras 3.7 e 3.8 mostram o comportamento da *capacidade efetiva* $\tilde{C}$ em função da capacidade de transmissão do enlace ATM C. Os valores calculados para a *capacidade efetiva* foram obtidos a partir do algoritmo da seção 2.2 implementado no Módulo de Admissão.



1 fonte com
Pico = 4Mbps, MBS = 100ms
Gráfico A

5 fontes com
Pico = 40Mbps, MBS = 10ms
Gráfico B

Figura 3.6: Capacidade Efetiva em função da utilização das fontes

Para estes gráficos, calculamos o valor da *capacidade efetiva com perda zero* c_0 pela equação 2.26 e calculamos todas as demais equações (com excessão da equação 2.33) do capítulo 2 usando X/C fixo. O próximo passo foi calcular o valor da *capacidade efetiva estatisticamente multiplexada* através da equação 2.33 variando o valor da capacidade de transmissão C de 0 a 160Mbps. Podemos observar na Figura 3.7 que para valores de C menores que 22Mbps a *capacidade efetiva estatisticamente multiplexada* é negativa. Isto devido à condição de existência da função $F_k(s^*)$, neste caso esta função não existe para valores de C inferiores a 22Mbps. A partir deste ponto em diante a função passa a existir.

Se não tivéssemos calculado $F_k(s^*)$ com X/C fixo, onde $C = 45$Mbps no primeiro passo, a função $F_k(s^*)$ não existiria e consequentemente as demais equações que dependem desta função também não poderiam ser calculadas. Logo, foi calculado $F_k(s^*)$ com X/C fixo apenas uma vez e as demais equações utilizam os resultados deste cálculo.

Na Figura 3.8, para valores de capacidade de transmissão C menores que 22Mbps a *capacidade efetiva* permanece constante c_0 e para valores de C maiores que 22Mbps, a *capacidade efetiva* decai rapidamente. Note que consideramos apenas um tipo de fonte em cada Gráfico. O gráfico A foi retirado a partir do valor calculado da *capacidade efetiva*, e é equivalente à Figura 9 do artigo [31]. O gráfico B da Figura 3.8, mostra a *capacidade efetiva*, considerando outro tipo de fonte. O gráfico B também foi retirado a partir do valor calculado da *capacidade efetiva*, e é equivalente à Figura 10 do artigo [31].

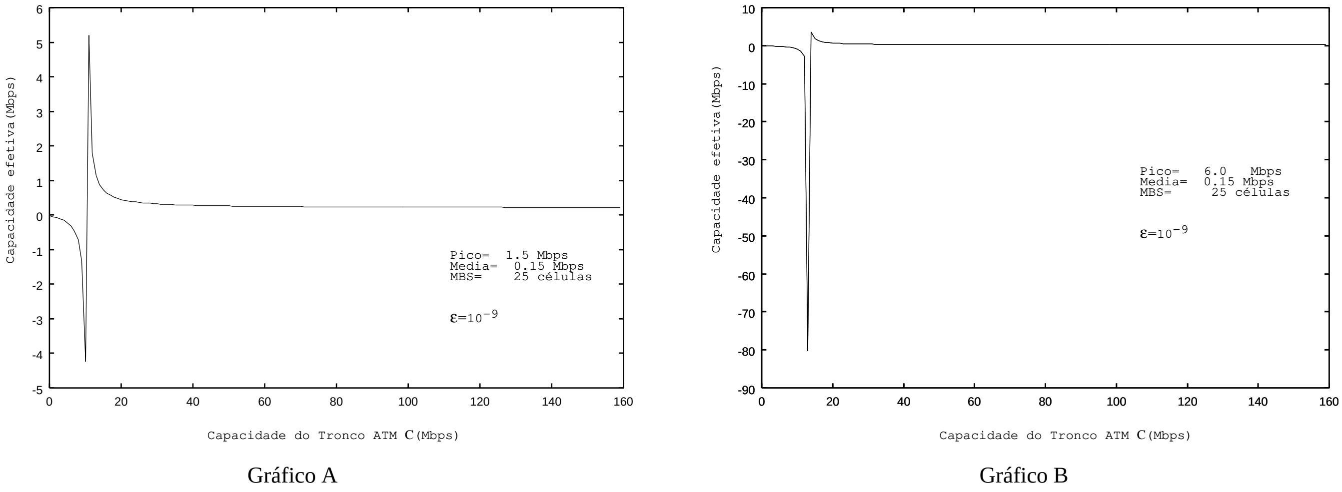


Figura 3.7: Capacidade Efetiva estatisticamente multiplexada

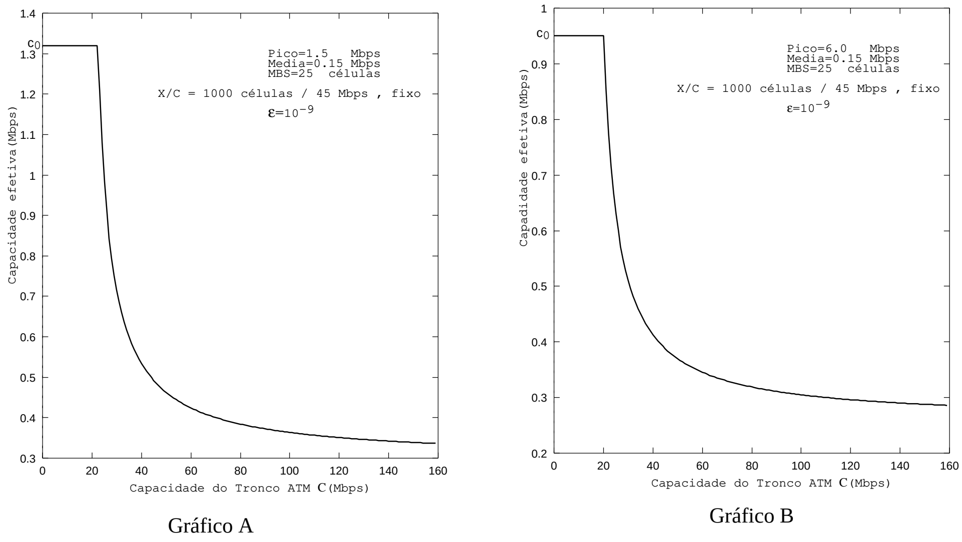


Figura 3.8: Capacidade Efetiva versus Capacidade do tronco ATM, onde X/C é mantido fixo

Capítulo 4

Controle de Admissão do Comutador ATM Fore-Systems ASX-1000

O objetivo deste capítulo é analisar como pode ser feito o controle de admissão no comutador ATM da Fore, apresentando os resultados da análise.

Este capítulo mostra como os algoritmos de CAC descritos nas seções 2.1 e 2.2 podem ser usados na implementação de um CAC para o comutador ATM ASX-1000 [38] da Fore Systems. Através de parâmetros configuráveis, este comutador permite ajustar o número de fontes admitidas em uma porta, mas não implementa nenhum algoritmo de CAC. A decisão de admissão do comutador é baseada na taxa de pico da fonte por default. Na seção 4.1, iremos discutir o comportamento destes parâmetros. Na seção 4.2 discutimos como poderiam ser ajustados os parâmetros do comutador para que este implemente um algoritmo de CAC teórico.

4.1 Método de Admissão do Comutador ATM Fore-Systems modelo ASX-1000

Para os testes realizados no comutador, foram configurados diferentes contratos UPC que estão indicados na Tabela 4.1. Através do contrato, os usuários devem declarar as características do tráfego, selecionando assim o tipo de serviço oferecido pela rede. Como indica a Tabela 4.1, selecionamos 4 tipos de tráfego, onde a coluna *MBS* indica o tamanho máximo da rajada em células. Os contratos abaixo foram retirados das propostas originais de algoritmos de CAC selecionadas.

Tipo do tráfego	**Pico em Mbps**	**Média em Mbps**	**MBS em células**
1	42.2	29.3	5000
2	4	2.1	366
3	6	0.15	25
4	1.5	0.15	250

Tabela 4.1: Tipos de tráfego configurados no comutador ATM

Foram também configurados VC(s) tipo VBR(s), associando-os à algum tipo de tráfego anteriormente configurado. Quando um circuito virtual permanente (PVC) VBR é configurado, necessariamente algum recurso tipo banda/buffer é reservado ao novo PVC VBR, visando garantir a QoS.

Após a configuração de VC(s) com QoS, é possível obter uma indicação dos recursos alocados para VC(s) VBR(s) em termos de banda. A capacidade indicada em uma porta do comutador é obtida na seção de VPT (Virtual Path Terminator). Esta seção apresenta a *banda nominal* para todos os VC(s) chegando e saindo, associados ao VPi correspondente, em cada porta do comutador. Os valores de *capacidade nominal* obtidos para os resultados abaixo são tirados desta seção (VTP) no comutador ATM.

É importante notar que enquanto o comutador libera a configuração de VC(s) VBR(s) com destino à mesma porta de saída, conexões serão admitidas, até o instante em que mais nenhum VC possa ser configurado. Neste ponto os recursos correspondentes àquela porta estão totalmente alocados. Desta forma, é possível determinar quantas conexões são admitidas e comparar com os resultados do módulo de admissão implementado.

No comutador Fore ASX-1000, a *banda nominal* para um VC é sensível a dois parâmetros que são configuráveis em cada porta do comutador: *vbroverbooking* *(vbrob)* e *vbrbufferoverbooking (vbrbuffob)*.

O parâmetro *vbrob* influencia a banda alocada à cada VC. Este parâmetro é uma porcentagem em relação à *banda nominal* reservada por VC. Por exemplo, se o parâmetro *vbrob* é 200, a banda alocada para cada VC é dividida por 2, se o parâmetro *vbrob* é 300, a banda alocada é dividida por 3, etc. O valor máximo do parâmetro é 32767.

O parâmetro *vbrbuffob* determina também uma porcentagem, mas em relação ao número de células do buffer alocado a cada VC. O valor inicial configurado de fábrica para os parâmetros são: *vbrob* = 100 e *vbrbuffob* = 100. Com os valores "default" destes parâmetros, a *capacidade nominal* alocada por VC é exatamente a taxa de pico. O valor do buffer alocado não é especificado.

As Tabelas 4.2, 4.3 e 4.4 mostram o comportamento de um VC VBR do tipo 1, em função do parâmetro $vbrbuffob$ variando de (100 a 32767), e $vbrob$ variando de (100, 150 e 200).

Podemos observar que, na Tabela 4.2 a *capacidade nominal* é a taxa de pico do tipo de tráfego 1 (42.2Mbps) para os valores default dos parâmetros do comutador, $vbrob = 100$ e $vbrbuffob = 100$. Se dobrarmos o valor de $vbrob$ para $vbrob = 200$, a *capacidade nominal* é dividida por 2, como indicado na primeira linha da Tabela 4.4. A *capacidade nominal* decai linearmente com o aumento do parâmetro $vbrob$. Ainda pelas Tabelas 4.2 , 4.3 e 4.4, podemos observar que variando o parâmetro $vbrbuffob$ a *capacidade nominal* apresenta uma queda não linear com o aumento do parâmetro $vbrbuffob$.

Na Figura 4.1 queremos analisar como a *capacidade nominal* é afetada apenas em função da variação do parâmetro $vbrbuffob$. Note que variando o parâmetro vbrbuffob, a *capacidade nominal* do VC VBR do tipo de tráfego 1 pode assumir o valor de pico de 42.2Mbps ou até 32Mbps, ou seja, a *capacidade nominal* varia de 10Mbps (de 42 a 32). Esta variação corresponde a 25% da taxa de pico da conexão, dependendo apenas do parâmetro *vbrbuffob* configurado.

As Tabelas 4.2, 4.3 , 4.4 e a Figura 4.1 mostram que o método de CAC do comuta-

Capacidade Nominal	vbrob	vbrbuffob	Tipo	Porcentagem do Pico
42.2	100	100	1	100%
42.1	100	150	1	99.76%
42.0	100	200	1	99.52%
42.0	100	250	1	99.52%
41.9	100	300	1	99.29%
41.8	100	400	1	99.05%
41.7	100	500	1	98.81%
41.5	100	600	1	98.34%
41.4	100	700	1	98.10%
41.3	100	800	1	97.87%
41.2	100	900	1	97.63%
41.1	100	1000	1	97.39%
40.1	100	2000	1	95.02%
39.3	100	3000	1	93.13%
38.5	100	4000	1	91.23%
37.9	100	5000	1	89.81%
37.3	100	6000	1	83.39%
36.8	100	7000	1	87.20%
36.3	100	8000	1	86.02%
35.9	100	9000	1	85.07%
35.5	100	10000	1	84.12%
34.2	100	15000	1	81.04%
33.3	100	20000	1	78.91%
32.7	100	25000	1	77.49%
32.3	100	30000	1	76.54%
32.1	100	32767	1	76.07%

Tabela 4.2: Variação da capacidade nominal para os parâmetros de overbooking, onde $vbrob = 100$ no comutador ATM Fore-Systems ASX-1000

Capacidade Nominal	vbrob	vbrbuffob	Tipo	Porcentagem do Pico
28.1	150	100	1	65.59%
28.1	150	150	1	65.59%
28	150	200	1	66.35%
28	150	250	1	66.35%
27.9	150	300	1	66.11%
27.9	150	400	1	66.11%
27.8	150	500	1	65.88%
27.7	150	600	1	65.64%
27.6	150	700	1	65.40%
27.5	150	800	1	65.17%
27.5	150	900	1	65.17%
27.4	150	1000	1	64.93%
26.8	150	2000	1	63.51%
26.2	150	3000	1	62.09%
25.7	150	4000	1	60.90%
25.2	150	5000	1	59.72%
24.9	150	6000	1	59.00%
24.5	150	7000	1	58.06%
24.2	150	8000	1	57.35%
23.9	150	9000	1	56.64%
23.7	150	10000	1	56.16%
22.8	150	15000	1	54.03%
22.2	150	20000	1	52.61%
21.8	150	25000	1	51.66%
21.5	150	30000	1	50.95%
21.4	150	32767	1	50.71%

Tabela 4.3: Variação da capacidade nominal para os parâmetros de overbooking, onde $vbrob = 150$ no comutador ATM Fore-Systems ASX-1000

Capacidade Nominal	vbrob	vbrbuffob	Tipo	Porcentagem do Pico
21.1	200	100	1	50.00%
21.1	200	150	1	50.00%
21.0	200	200	1	49.76%
21.0	200	250	1	49.76%
21.0	200	300	1	49.76%
20.9	200	400	1	49.52%
20.8	200	500	1	49.29%
20.8	200	600	1	49.29%
20.7	200	700	1	49.05%
20.7	200	800	1	49.05%
20.6	200	900	1	48.82%
20.5	200	1000	1	48.58%
20.1	200	2000	1	47.63%
19.6	200	3000	1	46.45%
19.3	200	4000	1	45.73%
18.9	200	5000	1	44.79%
18.6	200	6000	1	44.08%
18.4	200	7000	1	43.60%
18.2	200	8000	1	43.13%
18.0	200	9000	1	42.65%
17.8	200	10000	1	42.18%
17.1	200	15000	1	40.52%
16.7	200	20000	1	39.57%
16.3	200	25000	1	38.63%
16.1	200	30000	1	38.15%
16.0	200	32767	1	37.91%

Tabela 4.4: Variação da capacidade nominal para os parâmetros de overbooking, onde $vbrob = 200$ no comutador ATM Fore-Systems ASX-1000

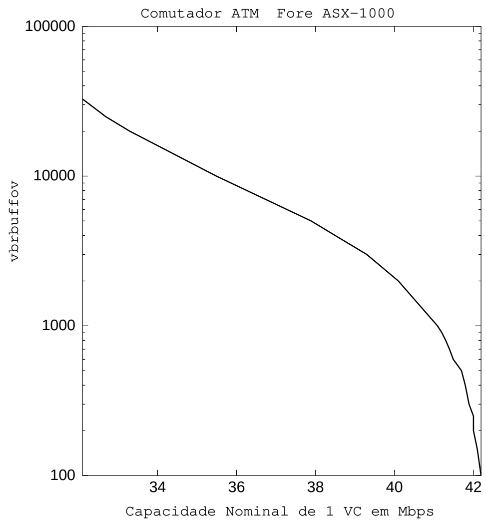


Figura 4.1: vbrbuffob em função da *Capacidade Nominal* de um VC VBR do tipo 1

dor ATM Fore-Systems ASX-1000 pode ser ajustado, baseando-se em 2 parâmetros configuráveis pelo usuário. A tarefa de ajustar a *capacidade alocada ideal* dentro de um ambiente real fica por conta do administrador de rede. O método de CAC do comutador é bastante flexível, mas nem sempre o administrador conhece a fundo o problema de controle de admissão.

4.2 A Implementação de um CAC para o Comutador Usando Algoritmos de CACs Teóricos

O objetivo desta seção é estudar se, através dos parâmetros configuráveis no comutador Fore discutidos na seção anterior, é possível implantar os algoritmos de CAC teóricos discutidos no capítulo 2.

As curvas teóricas desta seção foram obtidas a partir de valores calculados da *capacidade efetiva*, usando os algoritmos implementados no **Módulo de Admissão** descritos detalhadamente no capítulo anterior. As curvas para o comutador ATM foram obtidas a partir da *capacidade nominal* retirada da seção de VPT(Virtual Path Terminator) do comutador Fore.

A capacidade de transmissão C do comutador ATM da Fore Systems ASX-1000 em nosso laboratório é 366000 células/s em cada porta de saída. O tamanho do buffer de cada porta de saída, para este modelo pode variar. Segundo o manual do fabricante os valores podem ser 16 Kcélulas e no máximo 64 Kcélulas. Diante disto, para as curvas teóricas dos algoritmos de CAC usamos estes 2 valores de tamanho de buffer.

Para a Figura 4.2, calculamos o valor da *capacidade efetiva* e o número de fontes admitidas, usando o algoritmo de CAC teórico da seção 2.1, para um enlace ATM onde o tamanho do buffer é $X = 65535$ ou $X = 16384$ células, a QoS desejada $\epsilon = 10^{-5}$ e a capacidade de transmissão $C = 366000$ células/s. Os valores foram escolhidos a partir das características do comutador ATM que estamos analisando.

Os gráficos A e B da Figura 4.2 mostram uma comparação entre o comutador ATM e o algoritmo de CAC teórico com diferentes valores de tamanho de buffer.

A fonte utilizada no gráfico A possui taxa de pico 42.3 Mbps, taxa média 29.3 Mbps e tamanho médio da rajada 100ms. No gráfico A, a *capacidade efetiva* calculada é 33.76 Mbps para um buffer $X = 65535$ e 37.78 Mbps para um buffer 16384 células. Os valores dos parâmetros do comutador que melhor se ajustaram foram $vbrob = 100$ e $vbrbuffob = 18500$ com *capacidade nominal* 33.7 Mbps e $vbrob = 100$ e $vbrbuffob = 5500$ com *capacidade nominal* 37.7 Mbps. A conexão no comutador é do tipo 1 da Tabela 4.1.

O gráfico B mostra fontes homogêneas do tipo2 da Tabela 4.1, fixando os parâmetros em $vbrob = 100$ e $vbrbuffob = 18500$ do comutador ATM. A curva do algoritmo de CAC teórico, foi gerada usando-se os mesmos valores do enlace ATM do gráfico A, sendo apenas a fonte diferente. A fonte utilizada possui taxa de pico 4.2 Mbps, taxa média 29.3 Mbps e tamanho médio da rajada 10ms. A *Capacidade efetiva* calculada para uma fonte é 3.06 Mbps para buffer $X = 16384$, e 2.9 Mbps para buffer $X = 65535$. A *Capacidade Efetiva Agregada* é 29.4 Mbps para buffer $X = 65535$, e 30.68 para buffer $X = 16384$ e a *capacidade nominal agregada* no comutador ATM é 26.9 Mbps.

Quando o número de fontes homogêneas multiplexadas não é grande, é possível verificar que os parâmetros do comutador ATM podem ser ajustados de forma a simular o algoritmo de CAC proposto por [17]. Existe uma diferença esperada entre as curvas teóricas com tamanho de buffer diferente. No gráfico B, mantendo fixo os parâmetros $vbrob = 100$ e $vbrbuffob = 18500$, podemos notar que a medida que o número de fontes cresce existe uma tendência de divergência entre o algoritmo de CAC teórico e o comutador ATM usando os valores configuráveis $vbrob$ e $vbrbuffob$.

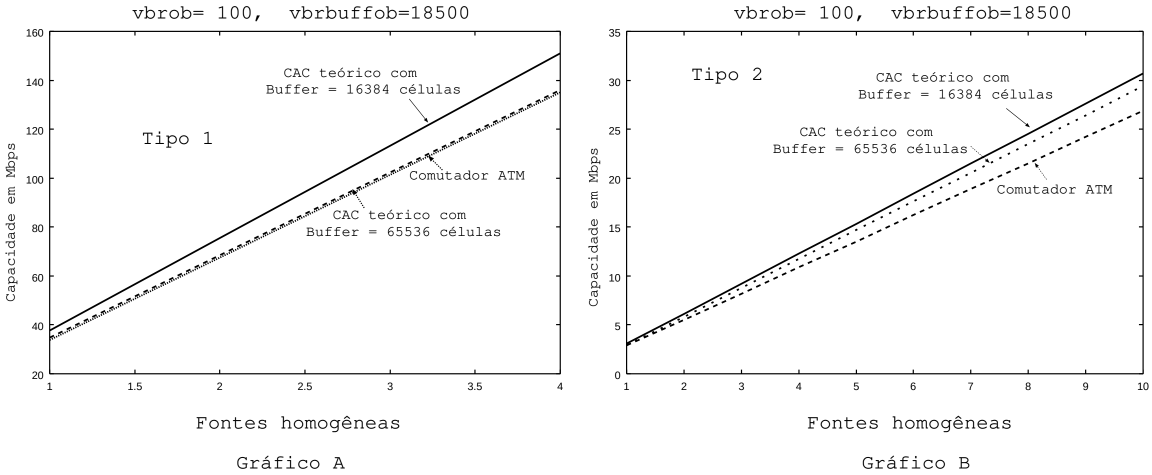


Figura 4.2: Comparativo entre comutador e algoritmo de CAC teórico proposto por [17] para fontes homogêneas

A Figura 4.3 foi gerada para os mesmos valores anteriores dos parâmetros do comutador $vbrob = 100$ e $vbrbuffob = 18500$, onde encontramos o número de fontes admitidas para o tipo de tráfego 1 e 2 da Tabela 4.1. Na curva do algoritmo teórico, mantemos a especificação dos gráficos A e B da Figura 4.2, mas obtendo o número de fontes admitidas. O número de fontes admitidas para os diferentes tamanhos de buffer é o mesmo para o algoritmo de CAC teórico.

Observando a região de admissão da Figura 4.3, notamos a diferença entre o número de fontes admitidas pelo algoritmo teórico e o número de fontes admitidas no comutador para os valores $vbrob = 100$ com $vbrbuffob = 18500$. No caso das fontes de tráfego do tipo 1 o número de fontes é 4 para ambos. Enquanto que no caso das fontes de tráfego do tipo 2, temos 44 fontes admitidas no comutador e 48 fontes admitidas no algoritmo teórico. Quando fontes de diferentes tipos (heterogêneas) são multiplexadas e os parâmetros $vbrob$ e $vbrbuffob$ são ajustados, podemos notar que não é possível simular o algoritmo teórico de CAC proposto por [17]. Para fontes homogêneas é possível simular o algoritmo teórico proposto por [17], mas para um número não muito grande de fontes de tráfego. À medida que o número de fontes multiplexadas aumenta, existe uma tendência do comutador divergir do algoritmo teórico.

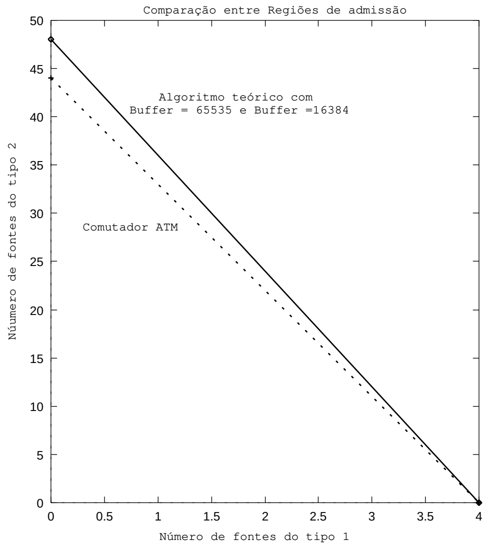


Figura 4.3: Região de Admissão

Na Figura 4.4, fazemos uma comparação entre as regiões de admissão do comutador ATM e outro algoritmo teórico de CAC discutido na seção 2.2 proposto por [31]. Os resultados das curvas do algoritmo teórico são obtidos da implementação deste algoritmo no **Módulo de Admissão**. Nesta Figura visamos mostrar uma possível simulação do algoritmo teórico de CAC, ajustando valores dos parâmetros do comutador ATM para um número grande de fontes.

O número de fontes admitidas no algoritmo de CAC teórico, foi obtida para um enlace ATM onde o tamanho do buffer é $X = 65535$ ou $X = 16384$ células, a QoS desejada $\epsilon = 10^{-5}$ e a capacidade de transmissão $C = 366000$ células/s. Os valores foram escolhidos a partir das características do comutador ATM que estamos analisando. As fontes utilizadas são as fontes de tráfego tipo 3 e tipo 4 da Tabela 4.1.

Para o tráfego tipo 3 com tamanho de buffer $X = 65535$, o número de fontes admitidas é 1000, a *capacidade efetiva* calculada é 0.15 Mbps, com tamanho de buffer

$X = 16384$ o número de fontes admitidas é 890, a *capacidade efetiva* calculada é 0.174 Mbps. Para o tráfego tipo 4 com buffer $X = 65535$ o número de fontes admitidas é 805 e a *capacidade efetiva* calculada é 0.192 Mbps, com tamanho de buffer $X = 16384$, o número de fontes admitidas é 685 e a *capacidade efetiva* é 0.226 Mbps.

Para o comutador, no gráfico A da Figura 4.4, ajustamos valores de *capacidade nominal* próximos às *capacidades efetivas* calculadas pelo algoritmo teórico para o tráfego do tipo 4. Os valores dos parâmetros que ficaram próximos são $vbrob = 150$ e $vbrbuffob = 2800$. Sendo assim, mantendo os valores de $vbrob = 150$ e $vbrbuffob = 2800$, obtivemos também o número de fontes admitidas para o tráfego do tipo 3.

Na curva do comutador do gráfico B da Figura 4.4, ajustamos valores de *capacidade nominal* próximo às *capacidade efetivas* calculadas pelo algoritmo teórico para o tráfego do tipo 3. Os valores dos parâmetros que ficaram próximos às *capacidades efetivas* calculadas são $vbrob = 150$ e $vbrbuffob = 500$. Para estes valores ajustados, obtivemos o número de fontes admitidas para o tráfego do tipo 4.

No gráfico A, fixando $vbrob = 150$ e $vbrbuffob = 2800$, temos 780 fontes admitidas no comutador em relação a 805 e 685 fontes teóricas admitidas para o tipo 4 de tráfego. Para estes mesmos parâmetros temos 1400 fontes admitidas no comutador em relação a 1000 e 890 fontes teóricas admitidas para o tipo 3 de tráfego.

No gráfico B, fixando $vbrob = 150$ e $vbrbuffob = 500$, temos 980 fontes admitidas no comutador em relação a 1000 e 890 fontes teóricas admitidas para o tráfego tipo 3. Para os mesmos parâmetros temos 202 fontes admitidas no comutador em relação a 805 e 685 fontes teóricas admitidas para o tráfego tipo 4.

Observando a região de admissão da Figura 4.4, notamos a diferença entre o número de fontes admitidas pelo algoritmo teórico e o número de fontes admitidas no comutador para diferentes valores de $vbrob$ com $vbrbuffob$. Novamente, quando fontes heterogêneas são multiplexadas e os parâmetros $vbrob$ e $vbrbuffob$ são ajustados, podemos notar que não é possível simular o algoritmo teórico de CAC proposto por [31].

Em suma, através dos parâmetros do comutador ATM da Fore-Systems ASX-1000 é possível ter uma boa flexibilidade no número de fontes admitidas em uma porta do comutador. Para um número não muito grande de fontes homogêneas, é possível simular algoritmos de CAC teóricos ajustando os parâmetros. À medida que o número de fontes multiplexadas aumenta, existe uma tendência do número de fontes no comutador divergir do número de fontes do algoritmo teórico. Para fontes heterogêneas não obtivemos uma boa simulação de nenhum dos 2 algoritmos teóricos testados.

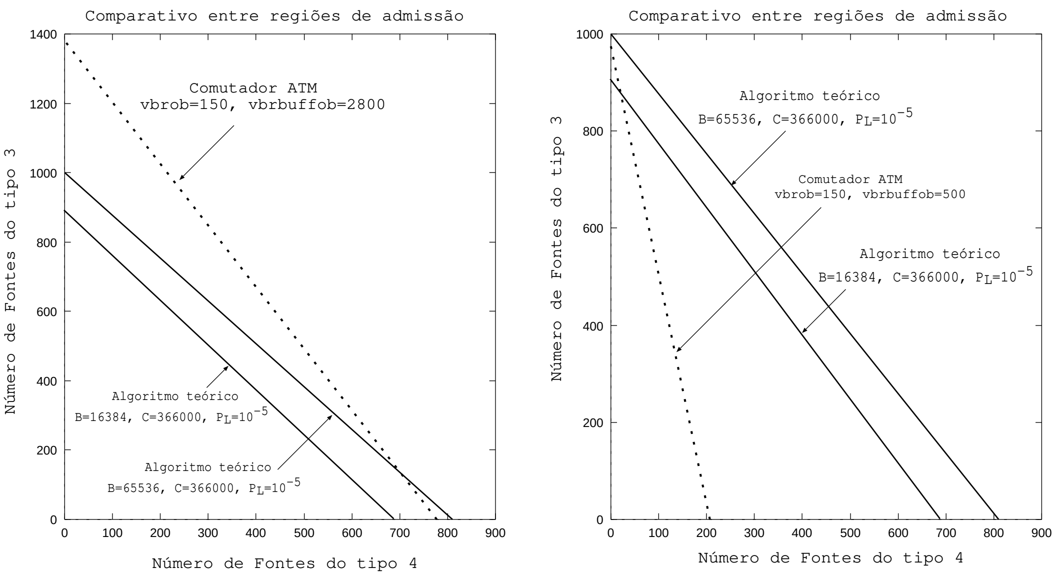


Figura A Figura B

Figura 4.4: Comparativo entre Regiões de Admissão

Capítulo 5

Medições e Análise de QoS no Comutador ATM

O objetivo deste capítulo é analisar a QoS oferecida pelo comutador em termos de perda de células, quando os algoritmos de CAC estudados no capítulo 2 são aplicados. A análise foi feita utilizando o ambiente de testes implementado, comparando os resultados teóricos com as medidas coletadas.

Também estamos interessados em obter medidas que demonstrem a eficiência do comutador ATM em vários aspectos. Entre os principais aspectos destacamos o tamanho do buffer em cada porta do comutador, a política de escalonamento implementada e a QoS oferecida pelo comutador para tráfegos regulados e não-regulados. Nos testes com tráfegos não-regulados o mecanismo de policiamento do comutador é desabilitado, e para tráfegos regulados o mecanismo de policiamento do comutador é habilitado.

O tráfego de fundo CBR que utilizamos nos experimentos, serve simplesmente para diminuir a banda disponível aos circuitos VBRs, de forma a obter perda de células. Para cada amostra coletamos o número de células perdidas, através da interface de gerência disponível no comutador da Fore-Systems. O comutador Fore-Systems oferece um servidor que realiza consultas SNMP à MIB do comutador mostrando o número de células perdidas em cada VC.

A Figura 5.1 mostra, de uma maneira geral, o ambiente de testes. Nesta Figura temos 5 estações que possuem placa ATM e um comutador ATM modelo ASX-1000 conectados via OC3 (155Mbps). Nos experimentos que serão apresentados adiante, as estações alocam taxas nas placas ATM das mesmas. No comutador ATM, observe que existe um conjunto de PVCs VBRs ou CBRs configurados, onde todos estão direcionados a mesma porta de saída. Neste ambiente, note que as taxas alocadas nas estações, os quais enviam dados aos PVCs CBRs e VBRs do comutador ATM. Em suma, temos do lado das estações geradores enviando tráfego CBR com taxas alocadas nas placas locais de cada máquina, equivalentes a taxa de pico do tráfego CBR. O tráfego transmitido pelos geradores é recebido em um VC do comutador com destino a uma porta de saída, e isto vale para todos os VCs. As medidas de perda de células são coletadas na porta de saída do comutador ATM.

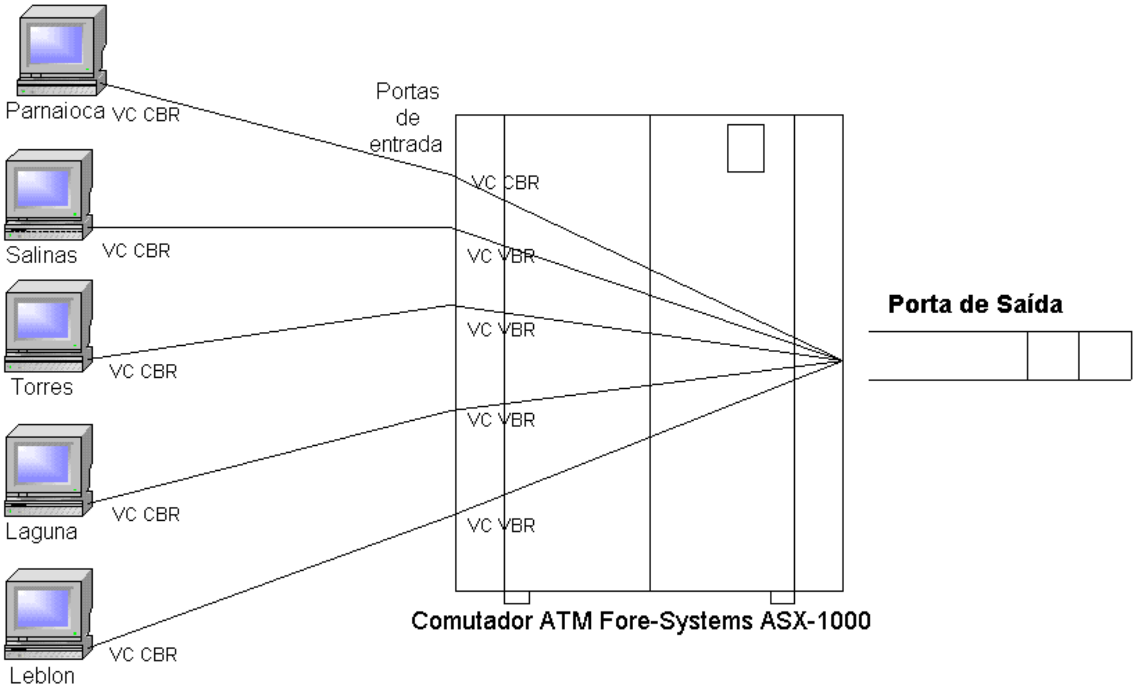


Figura 5.1: Ambiente de testes - LAND(Laboratório de Análise e Desempenho de Sistemas de Computação e Comunicação)

5.0.1 Tamanho do Buffer no Comutador ATM

Nesta seção queremos analisar o tamanho total do buffer disponível em uma porta do comutador através de medições. Neste ponto ainda não realizamos medidas para determinar a política de escalonamento do comutador ATM, mas segundo especificação do fabricante a política de escalonamento implementada é a chamada Per-VC Queueing. Logo, também estamos interessados em determinar a fração do buffer alocada por VC.

Para cada experimento, analisamos o crescimento do processo de perda em uma porta do comutador. O número de células perdidas é coletado em cada VC para um dado tempo total de geração T. Em seguida, o tempo total de geração é incrementado $T+1$ e novas medidas são coletadas, e assim sucessivamente. Note que, apenas neste experimento, os tráfegos gerados são todos determinísticos. Para uma análise precisa do crescimento das células perdidas em diversos instantes de tempo, manter a taxa do tráfego de chegada constante no comutador é a forma mais viável de obter a curva de crescimento de células perdidas.

No comutador ATM, configuramos 4 VC(s) VBR(s) com contrato UPC (taxa de pico $R = 27084$ células/s, taxa média $m = 17411$ células/s e tamanho máximo da rajada $b = 1095$ células) e 1 VC CBR (taxa de pico 118000 células/s). Os geradores foram divididos em 4 estações, cada uma com 1 gerador de tráfego, além de uma quinta estação com um gerador de tráfego de fundo CBR. O policiamento foi desabilitado em todas as portas do comutador.

Nas estações, cada gerador de tráfego abre um VC CBR de taxa 60289 células/s e envia tráfego determinístico de modo espalhado com taxa 60289 células/s. Apenas o gerador de tráfego de fundo abre um VC CBR com taxa 118000 células/s. Estes valores foram escolhidos com o objetivo de que o tráfego gerado agregado seja maior que a taxa de serviço em uma porta de saída do comutador.

A Figura 5.2 mostra 3 experimentos, sobre o número de células perdidas em cada VC, em função do tempo total de geração T. Através das curvas de crescimento de perda da Figura 5.2 calculamos o tamanho do buffer X_i do i-ésimo VC como o instante de início do processo de perda $t_{0,i}$ multiplicado pela taxa média de crescimento da perda A_i (coeficiente angular da reta). Logo temos:

$$X_i = A_i * t_{0,i} \tag{5.1}$$

,onde o tamanho total médio do buffer do comutador é $X = \sum_i X_i$.

Calculando o tamanho total médio do buffer, através das medidas da Figura 5.2, temos $X = 17805$ células com $(+_{-}3651)$ células (intervalo de confiança de 95%). O tamanho médio do buffer calculado em cada VC é $X_1 = 4464$, $X_2 = 4465$, $X_3 = 4458$ e $X_4 = 4428$.

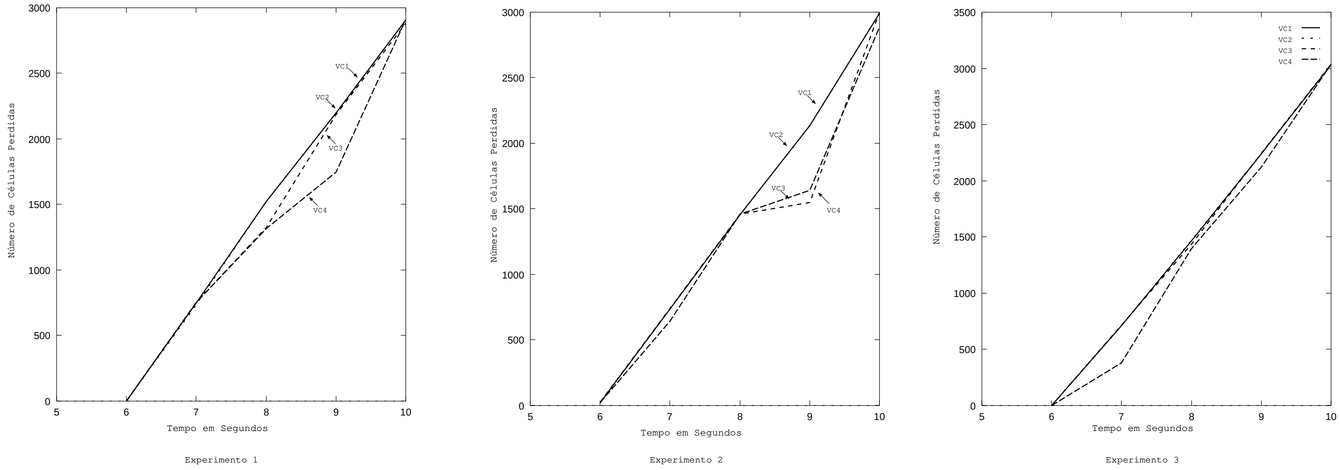


Figura 5.2: Crescimento do número de células perdidas por VC, com tráfego de fundo CBR de $50Mbps$ e policiamento desabilitado no comutador

Nos próximos experimentos desejamos mostrar o impacto no processo de perda, quando um contrato UPC é alterado. Com diferentes contratos e mantendo as taxas dos geradores de tráfego, desejamos avaliar as diferenças de células perdidas em cada VC.

Ainda neste experimento, alteramos no comutador ATM os contratos UPC ($R = 27084$, $m = 27000$ e $b = 1095$) em todos os VCs VBRs. Então, repetimos exaustivamente a geração de tráfego, e o processo de perda de células manteve a mesma ordem de crescimento. Também para o contrato UPC ($R = 27084$, $m = 27000$ e $b = 10000$) o processo de perda permaneceu com a mesma ordem de crescimento.

Para as medidas das curvas B da Figura 5.3 criamos no comutador ATM um novo contrato UPC (R = 37500, m = 26500 e b = 1095) e associamos a este um VC VBR. Os demais VCs VBRs foram deixados com o contrato anterior (R = 27084,m = 17411 e b = 1095). O tráfego de fundo CBR também foi mantido como no experimento anterior. Os geradores foram distribuídos da mesma forma e a **taxa de cada gerador foi mantida**.

Para as medidas das curvas C da Figura 5.3 associamos 2 VCs ao novo contrato UPC (R = 37500, m = 26500 e b = 1095). Os demais VCs VBRs foram deixados com o contrato anterior (R = 27084,m = 17411 e b = 1095). O tráfego de fundo CBR também foi mantido como no experimento anterior. Os geradores foram distribuídos da mesma forma e a **taxa de cada gerador foi mantida**. Nas curvas A da Figura 5.3 mostramos um dos experimentos da Figura 5.2.

A partir das medições das curvas B da Figura 5.3, calculamos o tamanho total médio do buffer usando a equação 5.1 resultando em $X = 14039(+_-1603)$ células, onde não houve perda no VC associado ao contrato UPC (R = 37500, m = 26500 e b = 1095). Para as medidas das curvas C da Figura 5.3 o tamanho total médio do buffer calculado foi $X = 12078(+_-1680)$ células, onde também não ocorreram perdas nos 2 VCs VBRs associados ao contrato UPC (R = 37500, m = 26500 e b = 1095).

Analisando as Figuras 5.2 e 5.3, podemos observar que alterando a taxa de pico de VCs VBRs, ocorre um crescimento na perda de células dos demais VCs, mesmo mantendo o tráfego de chegada determinístico e com as mesmas taxas nos geradores de cada estação. O tamanho total do buffer calculado caiu de 17805 para 14039 e 12078 indicando que as porções do buffer alocadas aos VCs com menores taxas de pico foram reduzidas. Outra forma de observar a redução do buffer alocado aos VCs de menores taxas de pico é visualizar nas curvas B e C que o instante inicial do processo de perda t_0 diminui de tal forma que, mesmo com aumento da taxa de crescimento, o produto calculado nas curvas B 14039 e C 12078 é menor que o produto calculado nas curvas A 17805.

Supondo que na arquitetura do comutador ATM ASX-1000 o arranjo total de buffer não é compartilhado entre portas, então podemos afirmar que em uma porta do comutador em nosso laboratório, o tamanho total do buffer compartilhado entre todos os VCS ativos é 17805($+_-$3651) células. A alocação de buffer é realizada em função da taxa de pico R dos VCs, não sendo influenciada pelo aumento da taxa média m ou tamanho máximo da rajada b.

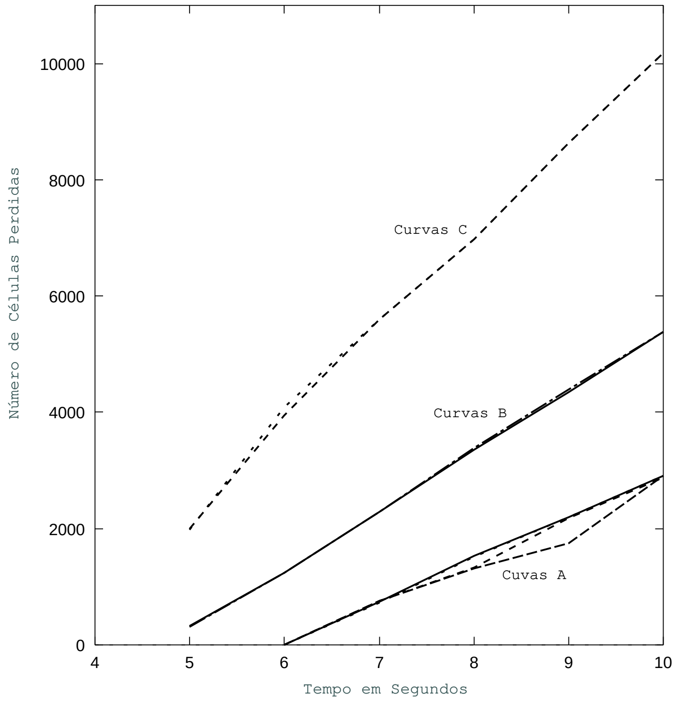


Figura 5.3: Crescimento do número de células perdidas para VCs com diferentes taxas de pico, com tráfego de fundo CBR de $50Mbps$ e policiamento desabilitado nas portas de entrada

5.1 A Política de Escalonamento Implementada no Comutador

Algoritmos de escalonamento de tráfego constituem um componente crítico nas redes multimídia. Muitos algoritmos de escalonamento têm sido propostos de forma a oferecer diferentes garantias de QoS [40]. O projeto de um algoritmo de escalonamento envolve, inevitavelmente, a preocupação com o retardo e perdas na fila, juntamente com a complexidade de implementação, além de alocação de recursos.

Nesta seção queremos demonstrar, através de experimentos, a política de escalonamento implementada no comutador.

No comutador ATM, configuramos 5 VC(s) VBR(s) associados ao contrato UPC (taxa de pico $R = 27084$ células/s, taxa média $m = 12950$ células/s e tamanho máximo da rajada $b = 1095$ células) e 1 VC CBR com taxa de pico 280660 células/s. Os geradores foram divididos em 4 estações, cada uma com 1 gerador de tráfego e 1 estação com 2 geradores, além de uma quinta estação com um gerador de tráfego de fundo CBR. O policiamento foi desabilitado em todas as portas do comutador.

Nas estações, cada gerador de tráfego abre um VC CBR de taxa 27084 células/s e envia tráfego modelado por uma cadeia de markov de modo espalhado. Apenas o gerador de tráfego de fundo abre um VC CBR com taxa 280660 células/s e envia tráfego determinístico.

Neste experimento, variamos a taxa média de uma das fontes, analisando o impacto que esta fonte traz sobre o total de células perdidas. O tempo total de geração de tráfego foi de $T = 120$ segundos. A fração de perda em cada VC (FP) foi calculada através da divisão do número de células perdidas em um único VC (p) pelo número total de células perdidas (P), i.e, $FP = \frac{p}{P}$. Várias amostras foram coletadas repetindo este experimento.

A Figura 5.4 mostra o teste de política de escalonamento de células implementada no comutador ATM. No gráfico A da Figura 5.4, geramos tráfego modelado por uma cadeia de Markov, onde 5 fontes têm taxa de pico 27084, tamanho médio da rajada $b = 100$ms, 4 fontes têm taxa média $m = 12950$ e a fonte gerada no VC5 tem taxa média $m = 16933$. Para um número total médio de perdas de células de 255925($+_-$5118) células, 89712($+_-$5382) células, em média, correspondem às perdas ocorridas no VC5.

No gráfico B da Figura 5.4, geramos tráfego modelado por uma cadeia de Markov, onde as 5 fontes têm taxa de pico 27084, tamanho médio da rajada $b = 100$ms, 4 fontes têm taxa média $m = 12950$ e a fonte gerada no VC5 tem taxa média $m = 8405$. Para um total de perdas 68968($+_-$927) células, nenhuma perda de células ocorreu no VC5.

Note que no gráfico A, VC5 é responsável pela maior fração de perda, enquanto no gráfico B não existe perda em VC5. Se no gráfico B, VC5 não tivesse uma porção do buffer alocada, certamente células seriam perdidas neste VC, já que ocorreram perdas de células nos demais VCs. E paralelamente no gráfico A, VC5 também teria uma influência menor no total de células perdidas. Note que o número de células perdidas em cada VC diminui sensivelmente quando a taxa média do VC5 é reduzida, este resultado também era esperado visto que a taxa média agregada de chegada é reduzida.

As diferenças entre as frações de perda é devido a aleatoriedade do tráfego markoviano de cada VC. Tráfegos com mesma média, pico e rajada podem ter um número diferente de células perdidas em cada VC, principalmente para sessões de média e curta duração.

A política de alocar uma porção do buffer compartilhado à cada VC dinamicamente ativo, chama-se *Per-VC Queueing*. Este mecanismo é implementado no comutador ATM Fore-Systems ASX-1000.

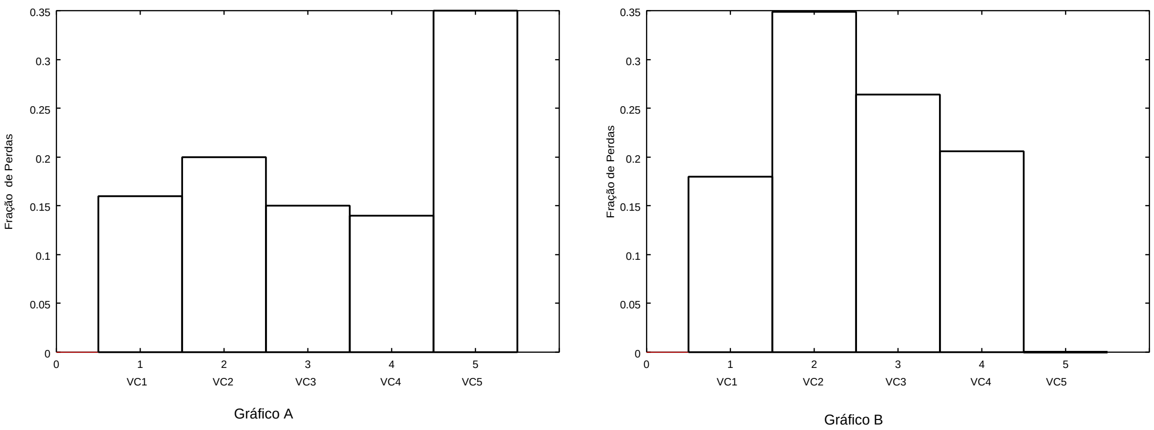


Figura 5.4: Fração de perda por VC, com tráfego de fundo CBR de $120Mbps$ e policiamento desabilitado

5.2 Medições de QoS do Comutador ATM

O objetivo desta seção é analisar a viabilidade da simulação dos algoritmos de CAC das seções 2.1 e 2.1 garantindo a QoS especificada no comutador ATM. Como o comutador possui parâmetros que permitem aumentar o número de fontes admitidas em uma porta, queremos analisar se a QoS é garantida mesmo alterando estes parâmetros. É importante observar que até este momento não sabemos qual a garantia de QoS quando ajustamos estes parâmetros no comutador.

O método de experimentação funciona da seguinte forma:

- Inicialmente, especificamos a QoS desejada, o tamanho do buffer e a capacidade de transmissão do comutador para os algoritmos de CAC teóricos. Estes primeiros parâmetros definem as características do enlace ATM. Em seguida, caracterizamos a fonte de tráfego a partir dos descritores de cada algoritmo de CAC. Os algoritmos de CAC, implementados no módulo de admissão, calculam a *capacidade efetiva* da fonte e o número de fontes admitidas no enlace ATM.

- No comutador, criamos um contrato UPC VBR para caracterizar a fonte de tráfego utilizada no algoritmo de CAC. Alteramos também os valores dos parâmetros de overbooking do comutador. Para determinada configuração, encontramos o número de VCs admitidos em uma porta de saída do comutador.

- O passo seguinte é gerar tráfego usando o gerador discutido no capítulo 3. O tráfego do gerador é modelado por uma cadeia de Markov especificada através dos mesmos descritores dos algoritmos de CAC teóricos. Para cada VC VBR estabelecido no comutador, temos um gerador de tráfego enviando dados na estação, onde cada aplicativo aloca a taxa de pico na placa da estação exatamente igual à taxa de pico do VC VBR do comutador. O tráfego gerado é espalhado de forma uniforme no intervalo de geração.

- Em cada experimento coletamos o número de células perdidas. Utilizamos a interface de gerência disponível no comutador da Fore-Systems para coletar medidas de células perdidas em cada experimento.

5.2.1 QoS para Tráfegos Não-Regulados

Nesta seção queremos extrair medidas que representam a QoS do comutador ATM. Nos experimentos a seguir, os tráfegos gerados são modelados por cadeia de Markov e o policiamento é desligado no comutador ATM. A capacidade de transmissão usada nos cálculos teóricos é obtida subtraindo a capacidade de transmissão da porta de saída pelo tráfego de fundo.

Utilizando o módulo de admissão implementado, selecionamos o algoritmo teórico de CAC da seção 2.1, para calcular a *capacidade efetiva* de uma fonte on-off com taxa de pico $R = 27084$ células/s , taxa média $m = 12950$ e $b = 1095$ células, onde b é o tamanho médio da rajada. A QoS utilizada é $\epsilon = 10^{-5}$, o tamanho do buffer $X = 17805$ e a capacidade de transmissão $C = 70754$ células/s. A *Capacidade efetiva* calculada para uma fonte é $\tilde{c_i} = 18503$ células/s e 3 fontes são admitidas, onde a *Capacidade efetiva agregada* é $\tilde{C} = 55510$ células/s.

Configuramos os parâmetros do comutador $vbrob = 150$ e $vbrbuffob = 100$, onde podem ser admitidos 3 VC(s) VBR(s) com taxa de pico $R = 27084$ células/s, taxa média $m = 12950$ células/s e $b = 1095$ células (b no comutador é o tamanho máximo da rajada). Também configuramos um tráfego de fundo CBR com taxa de 120Mbps para reduzir a banda disponível para os VCs VBRs.

Geramos tráfego divididos em 3 estações, cada uma com 1 gerador de tráfego, além de uma quarta estação com um gerador de tráfego de fundo CBR com taxa de $120Mbps$. O policiamento foi desabilitado em todas as portas do comutador. O VC alocado em cada estação tem taxa máxima de $R = 27084$ células/s, equivalente à taxa de pico da fonte. Note que todos os VC(s) das estações estão conectados com PVC(s) VBR(s) previamente configurados no comutador ATM com a mesma taxa de pico 27084.

O experimento foi realizado para diversos tempos de geração $T = 30$, $T = 60$, $T = 120$, $T = 300$ e $T = 600$, mantendo os parâmetros do comutador $vbrob = 150$

e $vbrbuffob = 100$. Neste caso nenhuma perda foi observada, indicando que na utilização do algoritmo de admissão da seção 2.1, a QoS foi atingida. De acordo com este resultado, é possível admitir 3 fontes, garantindo a QoS desejada no comutador ATM.

A sequência das primeiras 40 variáveis aleatórias obtidas através do gerador de tráfego estão indicadas na Figura 5.5. Esta sequência foi gerada a partir de um gerador de tráfego modelando uma fonte on-off.

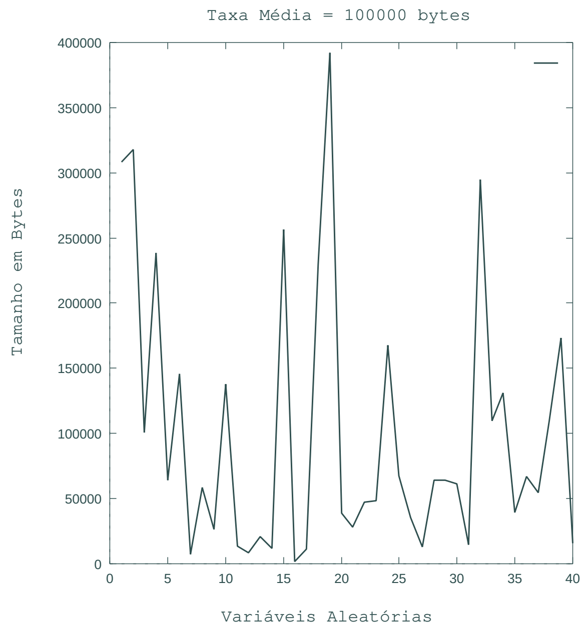


Figura 5.5: Variáveis aleatórias exponenciais com média 100000 bytes obtidas do gerador de tráfego

Para 4 fontes, o experimento foi realizado com diversos tempos de geração ($T = 30$, $T = 60$, $T = 120$, $T = 300$ e $T = 600$) mantendo os parâmetros do comutador $vbrob = 150$ e $vbrbuffob = 100$. Neste caso nenhuma perda foi observada, indicando que a QoS foi atingida.

Nas tabelas 5.1, 5.2, 5.3, 5.4 e 5.5 mostramos experimentos para 5 fontes com as mesmas características do experimento anterior, onde os parâmetros do comutador são $vbrob = 200$ e $vbrbuffob = 100$. Os geradores foram distribuídos em 4 estações, onde 3 estações ficaram com 1 gerador e 1 estação ficou com 2 geradores. Cada gerador sempre está associado a um único VC no comutador ATM. O tráfego de fundo CBR foi mantido como no experimento anterior e o policiamento também foi desligado nas portas no comutador. O experimento foi realizado com diversos tempos de geração ($T = 30$, $T = 60$, $T = 120$, $T = 300$ e $T = 600$).

A fração de perda FP (total de células perdidas d dividido pelo total de células geradas G, $FP = \frac{d}{G}$), foi calculada para cada uma das 5 fontes nas tabelas 5.1, 5.2, 5.3, 5.4 e 5.5. Os valores calculados destas frações são da ordem de 10^{-2}, indicando que a garantia de QoS para 5 fontes é violada no comutador ATM.

O algoritmo de CAC proposto por [17] mostrou-se conservador, visto que o comutador ATM garantiu QoS para 4 fontes em vez de 3 fontes como sugeridas pelo algoritmo. De qualquer forma, o algoritmo de CAC proposto por [17] pôde ser aplicado no comutador ATM e os parâmetros usados foram $vbrob = 150$ e $vbrbuffob = 100$.

VCs	**Tempo de Geração**	**Células perdidas**	**Células Geradas**	**FP**
VC1	30	27529	373616	0.0737
VC2	30	27489	373676	0.0736
VC3	30	27878	371714	0.0750
VC4	30	22107	373616	0.0592
VC5	30	22171	373616	0.0593

Tabela 5.1: Células perdidas por VC, medidas no comutador ATM para 5 fontes on-off em 30 segundos de geração

VCs	**Tempo de Geração**	**Células perdidas**	**Células Geradas**	**FP**
VC1	60	27706	756684	0.0366
VC2	60	32032	759017	0.0422
VC3	60	33357	752986	0.0443
VC4	60	27416	759017	0.0361
VC5	60	27274	759017	0.0359

Tabela 5.2: Células perdidas por VC, medidas no comutador ATM para 5 fontes on-off em 60 segundos de geração

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	120	47294	1495253	0.0316
VC2	120	54544	1502769	0.0363
VC3	120	54451	1495257	0.0362
VC4	120	45874	1502769	0.0305
VC5	120	47544	1502769	0.0316

Tabela 5.3: Células perdidas por VC, medidas no comutador ATM para 5 fontes on-off em 120 segundos de geração

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	300	157609	3872610	0.0407
VC2	300	173337	3875599	0.0447
VC3	300	227411	3872610	0.0587
VC4	300	251946	3884994	0.0649
VC5	300	251050	3875599	0.0648

Tabela 5.4: Células perdidas por VC, medidas no comutador ATM para 5 fontes on-off em 300 segundos de geração

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	600	219649	7762863	0.0283
VC2	600	370086	7762863	0.0477
VC3	600	377005	7769720	0.0485
VC4	600	256198	7770762	0.0330
VC5	600	260635	7770762	0.0335

Tabela 5.5: Células perdidas por VC, medidas no comutador ATM para 5 fontes on-off em 600 segundos de geração

5.2.2 QoS para Tráfegos Regulados pelo Leaky-Bucket

Nesta seção repetimos experimentos da seção anterior, onde as principais modificações são: o policiamento está habilitado em todas as portas do comutador ATM e o algoritmo de CAC teórico utilizado é o da seção 2.2 para tráfegos-regulados proposto por [31].

Utilizando o módulo de admissão implementado, selecionamos o algoritmo teórico de CAC da seção 2.2 e calculamos a *capacidade efetiva* de uma fonte regulada pelo leaky-bucket com taxa de pico $R = 27084$, taxa média $m = 12950$ e $b = 1095$ células (b é o tamanho máximo da rajada). A QoS desejada é $\epsilon = 10^{-5}$, o tamanho do buffer $X = 17805$ e a capacidade de transmissão $C = 70754$ células/s. A *Capacidade efetiva* calculada para uma fonte é $\tilde{c}_i = 12950$ células/s e 5 fontes são admitidas, onde a *Capacidade efetiva agregada* é $\tilde{C} = 64750$ células/s.

Configuramos os parâmetros do comutador $vbrob = 200$ e $vbrbuffob = 100$, onde são admitidos 5 VC(s) VBR(s) com taxa de pico $R = 27084$ células/s, taxa média $m = 12950$ células/s e $b = 1095$ células. Também configuramos um tráfego de fundo CBR com taxa de 120Mbps para reduzir a banda total nos VCs VBRs.

Os 5 VC(s) foram distribuídos em 4 estações, onde 3 estações ficaram com 1 gerador cada e 1 estação ficou com 2 geradores, além de uma quinta estação que ficou com um gerador de tráfego de fundo CBR ($120Mbps$). O policiamento foi habilitado em todas as portas do comutador. O VC alocado em cada estação tem taxa máxima de $R = 27084$ células/s, equivalente à taxa de pico da fonte. Note que todos os VC(s) das estações, estão conectados a PVC(s) VBR(s) previamente configurados no comutador ATM com a mesma taxa de pico. Cada gerador sempre está associado a um único VC no comutador ATM.

Foram gerados tráfegos para diversos tempos de geração ($T = 30$, $T = 60$, $T = 120$, $T = 300$ e $T = 600$). Neste caso nenhuma perda de célula foi observada, indicando que na utilização do algoritmo de CAC da seção 2.2, junto ao mecanismo de policiamento leaky-bucket habilitado nas portas do comutador, a QoS foi garantida. De acordo com este resultado, é possível admitir 5 fontes, garantindo a QoS desejada. O comutador Fore ASX-1000, implementa o mecanismo de policiamento leaky-bucket, padronizado como GCRA(Generic Cell Rate Algorithm) viabilizando a aplicação do algoritmo de CAC para tráfego regulado pelo leaky-bucket discutido na seção 2.2.

Nas tabelas 5.6, 5.7, 5.8, 5.9 e 5.10 mostramos experimentos para 6 fontes on-off com as mesmas características, onde os parâmetros do comutador são $vbrob = 250$ e $vbrbuffob = 100$. Os geradores foram distribuídos em 4 estações, onde 2 estações ficaram com 1 gerador cada e 2 estações ficaram com 2 geradores cada. O tráfego de fundo CBR foi mantido e o policiamento também foi habilitado nas portas no comutador. Cada gerador sempre está associado a um único VC no comutador ATM.

A fração de perda FP (total de células perdidas d dividido pelo total de células geradas G, $FP = \frac{d}{G}$), foi calculada para cada uma das 6 fontes nas tabelas 5.6, 5.7, 5.8, 5.9 e 5.10. Os valores calculados destas frações são da ordem de 10^{-2}, indicando que a garantia de QoS para 6 fontes é violada no comutador ATM.

O algoritmo de CAC proposto por [31] mostrou-se bem aplicável ao comutador ATM, garantindo a QoS para 5 fontes. Quando acrescentamos a sexta fonte, a FP ficou na ordem de 10^{-2}, indicando que para 6 fontes on-off reguladas pelo leaky-bucket, a garantia de QoS é violada no comutador ATM.

VCs	**Tempo de Geração**	**Células perdidas**	**Células Geradas**	**FP**
VC1	30	11333	373616	0.0303
VC2	30	13011	373676	0.0348
VC3	30	12078	371714	0.0325
VC4	30	11090	371714	0.0297
VC5	30	14303	373616	0.0383
VC6	30	10012	373216	0.0268

Tabela 5.6: Células perdidas por VC, medidas no comutador ATM para 6 fontes on-off com policiamento habilitado nas portas do comutador ATM, para 30 segundos de geração de tráfego

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	60	17706	756684	0.0234
VC2	60	20032	758200	0.0265
VC3	60	18357	752986	0.0243
VC4	60	19416	759017	0.0257
VC5	60	17274	759017	0.0228
VC6	60	13357	752300	0.0177

Tabela 5.7: Células perdidas por VC, medidas no comutador ATM para 6 fontes on-off com policiamento habilitado nas portas do comutador ATM, para 60 segundos de tempo de geração

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	120	31002	1502769	0.0206
VC2	120	28992	1495257	0.0194
VC3	120	25099	1502769	0.0167
VC4	120	25889	1502769	0.0172
VC5	120	27982	1501900	0.0186
VC6	120	30923	1495257	0.0207

Tabela 5.8: Células perdidas por VC, medidas no comutador ATM para 6 fontes on-off com policiamento habilitado nas portas do comutador ATM, para 120 segundos de tempo de geração

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	300	79839	3884994	0.0206
VC2	300	89228	3872610	0.0230
VC3	300	70233	3873833	0.0181
VC4	300	92838	3872920	0.0240
VC5	300	91292	3875599	0.0236
VC6	300	82993	3881335	0.0214

Tabela 5.9: Células perdidas por VC, medidas no comutador ATM para 6 fontes on-off com policiamento habilitado nas portas do comutador ATM, para 300 segundos de tempo de geração

VCs	Tempo de Geração	Células perdidas	Células Geradas	FP
VC1	600	140832	7762863	0.0181
VC2	600	159233	7770762	0.0205
VC3	600	139344	7769720	0.0179
VC4	600	138492	7769009	0.0178
VC5	600	159832	7762863	0.0206
VC6	600	147293	7766552	0.0190

Tabela 5.10: Células perdidas por VC, medidas no comutador ATM para 6 fontes on-off com policiamento habilitado nas portas do comutador ATM, para 600 segundos de tempo de geração

Capítulo 6

Conclusão

O principal objetivo do nosso estudo foi analisar a eficiência de 2 algoritmos de CAC propostos na literatura em [17, 31], efetuando comparações baseadas em medições feitas em um comutador ATM Fore-Systems ASX-1000. Isto nos motivou a utilizar algoritmos de CAC baseados em parâmetros UPC, que são padronizados pelo ATM Forum e usados pelos comutadores ATM. Para a realização dos experimentos, construímos um ambiente de testes composto de um gerador de tráfego de uso geral com suporte IP, ATM-nativo e módulo de admissão que estão integrados à ferramenta de domínio público Tangram-II [7, 35].

Ao longo do trabalho apresentamos um conjunto de experimentações e ferramentas e podemos enumerar as principais contribuições:

- Implementação de um gerador de tráfego de uso geral com suporte IP, ATM-nativo. O gerador de tráfego é uma ferramenta flexível capaz de gerar pacotes para a rede em tamanho e intervalos distribuídos conforme especificação do usuário. O usuário pode optar em enviar dados sobre UDP/IP ou sobre ATM nativo através da camada de adaptação AAL5. O gerador suporta 2 modos de geração de pacotes (espalhado uniformemente no intervalo ou gerados no início do intervalo). É possível especificar 3 modelos de tráfego (CBR, Markoviano ou Trace) provendo uma enorme flexibilidade ao usuário na descrição do tráfego. Todos estes modelos podem ser obtidos na seção de *Modeling Environ-*

ment da ferramenta de análise e modelagem Tangram-II [7, 35]. A ferramenta desenvolvida possui ajuda através da linha de comando, além da interface integrada ao Tangram-II indicada na Figura 3.1, facilitando a especificação dos parâmetros necessários.

- Implementação de 2 algoritmos de CAC em um módulo de admissão. Este módulo simula o funcionamento de algoritmos de CAC de acordo com parâmetros fornecidos pelo usuário. É possível descrever as características do enlace ATM, através dos parâmetros de *Capacidade de Transmissão*, *Probabilidade de Perda Desejada (QoS)* e *Tamanho do Buffer*. A partir da especificação do enlace ATM cada algoritmo calcula a *Capacidade efetiva* e o *número de fontes admitidas* para fontes especificadas através de descritores de tráfego. O módulo de admissão também pode funcionar como um servidor que aguarda requisições de clientes em uma porta configurável, sendo facilmente extendível a outros aplicativos no futuro (por exemplo um servidor de voz ou de vídeo). O módulo de admissão também possui interface integrada a ferramenta de modelagem e análise Tangram-II indicada na Figura 3.4, facilitando a especificação dos parâmetros necessários.

- É possível ajustar os parâmetros $vbrob$ e $vbrbuffob$ e com isso modificar o número de fontes admitidas no comutador ATM Fore-Systems ASX-1000. As Tabelas 4.2,4.3,4.4 demonstram o comportamento destes parâmetros e servem como referência para modificações refinadas desses parâmetros do comutador ATM. Na última coluna das Tabelas anteriormente citadas, temos uma porcentagem da taxa de pico. Através desta coluna, é possível ajustar os parâmetros para qualquer tipo de fonte, visto que as porcentagens indicadas são relativas aos parâmetros $vbrob$ e $vbrbuffob$ e sendo assim, independentes do tipo do tráfego (contrato UPC). Desta forma, o administrador da rede que possui um comutador ATM ASX-1000 pode consultar estas Tabelas e ajustar o número de fontes dentro de um ambiente real de redes integradas.

- Através de medições e testes: os algoritmos de CAC estudados quando usados no comutador ATM provêm a QoS desejada, sendo assim, ambos poderiam ser aplicados no comutador ATM. O algoritmo de CAC proposto por [17] mostrou-se mais conservador em relação ao algoritmo proposto por [31], segundo as medições de QoS, visto que o algoritmo de CAC [17] sugeriu 3 fontes admitidas e vimos que até 4 fontes foram admitidas com garantias de QoS no comutador. Por outro lado, o algoritmo de CAC [31] mostrou-se melhor aplicável sugerindo exatamente 5 fontes admitidas com garantias de QoS no comutador, de forma que quando adicionamos a sexta fonte, perdas de células ocorreram. Outro ponto importante é que a banda residual (Capacidade de Transmissão menos Capacidade efetiva calculada) é maior no algoritmo proposto por [17] indicando que este resíduo foi suficiente para garantir QoS para mais uma fonte. Estes resultados foram obtidos para um número reduzido de fontes.

- Apresentação de uma metodologia de medição de tamanho de buffer no comutador ATM, através do crescimento do processo de perda de células. Vimos que o manual do fabricante deixou dúvidas quanto ao tamanho do buffer do comutador ATM ASX-1000, podendo este tamanho variar entre $16K$ e $64K$células. Através de medições constatamos que o tamanho do buffer medido em uma porta foi 17805($+_{-}$3651) células, sugerindo que o tamanho total do buffer em uma porta de saída do comutador ATM ASX-1000 em nosso laboratório tem $16K$células. Também observamos que a alocação de recursos em termos de capacidade e buffer no comutador ATM ASX-1000 é variável em função da taxa de pico e não é sensível a variação da taxa média ou tamanho máximo da rajada em um contrato UPC.

6.1 Trabalhos Futuros

Quando o número total de células em um buffer de saída de um comutador atinge um limiar pré-calculado, o mecanismo de EPD (Early Packet Discard) descarta inteiramente um novo pacote, quando este chega ao enlace, prevenindo que novos pacotes preencham rapidamente os buffers, onde quase certamente ocorreria perda de células deste novo pacote devido à falta de espaço para todo o pacote. EPD reserva o restante do espaço do buffer para pacotes que já tenham entrado no buffer, maximizando as chances destes pacotes serem transmitidos com sucesso.

Quando ocorre um overflow em um buffer e uma célula deve ser descartada, um mecanismo chamado de PPD(Partial Packet Discard) descarta todas as células deste pacote que ainda permanecem no buffer. Isto reduz o número de pacotes que devem ser retransmitidos. PPD é ativado somente quando um buffer de saída está completamente cheio.

Um estudo interessante poderia ser a análise da influência destes mecanismos nas garantias de QoS do comutador ATM, através de comparações entre o número de células perdidas e o número de pacotes perdidos, observando-se o impacto destas perdas em sessões multimídia. Um exemplo poderia ser a geração de um modelo de tráfego caracterizando uma transmissão de vídeo ou de voz.

Uma outra sugestão interessante poderia ser a extensão e integração do módulo de admissão com um servidor de vídeo. Um servidor de vídeo sob demanda poderia incorporar os algoritmos de admissão, permitindo um estudo comparativo da garantia de QoS oferecida pelos algoritmos de CAC quando aplicados a este servidor.

Apêndice A

Expressões para a Capacidade Efetiva

Neste apêndice apresentamos um exemplo passo a passo de como obter expressões para a *capacidade efetiva* proposto por [10, 11, 12] utilizando *Chernoff bound* [20] e discutido no livro [36]

Considere um modelo de *fila infinita* da Figura A.1 que representa um enlace com capacidade C cells/s em uma rede integrada. K fluxos de tráfego independentes estão compartilhando este *buffer*. A variável aleatória $q(t)$ representa a ocupação do *buffer* em intervalos discretos de tempo t. É adotado como unidade, o tempo necessário para servir C unidades de capacidade.

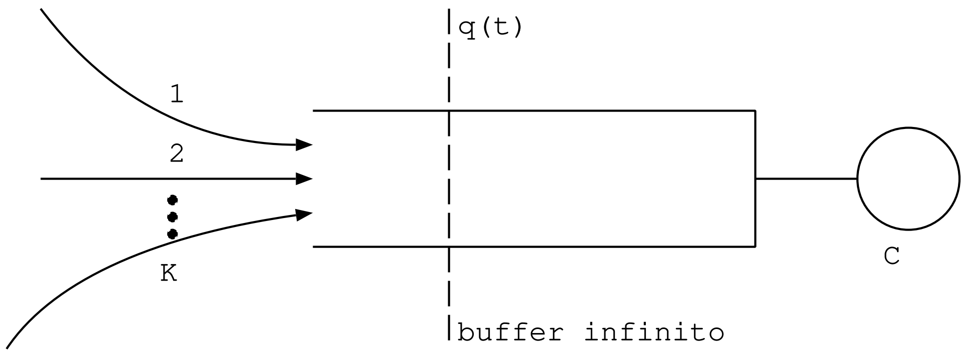


Figura A.1: Modelo de Fila para análise de *Capacidade Efetiva*

Considere agora, o intervalo $[t-1, t]$ mostrado na Figura A.2. Vamos denotar o número de chegadas (células) neste intervalo de uma fonte k pela variável aleatória $a_k(t)$. Logo o número total de chegadas em $[t-1, t]$ é:

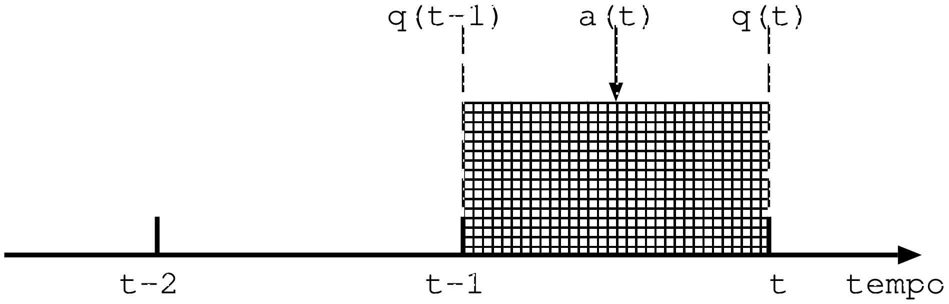


Figura A.2: Análise discreta de tempo

$$\mathbf{a}(t) = \sum_{k=0}^{K} \mathbf{a_k(t)} \tag{A.1}$$

É importante notar que $a(t)$ representa uma *taxa* , visto que é definido como sendo o número de chegadas em um intervalo. O número total de chegadas da fonte k no intervalo $[t1, t2]$, é uma outra variável aleatória, definida como:

$$\mathbf{A_k}(t1, t2) = \sum_{t=t1}^{t2-1} \mathbf{a_k(t)} \tag{A.2}$$

O tamanho da fila $q(t)$ depende destas chegadas. Para limitar $q(t)$ estatisticamente, ou seja, garantir uma fila estável, temos que limitar o número de chegadas $a(t)$ em um intervalo, bem como o número de chegadas $A(t1, t2)$ em um intervalo maior $[t1, t2)$.

Seja a transformada de Fourier(função geradora de momentos) de uma variável aleatória $x(t)$, dada por:

$$\mathbf{E}[\mathbf{e}^{\theta \mathbf{x(t)}}] \tag{A.3}$$

Onde θ é a variável da transformada.

Desta forma a *mgf(moment generator function)* do processo de chegada $A_k(t)$ é, por definição, limitada estocasticamente se:

$$\mathbf{E[e^{\theta A_k(t1,t2)}]} \leq e^{\theta \tilde{A}_k(\theta, t2-t1)} \tag{A.4}$$

A função geradora de momentos *mgf* é por definição exponencialmente limitada. Aplicando o logaritmo natural em ambos os lados, teremos:

$$\ln \mathbf{E[e^{\theta A_k(t1,t2)}]} \leq \theta \tilde{A}_k(\theta, t2-t1) \tag{A.5}$$

Ou seja, o logaritmo natural da *mgf* do processo de chegada k é limitado por uma função de θ e $t2-t1$.

A função do lado direito da equação é chamada de *processo envelope* de $A_k(t1,t2)$. O *processo envelope* é assumido como estacionário no tempo, dependendo de $(t2-t1)$ apenas, e não de $t1$ e $t2$ individualmente. Como um caso particular de um *processo envelope*, temos o seguinte:

$$\mathbf{\tilde{A}_k(\theta, t2-t1) = \tilde{a_k}(\theta)(t2-t1) + \tilde{\sigma_k}(\theta)} \tag{A.6}$$

A equação acima é chamada *processo envelope linear.* Note que $\tilde{a}_k(\theta)$ é um parâmetro de taxa. Informalmente, $A_k(t)$ é (embora expressado probabilisticamente) limitada por uma taxa máxima de chegada $\tilde{a}_k(\theta)$.

Com este limite na função geradora de momento de $A_k(t1,t2)$, podemos proceder para estabelecer um limite na variação da ocupação da fila $q(t)$, que resulta na definição da *capacidade efetiva* do fluxo de tráfego k. É necessário relacionar $q(t)$ com o processo de chegada agregado $a(t) = \sum_{k=1}^{K} a_k(t)$. A evolução da ocupação do buffer é dada pela equação:

$$\mathbf{q(t) = max(0, q(t-1) + a(t) - C)} \tag{A.7}$$

Seguindo [11] e expandindo $q(t)$ recursivamente, começando com fila vazia e tempo $t = 0$, obtemos:

$$\mathbf{q(t) = max_s(A(t-s) - sC)} \tag{A.8}$$

Fazendo *mgf* de $q(t)$ e usando a relação anterior A.8 , temos:

$$\mathbf{E[e^{\theta q(t)}] = E[e^{\theta max_s[A(t-s,t)-sC]}] = max_s E[e^{\theta(A(t-s,t)-sC)}]} \tag{A.9}$$

É possível eliminar a operação de max seguindo a inequação:

$$\mathbf{max(x_1, x_2, ..., x_n) \leq \sum_{j=1}^{n} x_j, \qquad x_j \geq 0} \tag{A.10}$$

Visto que a *mgf* é não-negativa, e usando A.10 em A.9, obtemos a equação:

$$\mathbf{E[e^{\theta q(t)}] \leq \sum_{s=0}^{t} E[e^{\theta[A(t-s,t)-sC]}]} \tag{A.11}$$

Note que assumimos que todos os K processos de chegada são independentes. Temos o seguinte:

$$\mathbf{A(t1, t2) = \sum_{k=1}^{K} A_k(t1, t2)} \tag{A.12}$$

Utilizando a definição em A.4, temos a seguinte equação:

$$\mathbf{E[e^{\theta(A(t-s,t)-sC)}] = [\prod_{k=1}^{K} E[e^{\theta(A_k(t-s,t))}]]e^{-\theta sC} \leq [\prod_{k=1}^{K} e^{\theta \tilde{A}_k(\theta,s)}]e^{-\theta sC}} \tag{A.13}$$

É importante notar que o termo constante $e^{-\theta sC}$ pode tomar valores fora do esperado. Seja a função envelope composta:

$$\mathbf{\tilde{A}(\theta, t) = \sum_{k=1}^{K} \tilde{A}_k(\theta, t)} \tag{A.14}$$

Isto torna possível a simplificação da equação A.13:

$$\mathbf{E}[\mathbf{e}^{\theta(\mathbf{A}(\mathbf{t}-\mathbf{s},\mathbf{t})-\mathbf{sC})}] \leq \mathbf{e}^{\theta[\tilde{\mathbf{A}}(\theta,\mathbf{s})-\mathbf{sC}]} \tag{A.15}$$

Podemos também reescrever o *processo linear composto*

$$\tilde{\mathbf{A}}(\theta, \mathbf{t}) = \tilde{\mathbf{a}}(\theta)(\mathbf{t}) + \tilde{\sigma}(\theta) \tag{A.16}$$

com

$$\tilde{\mathbf{a}}(\theta) = \sum_{\mathbf{k}=\mathbf{1}}^{\mathbf{K}} \tilde{\mathbf{a}_{\mathbf{k}}}(\theta) \tag{A.17}$$

e

$$\tilde{\sigma}(\theta) = \sum_{\mathbf{k}=\mathbf{1}}^{\mathbf{K}} \tilde{\sigma_{\mathbf{k}}}(\theta) \tag{A.18}$$

Retornando para a equação A.13 e usando a equação A.15, o *processo envelope linear composto* fica:

$$\mathbf{E}[\mathbf{e}^{\theta\mathbf{q}(\mathbf{t})}] \leq \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{t}} \mathbf{e}^{\theta[\tilde{\mathbf{A}}(\mathbf{t}-\mathbf{s},\mathbf{t})-\mathbf{sC}]} = \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{t}} \mathbf{e}^{\theta[\tilde{\mathbf{a}}(\theta)\mathbf{s}+\sigma(\theta)]}\mathbf{e}^{-\theta\mathbf{sC}} \tag{A.19}$$

Reescrevendo a equação acima de uma outra forma, temos:

$$\mathbf{E}[\mathbf{e}^{\theta\mathbf{q}(\mathbf{t})}] \leq \mathbf{e}^{\theta\tilde{\sigma}(\theta)} \sum_{\mathbf{s}=\mathbf{0}}^{\mathbf{t}} \mathbf{e}^{\theta\mathbf{s}[\tilde{\mathbf{a}}(\theta)-\mathbf{C}]} \tag{A.20}$$

A soma do lado direito da equação irá divergir se $\tilde{a}(\theta) \geq C$. A *mgf* de $q(t)$ será limitada se e somente se

$$\tilde{\mathbf{a}}(\theta) < \mathbf{C} \tag{A.21}$$

Examinando o significado da equação A.20, é importante notar que $\tilde{a}(\theta)$ representa o termo de taxa do *processo envelope linear*. Equação A.20 garante que a fila estará *estocasticamente limitada*, com o limite da *mgf* de $q(t)$, se e somente se a taxa $\tilde{a}(\theta)$ é menor que a capacidade do link C. Mais genericamente, esta abordagem define a *capacidade efetiva* de qualquer fonte de tráfego satisfazendo algumas condições simples. Igualmente importante é o formato da equação A.17 que indica que a *capacidade efetiva* de várias fontes independentes é a soma das capacidades individuais. A idéia é determinar a *capacidade efetiva* de modelos de fontes individuais e somar as capacidades para obter a *capacidade efetiva* de um modelo de fonte de tráfego composto muito mais complicado. Existe também algumas limitações:

- O resultado que será obtido vale para apenas uma fila.
- Todos os tráfegos de entrada são assumidos ter a mesma QoS (probabilidade de perda).
- A expressão da *capacidade efetiva* revela ser uma estimativa *conservadora* de capacidade, visto que a expressão ignora o efeito *smoothing*, implícito na multiplexação estatística.

Deixando $t \to \infty$ na equação A.20, temos:

$$\mathbf{E}[\mathbf{e}^{\theta \mathbf{q(t)}}] = \lim_{\mathbf{t} \to \infty} \mathbf{E}[\mathbf{e}^{\theta \mathbf{q(t)}}] \leq \mathbf{W}(\theta) \tag{A.22}$$

onde

$$\mathbf{W}(\theta) = \mathbf{e}^{\theta \tilde{\sigma}(\theta)} \sum_{\mathbf{s=0}}^{\infty} \mathbf{e}^{\theta \mathbf{s}[\tilde{\mathbf{a}}(\theta) - \mathbf{C}]} \simeq \frac{\mathbf{e}^{\theta \tilde{\sigma}(\theta)}}{\mathbf{1} - \mathbf{e}^{-\theta(\mathbf{C} - \tilde{\mathbf{a}}(\theta))}} \tag{A.23}$$

O lado direito da equação A.22 é a *mgf* em regime *estacionário* do tamanho da fila q, e determina que A.22 é limitada se $\tilde{a}(\theta) < C$.

Para limitar estatisticamente o tamanho da fila denotado pela variável aleatória q, é utilizado o chamado *Chernoff bound* [32]. Este limite relaciona a probabilidade de uma variável aleatória exceder um valor fixo, com a função geradora de momento desta variável aleatória. Estas classes de limites são relacionadas às inequações de Tchebycheff [32]. Logo temos:

$$\mathbf{P}[\mathbf{q} > \mathbf{x}] \leq \mathbf{e}^{-\theta \mathbf{x}} \mathbf{E}[\mathbf{e}^{\theta \mathbf{q}}] \tag{A.24}$$

Assumindo a *taxa envelope linear* $\tilde{a}(\theta) < C$, a equação A.22 é satisfeita, e inserindo em A.23,

$$\mathbf{P[q > x] \leq W(\theta)e^{-\theta x} = e^{-\theta x + \ln W(\theta)}} \tag{A.25}$$

Fazendo $\theta x >> 1$ e $P_L << 1$, *o modelo de fila infinita* temos:

$$\mathbf{P_L \rightarrow e^{-\theta x}} \tag{A.26}$$

Fazendo $W(\theta)$ igual a 1 na equação A.25, é possível encontrar um valor para o parâmetro θ em função do tamanho do *buffer* x e probabilidade de perda de células P_L.

$$\theta = \frac{\mathbf{1}}{\mathbf{x}} \ln(\mathbf{1/P_L}) = \frac{\mathbf{-lnP[q(\infty) > x]}}{\mathbf{x}} \tag{A.27}$$

A prova desta aproximação do limite exponencial para o tamanho da fila pode ser encontrada em [11].

É necessário estabelecer uma relação com a condição de estabilidade do buffer $\tilde{a}(\theta) < C$ aos processos conhecidos. Inicialmente é assumido que os processos acessando o buffer da Figura A.1 são estacionários com o tempo. Então:

$$\mathbf{A_k(t1, t2) = A_k(t2 - t1)} \tag{A.28}$$

Através da equação A.5, focando no intervalo $(0, t)$, temos:

$$\mathbf{\tilde{A_k}(\theta, t) \geq \frac{1}{\theta} \ln(E[e^{\theta A_k(t)}])} \tag{A.29}$$

O *processo envelope linear* tomado sobre o intervalo $(0, t)$, resulta em:

$$\mathbf{\tilde{A_k}(\theta, t) = \tilde{a_k}(\theta)t + \tilde{\sigma_k}(\theta)} \tag{A.30}$$

Tomando o limite $t \rightarrow \infty$, e isolando a taxa $\tilde{a}_k(\theta)$, teremos:

$$\tilde{\mathbf{a_k}}(\theta) = \lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \tilde{\mathbf{A_k}}(\theta, \mathbf{t}) \geq \lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \frac{\mathbf{1}}{\theta} \ln \mathbf{E}[\mathbf{e}^{\theta \mathbf{A_k(t)}}] \tag{A.31}$$

Mas sabemos que a *taxa envelope linear composta* $\tilde{a}(\theta)$ foi definida como a soma das taxas individuas $\tilde{a_k}(\theta)$. Onde $\tilde{A}(\theta, t) = \sum_{k=1}^{K} \tilde{A}_k(\theta, t)$ e $A(t) = \sum_{k=1}^{K} A_k(t)$, com os processos de chegadas individuais $A_k(t)$ assumidos como independentes. Então podemos obter a seguinte inequação:

$$\tilde{\mathbf{a}}(\theta) = \sum_{\mathbf{k=1}}^{\mathbf{K}} \tilde{\mathbf{a_k}}(\theta) = \lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \tilde{\mathbf{A}}(\theta, \mathbf{t}) \geq \lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \frac{\mathbf{1}}{\theta} \ln \mathbf{E}[\mathbf{e}^{\theta \mathbf{A(t)}}] \tag{A.32}$$

A condição de estabilidade do buffer é $\tilde{a}(\theta) < C$, combinada com a inequação A.32, resulta em:

$$\lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \frac{\mathbf{1}}{\theta} \ln \mathbf{E}[\mathbf{e}^{\theta \mathbf{A(t)}}] \leq \tilde{\mathbf{a}}(\theta) < C \tag{A.33}$$

O lado esquerdo da inequação é o que definimos como *capacidade efetiva* Ce do tráfego agregado acessando o buffer do multiplexador estatístico da Figura A.1. Se o fluxo agregado possui a propriedade expressa pela lado esquerdo da equação A.32, então a estabilidade do buffer é assegurada. A expressão da *capacidade efetiva* é:

$$\mathbf{Ce} = \lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \frac{\mathbf{1}}{\theta} \ln \mathbf{E}[\mathbf{e}^{\theta \mathbf{A(t)}}] \tag{A.34}$$

Visto que:

$$\mathbf{Ce} = \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{Ce_k} \tag{A.35}$$

com Ce_k *capacidade efetiva* da fonte k, dada por:

$$\mathbf{Ce_k} = \lim_{\mathbf{t}\to\infty} \frac{\mathbf{1}}{\mathbf{t}} \frac{\mathbf{1}}{\theta} \ln \mathbf{E}[\mathbf{e}^{\theta \mathbf{A_k(t)}}] \tag{A.36}$$

Esta é a expressão final que viemos discutindo através desta seção.

A.0.1 Aplicação da Capacidade Efetiva

O parâmetro θ é conhecido pela equação A.27, e estabelece uma relação entre a *probabilidade de perda* P_L e *tamanho do buffer* x. Portanto, dado o valor de P_L e x determinamos o valor de θ e usamos a equação A.36 para calcular a *capacidade efetiva* da k-ésima fonte. Somando as capacidades, obtemos a *capacidade efetiva* de todas as K fontes. Se $Ce < C$ a nova conexão é aceita, caso contrário é rejeitada, lembrando que estamos tratando apenas de um único nó ATM.

Embora esta abordagem seja conceitualmente simples, e fácil de implementar, e assumindo que todas as fontes desejam uma mesma QoS em termos de *probabilidade de perda* P_L e *tamanho do buffer* x iguais, revela ser uma estimativa *conservadora* para a *capacidade efetiva.* Isto porque requer que a *probabilidade de perda* P_L seja assintoticamente "suficientemente pequena", *tamanho do buffer* x seja "suficientemente grande" e depende das características individuais das fontes. Finalmente, o resultado obtido foi desprezando o termo $W(\theta)$, na frente da equação do limite exponencial A.25, e isto elimina efetivamente a vantagem do efeito da *multiplexação estatística* (smoothing) na soma de K fontes independentes.

Considere alguns exemplos de aplicação da equação A.36.

Começando com um caso trivial, de uma fonte (CBR) de taxa R células/seg, sabemos que a fonte é determinística. O número de chegadas no intervalo $[0, t)$ é $A(t) = Rt$. Aplicando a equação A.36, a *capacidade efetiva* é R. Este é o valor esperado.

No caso de uma fonte Poisson com taxa média λ_k células/seg. A função geradora de momentos *mgf* de um processo de Poisson $A_k(t)$, usando transformada-z é :

$$\mathbf{G_{A_k(t)}(z) = E[e^{\theta A_K(t)}] = e^{-\lambda_k(1-z)}} \tag{A.37}$$

Para converter a *mgf* em função da transformada de Fourier, basta substituir z por e^{θ}. Logo teremos:

$$\mathbf{E}[\mathbf{e}^{\theta \mathbf{A}_\mathbf{K}(\mathbf{t})}] = \mathbf{e}^{-\lambda_\mathbf{k}(\mathbf{1}-\mathbf{e}^\theta)} \tag{A.38}$$

Aplicando o logaritmo, dividindo por t e θ e aplicando o limite quando $t \to \infty$, surge a expressão para a capacidade efetiva de uma fonte Poisson acessando o buffer:

$$\mathbf{C}_\mathbf{k} = \frac{\lambda_\mathbf{k}(\mathbf{e}^\theta - \mathbf{1})}{\theta} \tag{A.39}$$

É importante notar pela equação A.35, que a *capacidade efetiva* de multiplas fontes tipo Poisson, é dada pela soma das *capacidades efetivas* de cada fonte, sendo também obtida somando-se os parâmetros de taxa do processo de Poisson. Este é um resultado esperado, visto que a soma de chegadas Poisson é Poisson com taxa média de chegada $\lambda = \sum_{k=1}^{K} \lambda_k$ [5].

Vamos considerar alguns exemplos para ver o impacto da escolha da *probabilidade de perda* P_L e *tamanho do buffer* x na *capacidade efetiva* .

Fazendo $P_L = 10^{-5}$ e usando a equação A.27, temos $\theta = 11.5/x$. Fazendo $x >> 11.5$, então $\theta << 1$, $C_k = \lambda_k$. Portanto para um *buffer* suficientemente grande, a *capacidade efetiva* de uma fonte Poisson é a taxa média da fonte. Mas se $x = 10$, então $\theta = 1.15$ e $C_k = 2\lambda_k$ que é o dobro do caso anterior, ou seja, a capacidade alocada pode estar super estimando a fonte, quando o tamanho do *buffer* x não é "suficientemente grande".

Fazendo $P_L = 10^{-9}$ e usando a equação A.27, temos $\theta = 20.7/x$. Se queremos manter $C_k \leq 1.1\lambda_k$, que é não mais que 10% maior que a taxa Poisson λ_k, temos que ter $\theta \leq 0.2$, ou $x \geq 100$ células. Mais genericamente, se queremos $C_k \leq 1.1\lambda_k$, então $\theta \leq 0.2$, ou $x > 5ln(1/P_L)$ células. Mas, novamente, P_L deve ser "suficientemente pequena", ou x "suficientemente grande" para termos $\ln W(\theta) << \theta x$ aproximação da equação A.25.

Vantagens :

- *Complexidade de implementação* - É possível, em tempo constante *O(1)* , determinar a *capacidade efetiva* de uma fonte. Portanto satisfaz a necessidade de aplicações em tempo-real.

- *Escalabilidade* - É escalonável para um grande número de fontes, pois não guarda o estado individual de cada fonte. Apenas determina a *capacidade efetiva* e agrega com as demais fontes já admitidas.

- *Dependência do modelo de tráfego* - A expressão de *capacidade efetiva agregada* é linear e cada fonte é independente, portanto a *capacidade efetiva agregada* é a soma das *capacidades efetivas individuais*. Mas as *capacidades efetivas individuais* necessitam de uma expressão para determinada fonte.

- *Garantia de QoS* - A probabilidade de perda requisitada por uma fonte é garantida, apesar de alocar banda maior que a banda realmente utilizada pela fonte.

Desvantagens:

- *Eficiência de banda* - A capacidade alocada para uma fonte, geralmente é conservadora.

- *Condições necessárias* - A condição necessária para aproximação ser boa, é o tronco ATM possuir *buffer "suficientemente grande"* e a *probabilidade ser "suficientemente pequena"* e este pode não ser o caso do nó ATM e dos requisitos de QoS da aplicação.

- *Expressão de Capacidade efetiva resultante* - A expressão requer que seja encontrada a *mgf* da fonte em questão. Nem sempre é possível achar uma boa aproximação para a *mgf* de uma fonte. Logo a expressão pode ser limitada a um grupo de fontes para as quais existam aproximações ou expressões fechadas para as *mgf* respectivas.

- *Efeito de amortização implícito na multiplexação estatística* - Como a expressão resultante simplificou $W(\theta)$ para 1, ou seja, desprezou o termo $W(\theta)$ na equação do limite exponencial A.25, isto eliminou o efeito da multiplexação estatística na soma de K fontes independentes.

Referências Bibliográficas

[1] Almesberger, W. URL ftp://lrcftp.epfl.ch/pub/linux/atm/. ATM on Linux release 0.4.

[2] Almesberger, W. URL http://lrcwww.epfl.ch/linux-atm/. ATM support for Linux.

[3] Balaji, e Dinkel, W. URL http://hegel.ittc.ukans.edu/projects/kurt/. The KU Real-Time Linux.

[4] Beran, J., Sherman, R., Taqqu, M. S., e Willinger, W. Long-Range Dependence in Variable-Bit-Rate Video Traffic. *IEEE Transactions on Communications 43*, 2/3/4 (1995), 1566–1579.

[5] Bertsekas, D., e Gallager, R. *Data Networks*, 2 ed. Prentice Hall, 1992.

[6] Butto, M. e. a. Effectiveness of the Leaky Bucket Policing Mechanism in ATM Networks. *IEEE Journal on selected areas in communications 9*, 3 (1991), 335–342.

[7] Carmo, R., de Carvalho, L., de Souza e Silva, E., Diniz, M., e Muntz, R. Performance/Availability Modeling with the TANGRAM-II Modeling Environment. *Performance Evaluation 33* (1998), 45–65.

[8] CCITT. Recommendation i.362, Adaptation Layer (AAL) Genova , 1992.

[9] CCITT. Recommendation i.363, Broadband ISDN Adaptation Layer(AAL) , 1992.

[10] Chang, C. Stability, Queue Length, and Delay. Stochastic Queueing Networks . *IBM Research* (1992).

[11] Chang, C. Stability, Queue Length, and Delay of Deterministic and Stochastic Queueing Networks . *IEEE ACM Transactions on Auto. Control 39*, 5 (1994), 913–931.

[12] Chang, C.-S., e Thomas, J. A. Effective bandwidth in high-speed digital networks. *IEEE Journal on Selected Areas in Communications 13*, 6 (1995), 1091–1100.

[13] Dapendg, W., e J., C. H. Efficient Bandwidth Allocation and Call Admission Control for VBR Service Using UPC Parameters. In *Infocom-99* (1999).

[14] Dziong, Z. A Framework for Bandwidth Management in ATM Networks-Aggregate Equivalent Bandwidth Estimation Approach. *IEEE ACM Transactions on Networking 5*, 1 (1997), 134–147.

[15] Elwalid, e Mitra, D. Effective Bandwidth of General Markovian Traffic Sources an Amission Control of High-Speed Networks. *IEEE ACM Transactions on Networking 1*, 3 (1993), 329–343.

[16] Grossglauser, M., e Tse, D. N. C. A Time-Scale Decomposition Approach to Measurement-Based Admission Control . In *Infocom-99* (1999).

[17] Guérin, R. H. A., e Naghshineh, M. Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks. *IEEE Journal on selected areas in communications 9*, 7 (1991), 968–981.

[18] ITU-T. Broadband aspects of isdn itu-t recommendation i.121, 1991.

[19] Kesidis G., J. W., e Chang, C. Effective Bandwidth for Multiclass Fluids and other ATM sources. *IEEE ACM Transactions on Networking 1*, 4 (1993), 424–428.

[20] Kleinrock, L. *Queueing Systems*, vol. 1 - Theory. John Wiley & Sons, 1975.

[21] Kobayashi, H., e Ren, Q. A mathematical theory for transient analysis of communication networks. *IECE Transactions on Communication E75-B*, 12 (1992), 1266–1276.

[22] Kohei Shiomoto, N. Y., e Takahashi, T. Overview of measurement-based connection admission control methods in ATM networks . *IEEE Communications Surveys 35* (1998).

[23] Kurz T., T. P., e Y, L. B. J. Regulation of a Connection Admission Control Algorithm . In *Infocom-99* (1999).

[24] Lee, T. K., e Zukerman, M. Efficiency Comparisons between different Model-Based and Measurement-Based Connection Admission Control Under Heavy Traffic . In *Globecom-99* (1999).

[25] Leocádio, M., e Rodrigues, P. H. A. Uma Ferramenta para Geração de Tráfego e Medição em Ambiente de Alto Desempenho . In *SBRC-2000* (2000).

[26] M., P. Asynchronous transfer mode solution for broadband isdn, 1991.

[27] M. Reisslein, K. W. R., e Rajagopal, S. Guaranteeing Statistical QoS to Regulated Traffic: The Single Node Case. In *Infocom-99* (1999).

[28] Mayor, G., e Silvester, J. Providing QoS for Long-Range Dependent Traffic. In *CAMAD-98* (1998), pp. 19–26.

[29] Mitra, D. Stochastic theory of a fluid model of producers and consumers coupled by a buffer. *Adv. Appl. Prob 20* (1988), 646–676.

[30] Mitra, D. A. D., e Sondhi, M. M. Stochastic theory of a data-handling system with multiple sources. *The Bell System Technical Journal 61*, 8 (1982), 1871–1894.

[31] Mitra, E. D., e H., R. A New Approach for Allocating Buffers and Bandwidth to Heterogeneous, Regulated Traffic in an ATM Node. *IEEE Journal on Selected Areas in Communications 13*, 6 (1995), 1115–1127.

[32] Papoulis, A. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 1991.

[33] Perros, H. G., e Elsayed, K. M. Call Admission Control Schemes: A Review . *IEEE Communications Magazine 10*, 11 (1996), 82–91.

[34] R.G. Garroppo, S. G., e Pagano, M. A CAC Algorithm for Per-VC Queueing Systems Loaded by Fractal Traffic . In *Globecom-99* (1999).

[35] Rosa M.M. Leão, E. d. S. e. S., e de Lucena, S. C. A Set of tools for Traffic Modelling, Analysis and Experimentation. In *Proceedings of TOOLS'2000* (2000).

[36] Schwartz, M. *Broadband Integrated Networks.* Prentice Hall, 1996.

[37] Stevens, W. R. *UNIX Network Programming - Network APIs:Sockets and XTI.* Prentice-Hall, 1998.

[38] Systems, F. URL http://www.fore.com/products/wp/index.html. Fore Systems white papers.

[39] Wang, J., e Erramilli, A. A Connection Admission Control Algorithm for Self-Similar Traffic . In *Globecom-99* (1999).

[40] Zhang, H. Service Disciplines for Guaranteed Performance Service in Packet-switched Networks . *IEEE Proceedings 83*, 10 (1995), 1374–1396.