FEEC
Faculdade de Engenharia Elétrica e de Computação

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Telemática

# Estudo Comparativo de Algoritmos de CAC para Redes ATM

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

**Niudomar Siqueira de Araújo Chaves**

Graduado em Engenharia Elétrica pela Universidade Federal do Ceará

**Banca Examinadora:**

Prof. Dr. Shusaburo Motoyama - Orientador
FEEC/UNICAMP

Dra. Eunice Luzivotto Medina Pissolato
Fundação CPqD

Prof. Dr. Akebo Yamakami
FEEC/UNICAMP

Prof. Dr. João Bosco Ribeiro do Val - Suplente
FEEC/UNICAMP

Este exemplar corresponde a redação final da tese defendida por Niudomar Siqueira de Araújo Chaves e aprovada pela Comissão Julgada em 29 / 10 / 1999

Orientador

Outubro de 1999

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C392e

Chaves, Niudomar Siqueira de Araújo
Estudo comparativo de algoritmos de CAC para redes ATM / Niudomar Siqueira de Araújo Chaves.-- Campinas, SP: [s.n.], 1999.

Orientador: Shusaburo Motoyama
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Comutação de pacotes (Transmissão de dados). 2. Rede digital de serviços integrados. 3. Telecomunicações. I. Motoyama, Shusaburo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

## Resumo

Este trabalho consiste no estudo comparativo de três algoritmos propostos na literatura para a implementação de Controle de Admissão de Conexões (CAC - *Connection Admission Control*) em redes ATM (*Asynchronous Transfer Mode*). Utilizando um simples modelo de multiplexador para um nó de rede ATM, os três algoritmos são comparados através de simulações. O modelo de fonte utilizado é do tipo fluido markoviano de dois estados, ou *on-off*. Baseado em um critério de tentativas exaustivas, determina-se o limite superior de aceitação de conexões, obedecendo a uma determinada taxa de perda de células. Os três algoritmos são comparados com relação a esse limite superior. São comparados, também, a utilização do enlace e o tempo médio de espera das células no *buffer*. Os resultados indicam que os algoritmos são, em geral, conservadores.



## Abstract

This work is a comparative analysis of three algorithms proposed in the literature for Connection Admission Control (CAC) implementation in ATM (Asynchronous Transfer Mode) networks. By using a simple ATM multiplexer model, the three algorithms are compared through simulation. A two state markovian fluid source model (on-off) is used. An upper limit for the connection acceptance that obey a specified cell loss ratio is estimated through exhaustive trials technique. The algorithms are compared using this upper limit. Also, utilization and mean waiting time in buffer are compared. Results show that the algorithms are, usually, conservative.

Este trabalho é dedicado a minha mãe, Raimunda Siqueira.

# Agradecimentos

Aos meus pais e irmãos, pelo apoio, confiança e encorajamento sempre presentes.

A Ricardo e Vilene, as pessoas mais importantes com as quais convivi durante o período de desenvolvimento deste trabalho.

Ao professor Shusaburo Motoyama, por sua orientação e ajuda em todos os momentos em que foi necessário.

Aos inúmeros amigos que fiz na cidade de Campinas e que tornaram a permanência aqui muito mais prazerosa.

# Trabalho Publicado

N. S. A. Chaves e S. Motoyama, "Estudo de desempenho de 2 algoritmos de CAC para redes ATM," Anais do XVII Simpósio Brasileiro de Telecomunicações, pp. 654-659, 1999.

# ÍNDICE

# Lista de Acrônimos

AAL - ATM Adaptation Layer

ABR - Available Bit Rate

ATM – Asynchronous Transfer Mode

B-ISDN – Broadband ISDN

BT – Burst Tolerance

CAC – Connection Admission Control

CBR – Constant Bit Rate

CDV – Cell Delay Variation

CDVT – CDV Tolerance

CLP – Cell Loss Priority

CLR – Cell Loss Ratio

CTD – Cell Transfer Delay

GCRA – Generic Cell Rate Algorithm

IBT – Intrinsic Burst Tolerance

ISDN – Integrated Services Digital Network

ISO – International Standards Organization

ITU-T – International Telecommunication Union
Telecommunication Standardization Sector

MaxCTD – Maximum Cell Transfer Delay

MBS – Maximum Burst Size

MCR – Minimum Cell Rate

NNI – Network-Node Interface

NPC – Network Parameter Control

OAM – Operation And Maintenance

OSI – Open Systems Interconnection

PCR – Peak Cell Rate

PDH – Plesiochronous Digital Hierarchy

PVC – Permanent Virtual Circuit

QoS – Quality of Service

RDSI – Rede Digital de Serviços Integrados, o mesmo que ISDN

RM – Resource Management

SBR – Statistic Bit Rate

SCR – Sustainable Cell Rate

SDH – Synchronous Digital Hierarchy

SONET – Synchronous Optical Network

SVC – Switched Virtual Circuit

UBR – Unspecified Bit Rate

UNI – User-Network Interface

UPC – Usage Parameter Control

VBR – Variable Bit Rate

VCC – Virtual Channel Connection

VPC – Virtual Path Connection

# CAPÍTULO 1

# INTRODUÇÃO GERAL

O modo de transferência assíncrono (ATM -*Asynchronous Transfer Mode*) foi a tecnologia escolhida pela ITU-T (International Telecommunications Union – Telecommunications Standardization Sector) para a implementação da rede digital de serviços integrados de faixa larga (RDSI-FL), que deve ser capaz de transportar tráfego multimídia (voz, dados, vídeo e texto). O objetivo da RDSI-FL é transportar, simultaneamente, os vários tipos de tráfego de maneira eficiente, com padrões de interface bem definidas aplicáveis desde centrais públicas de comutação às redes locais e operando em velocidades da ordem de dezenas a centenas de *Mbps*.

O ATM é baseado na técnica de comutação por pacote. Para facilitar a comutação e a armazenagem, o pacote tem um comprimento fixo e é denominado de célula, para diferenciar do pacote tradicional. Uma célula é composta por um cabeçalho de 5 octetos (*bytes*) contendo as informações de comutação, seguidos de outros 48 octetos, onde são colocadas as informações a serem transportadas.

Para facilitar o encaminhamento das células, a rede ATM utiliza a comutação por circuito virtual, portanto é uma rede orientada à conexão. Uma rede orientada à conexão possibilita uma fase de negociação entre o usuário e a rede, que resulta no contrato de tráfego. Nesse contrato, o usuário especifica o tráfego desejado, quantificando os parâmetros como a taxa de pico, a taxa média, o comprimento máximo do surto e a variação do atraso de célula, e, por outro lado, a rede, ao aceitar a conexão, garante que é capaz de atender o tráfego especificado.

Supõe-se que os tráfegos multimídia gerem informações de maneira descontínua no tempo; uma seqüência de células é seguida por um período de silêncio. Para aproveitar estes instantes de silêncio, os comutadores ATM utilizam a multiplexagem estatística. O objetivo é obter a maior utilização possível da largura de banda de transmissão dos enlaces. O contrato de tráfego contém a especificação do valor do parâmetro quantificador da

qualidade de serviço (QoS - *Quality of Service*) a ser garantida pela rede. Esta não pode aceitar novas conexões indiscriminadamente, pois precisa manter a QoS das conexões. O compromisso entre a qualidade de serviço e a maior utilização possível da largura de banda dos enlaces exigiu o desenvolvimento de mecanismos de controle de tráfego e de congestionamento; o primeiro tem caráter preventivo e o segundo, reativo.

A decisão sobre a aceitação ou recusa de uma conexão é efetuada pela rotina de controle de admissão de conexões (CAC - *Connection Admission Control*), que faz parte do controle de tráfego. Um usuário, ao solicitar a conexão, fornece os parâmetros descritores de tráfego ao nó comutador. Este deve decidir se é capaz de atender a nova conexão sem que isso prejudique as já estabelecidas, ou seja, mantendo a QoS de todas. Este procedimento é executado em todos os nós comutadores, desde a origem da solicitação até o usuário final. A conexão só será efetivada se for aprovada em todos os nós de comutação da rede ao longo do caminho. Em sua maioria, o parâmetro utilizado para quantificar a QoS é a taxa de perda de células (CLR - *Cell Loss Ratio*), muito embora o atraso de transferência de célula e a variação do atraso de célula sejam parâmetros importantes para algumas aplicações, como telefonia e videoconferência.

O algoritmo de CAC deve ser simples o suficiente para permitir seu uso em tempo real nos comutadores ATM, que trabalham em altas velocidades. Entretanto, a variedade das fontes multimídia que exigem diferentes larguras de banda e o uso de multiplexagem estatística fazem com que os modelos de algoritmos de CAC sejam bastante complexos. Isso faz com que a maioria dos algoritmos propostos na literatura sejam baseados em aproximações. O problema é, então, avaliar se essas aproximações são boas.

Vários algoritmos têm sido propostos para solucionar o problema de Controle de Admissão de Conexões em redes ATM. Uma boa parte utiliza o conceito de banda efetiva [1], [2], que é a capacidade que deve ser alocada para o atendimento isolado de uma fonte. Dessa forma o efeito da multiplexagem com outras fontes é desconsiderado. Ainda assim é um método bastante atrativo pela simplicidade de implementação devido à sua característica de aditividade linear.

O tráfego contratado por uma conexão pode não se concretizar; é razoável supor que os usuários desejem uma pequena margem de largura de banda de reserva nas suas conexões. Adicionalmente, a previsão do algoritmo pode falhar por não utilizar um modelo

de tráfego que se aproxime o suficiente do real. Na tentativa de evitar recursos ociosos na rede surgiram propostas de CAC dinâmico [4], [7], [8]. A decisão sobre a aceitação é baseada no estado da rede, que tem o tráfego constantemente monitorado. Numa outra vertente existem métodos que implementam diferentes classes de serviço ou prioridade [5], [9]. Todos os estudos acima têm visão probabilística do problema; Le Boudec e Nagarajan [10] propuseram uma solução determinística através da consideração do tempo de atraso de célula no pior caso.

O objetivo deste trabalho consiste em avaliar, através de simulação de eventos discretos, os algoritmos baseados em banda efetiva propostos por Guérin, Ahmadi e Naghshineh [1], Kesidis, Walrand e Chang [2] e o algoritmo baseado em alocação dinâmica proposto por Saito e Shiomoto [4].

O capítulo 2 contém uma descrição das categorias de serviço definidas pelo ATM Forum. Os modelos de tráfego mais significativos para redes multimídia são discutidos em seguida. O capítulo encerra com um levantamento das propostas de algoritmos de CAC existentes na literatura.

Os algoritmos escolhidos para análise são discutidos em detalhe no Cap. 3. A idéia básica de cada proposta é apresentada e os parâmetros descritores de tráfego utilizados são ressaltados.

As simulações efetuadas e os resultados obtidos são apresentados no Cap. 4. Os detalhes de implementação das simulações são apresentados. Os resultados são divididos em dois grupos: um estudo de todos os algoritmos juntos e um estudo exclusivo do algoritmo de alocação dinâmica. O capítulo termina com uma análise da confiabilidade dos resultados.

O Cap. 5 encerra o trabalho com as conclusões finais.

# CAPÍTULO 2

# CONTROLE DE CONEXÕES EM REDES ATM

## 2.1 Introdução

Neste capítulo são discutidos os aspectos relativos ao controle de tráfego nas redes ATM. Primeiramente, é feita uma descrição das categorias de serviço especificadas pelo ATM Forum. Estas categorias foram criadas para permitir a negociação entre usuário e rede do tipo de serviço necessário para atender às exigências do tráfego. São indicados os parâmetros de tráfego e de qualidade de serviço (QoS – *Quality of Service*) utilizados por cada categoria de serviço.

Em seguida, é apresentada uma breve discussão das opções de modelos matemáticos para tráfego multimídia. O modelo *on-off* é analisado com maiores detalhes por ser utilizado nas simulações discutidas no Cap. 4.

Uma apresentação sucinta dos principais esquemas de CAC (*Connection Admission Control*) encerra o capítulo.

## 2.2 Categorias de Serviço – ATM Forum

Para atender variados tipos de fontes eficientemente, um conjunto de categorias de serviço foi definido pelo ATM Forum [20]. Cada categoria pretende atender de maneira adequada um tipo de tráfego com requisitos bem específicos. Diferentes compromissos de QoS (*Quality of Service*) são oferecidos em termos de tolerância a atraso e perda de acordo com as exigências da conexão. Cada categoria também difere na forma como é alocada banda e nas funções de gerenciamento de tráfego utilizadas.

### 2.2.1 Categoria de Serviço CBR (*Constant Bit Rate*)

A banda é alocada pela rotina de CAC (*Connection Admission Control*) e garantida durante toda a conexão. Esta categoria é adequada para atender tráfego em tempo real e taxa de transmissão constante, que necessita de limites rigosos de atraso de transferência de célula, CTD (*Cell Transfer Delay*), e tolerância da variação do atraso, CDVT (*Cell Delay Variance Tolerance*). A taxa a ser alocada é determinada pela taxa de pico, PCR (*Peak Cell Rate*).

A fonte pode transmitir em taxas inferiores a contratada a qualquer momento e por qualquer duração de tempo, mas a rede deve ser capaz de atender transmissões na taxa de pico durante toda a duração da conexão. O compromisso da rede é manter a QoS negociada desde que a taxa de transmissão não ultrapasse o valor contratado. Os parâmetros de QoS negociados são a taxa de perda de células, CLR (*Cell Loss Rate*), a variação de pico a pico do atraso de células, ppCDV (*peak-to-peak Cell Delay Variation*), e o atraso máximo de transferência, MaxCTD (*Maximum Cell Transfer Delay*).

### 2.2.2 Categoria de Serviço rt-VBR (*real-time Variable Bit Rate*)

A banda não é mais alocada num valor fixo, mas numa faixa especificada pelo limite superior, PCR, e pelo limite inferior, SCR (*Sustainable Cell Rate*). O serviço é adequado para fontes de tempo real que tenham necessidade de limites rígidos de atraso de transferência e variação do atraso, mas que geram taxas variáveis. Serviços de videoconferência devem utilizar esta categoria de serviço. Além dos parâmetros PCR e SCR, o serviço também é especificado pelo tamanho máximo do surto a ser gerado, MBS (*Maximum Burst Size*), e CDVT.

Esta categoria permite o uso de multiplexagem estatística pelos comutadores. Assim como o serviço CBR, não é possível utilizar *buffers* grandes, pois isso poderia provocar a violação dos parâmetros de atraso e variação do atraso. Os parâmetros de QoS negociados

são a taxa de perda de células CLR, a variação pico a pico do atraso ppCDV e o atraso máximo MaxCTD.

### 2.2.3 Categoria de Serviço nrt-VBR (*non-real-time Variable Bit Rate*)

Voltado para aplicações de taxa variável, mas que não necessitam de garantias quanto ao tempo de atraso, CTD, e a variação do atraso, CDV. Assim como seu equivalente de tempo real, rt-VBR, é especificado pelos descritores PCR, CDVT, SCR e MBS, mas não utiliza qualquer restrição ao tempo de transferência. Possibilita ganho estatístico, pois permite a alocação pelo CAC de uma taxa mais próxima à taxa média, SCR, e o uso de *buffers* maiores para suportar eventuais surtos longos. Transferência de dados utilizando a categoria de serviço nrt-VBR, pode ser bem modelada por uma fonte *on-off*. Durante o período de *on*, células são geradas com taxa PCR. O único parâmetro de QoS negociado é a taxa de perda de células CLR.

### 2.2.4 Categoria de Serviço UBR (*Unspecified Bit Rate*)

Não há restrições ao atraso e variação do atraso. Utiliza apenas a banda que está sobrando, sendo, por isso, chamado de serviço do "melhor esforço" (*best effort*). Não há garantias quanto à taxa de perda de células, CLR. A taxa de pico pode ou não ser utilizada pelo CAC e UPC (*Usage Parameter Control*). Este serviço é apropriado para a transferência de arquivos e correio eletrônico, por exemplo.

### 2.2.5 Categoria de Serviço ABR (*Available Bit Rate*)

É a única categoria de serviço a permitir à rede alterar os limites contratados para a conexão após a fase de estabelecimento ter sido concluída. Durante a solicitação de conexão, são especificadas a taxa de pico, PCR, a tolerância ao atraso, CDVT, e a taxa

mínima aceitável pela fonte, MCR (*Minimum Cell Rate*). A rede pode informar a fonte através de células de gerenciamento de recursos, RM (*Resource Management*), para diminuir sua taxa de transmissão. A taxa não pode ser reduzida abaixo da MCR especificada. A rede deve utilizar este artifício para evitar, ou amenizar, situações de congestionamento e garantir um baixo CLR para a conexão. Não há restrições quanto ao tempo de atraso e sua variação.

### 2.2.6 Categoria de Serviço GFR (*Guaranteed Frame Rate*)

Esta categoria foi desenvolvida para atender PDU's (*Protocol Data Unit*) de camadas mais altas. Específica um valor mínimo para a banda a ser utilizada. A rede pode descartar PDU's inteiras, ao invés de células, no caso de congestionamento. Este serviço ainda está em processo de especificação e suas características são passíveis de mudança.

## 2.3 Modelagem de Tráfego

Os primeiros estudos de tráfego foram feitos em redes de telefonia. O problema consistia em alocar de maneira eficiente os recursos da rede (troncos) levando em conta o comportamento estatístico das chegadas de chamadas. Como uma conexão telefônica utiliza uma largura fixa de banda, o conhecimento da quantidade e duração das chamadas foi suficiente para determinar os recursos necessários.

Por sua vez, o tráfego multimídia pode necessitar de uma ampla gama de valores de banda. Durante a transmissão de vídeo codificado, a largura de banda necessária pode variar por uma ordem de dez. Comunicações de dados, normalmente, ocorrem em períodos de surtos e silêncio. O interesse não está apenas no número e duração das chamadas, mas também nas propriedades estatísticas do fluxo de informações durante a chamada, para possibilitar o uso eficiente de recursos ao mesmo tempo em que é garantida a qualidade de serviço.

A fonte *on-off* é um dos modelos mais utilizados nos estudos de desempenho de redes. Seu principal atrativo é a capacidade de representar tráfego intermitente, que alterna surtos e períodos de silêncio, aliada à sua simplicidade. A informação é enviada numa sucessão de períodos ativos (*on*) intercalados com períodos de silêncio (*off*).

Dentre as possibilidades para a taxa de transmissão do período *on*, as mais usadas são a taxa constante (determinística), processo de Bernoulli (discreto no tempo) e Poisson (contínuo no tempo).

Se os estados de *on* e *off* são comutados de acordo com uma cadeia de Markov, contínua ou discreta no tempo, de dois estados, temos um caso particular em que um dos estados tem taxa nula. Neste caso, a fonte é classificada de acordo com a característica de transmissão no período *on*. Se a transmissão durante o período ativo obedece a um processo de Poisson, a fonte é chamada de processo interrompido de Poisson (IPP – *Interrupted Poisson Process*). No caso discreto, processo de Bernoulli, é chamada de processo interrompido de Bernoulli (IBP – *Interrupted Bernoulli Process*). Se a taxa for constante, passa a ser chamada processo de fluido interrompido (IFP – *Interrupted Fluid Process*).

Na fonte IPP, os tempos de *on* e *off* são exponencialmente distribuídos, enquanto que para a fonte IBP a distribuição é geométrica. Essas fontes não representam a característica de autocorrelação do tráfego, pois os tempos entre chegadas sucessivas são independentes entre si, ou seja, o tempo entre chegadas é um processo de renovação.

Fontes mais gerais podem ser utilizadas para representar a autocorrelação. As fontes que podem atender essa necessidade são do tipo processo de Poisson modulado por Markov (MMPP), ou seus equivalentes discretos no tempo, processo de Bernoulli modulado por Markov (MMBP) ou processo de fluido modulado por Markov (MMFP). Um MMPP é composto de vários estados. Em cada estado, as chegadas ocorrem segundo um processo de Poisson onde a taxa é dependente do estado. As fontes MMBP e MMFP são similares, apenas diferindo no processo que caracteriza a geração de células em cada estado: Bernoulli ou fluido, no lugar de Poisson.

Recentemente, medidas de tráfego em redes de pacotes mostraram um comportamento de natureza auto-similar [13]. Para tráfego com dependência de curto prazo, é esperado que o decaimento da distribuição da probabilidade de um *buffer* sofrer

transbordamento seja exponencial. A dependência de longo prazo provoca um decaimento mais lento dessa distribuição e o aumento no tamanho do *buffer* não garante mais uma queda tão significativa na taxa de perda de células.

### 2.3.1 Modelo Matemático da Fonte *On-Off*

Nesta seção é apresentada a caracterização de uma fonte *on-off* discreta no tempo com períodos *on* e *off* geometricamente distribuídos [3]. O diagrama de transição de estados desta fonte pode ser visto na Fig. 2.1.

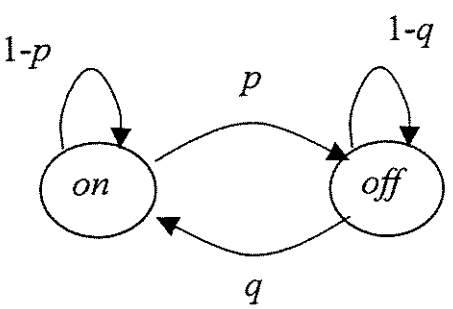


Figura 2.1: Diagrama de transição de estados de uma fonte *on-off*.

A matriz de transição de estados é

$$\Lambda = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}. \tag{2.1}$$

A distribuição de equilíbrio é dada por

$$P_{on} = \frac{q}{p+q};$$

$$P_{off} = \frac{p}{p+q}. \tag{2.2}$$

Define-se $p_n$ como a probabilidade do estado *on* ter comprimento $n$. Esta probabilidade é dada por

$$p_n = (1\text{-}p)^{n-1}p. \tag{2.3}$$

Dessa forma, o período *on* médio é

$$\overline{T}_{on} = \sum_{n=1}^{\infty} np_n = \frac{1}{p} \tag{2.4}$$

e,

$$\overline{T}_{off} = \frac{1}{q} \tag{2.5}$$

A carga $r$ da fonte é dada por

$$r = \frac{\overline{T}_{on}}{\overline{T}_{on} + \overline{T}_{off}}. \tag{2.6}$$

## 2.4 Controle de Admissão de Conexões

Uma conexão ATM passa por vários nós de comutação da rede. Para garantir a QoS (*Quality of Service*) contratada, cada nó deve reservar *buffer* e largura de banda para o atendimento da conexão. A decisão sobre a disponibilidade de recursos de um nó de comutação para aceitar uma nova conexão é função da rotina de controle de admissão de conexões CAC (*Connection Admission Control*). O controle de admissão de conexões é do tipo preventivo, pois tenta preservar a QoS das conexões existentes recusando o estabelecimento de novas chamadas, se estas puderem comprometer a qualidade das existentes.

A rotina de CAC não pode ser computacionalmente cara, pois é executada em tempo real pelos comutadores ATM. A velocidade de execução da rotina afeta diretamente a taxa de estabelecimento de conexões e a demora no estabelecimento porque o CAC é executado em todas as solicitações de conexão e em todos os comutadores ao longo do caminho. Como as conexões são estabelecidas e concluídas em tempo real, a rede deve ser capaz de

gerenciar uma taxa elevada de estabelecimento de conexões para o uso eficiente de SVC's (*Switched Virtual Circuits*).

A forma mais simples de decidir pela aceitação ou recusa de uma conexão é baseada na taxa de pico PCR. Se a banda disponível no enlace é superior ao valor PCR da conexão solicitada, aceita-se o estabelecimento, caso contrário, a conexão é recusada. Como a rede ATM deve atender tráfegos dos mais variados tipos, esse método de aceitação provocaria o uso ineficiente dos recursos disponíveis na rede, pois as fontes podem não transmitir na taxa PCR durante toda a duração da conexão. A tecnologia ATM utiliza o conceito de multiplexagem estatística e a rede pode se beneficiar dos períodos de silêncio, ou taxas abaixo da PCR das fontes para atender outras conexões. A principal missão da rotina de CAC é conseguir aceitar o maior número possível de conexões, baseando-se na multiplexagem estatística, garantindo a QoS de todas elas. Como uma forma de quantificar esse aproveitamento é definido o conceito de ganho estatístico:

$$\text{ganho estatístico} = \frac{\text{número de conexões admitidas com multiplexagem estatística}}{\text{número de conexões admitidas com alocação por taxa de pico}}$$

As categorias de serviço de tempo real CBR e rt-VBR têm exigências rigorosas quanto ao atraso de transferência de células, CTD, e à variação do atraso, CDV. O ITU-T especificou o limite de CTD para serviços de classe 1 em 400 ms, para conexões complexas até 27.500 Km em redes RDSI públicas [14]. Isto faz com que os *buffers* a serem utilizados sejam pequenos e o congestionamento a nível de células prevalece neste caso. Dessa forma, é difícil atingir um grande ganho estatístico para serviços CBR e rt-VBR. Nos serviços nrt-VBR, ABR e UBR não há exigência quanto ao tempo de atraso e sua variação, permitindo o uso de *buffers* grandes para armazenar os surtos. Neste caso, é possível atingir maior ganho estatístico e o congestionamento a nível de surtos prevalece. É importante salientar que a noção de grandeza de um *buffer* com relação ao tempo de atraso está intrinsicamente relacionada com a velocidade do enlace associado.

### 2.4.1 CAC para Tráfego CBR

A primeira solução aparente é utilizar a taxa de pico PCR como valor a ser alocado. Contudo, uma conexão compartilha os recursos com várias outras e a variação do atraso, CDV, pode inviabilizar a alocação por PCR. Há duas abordagens para solucionar o problema de considerar o efeito da CDV: métodos que desprezam a CDV e métodos que consideram a CDV [21].

**Métodos que Desprezam a CDV**

A CDV é considerada desprezível e o multiplexador pode ser modelado como uma fila M/D/1. Dado um *buffer* de tamanho $K$, capacidade do enlace $C$, e taxa de pico PCR, o método determina uma carga $r$ tal que a probabilidade do comprimento da fila exceder $K$ é menor que $\varepsilon$, onde $\varepsilon$ é um número pequeno como $10^{-10}$. A conexão é aceita de acordo com a seguinte equação:

$$\sum_i PCR_i \leq r \cdot C$$

Outro método consiste em utilizar um modelo de multiplexador nD/D/1 para representar $n$ fontes CBR idênticas sendo multiplexadas. Este modelo representa de maneira mais precisa e é menos conservativo que o modelo M/D/1. A medida que o número $n$ de conexões CBR aumenta, o tráfego agregado tende a se aproximar de uma distribuição poissoniana e o modelo M/D/1 apresenta resultados mais próximos da fila nD/D/1.

**Métodos que Consideram a CDV**

É razoável esperar que uma conexão CBR seja multiplexada ao longo da rede juntamente com outras conexões não periódicas como rt-VBR, por exemplo. Neste caso, a periodicidade da conexão CBR é afetada e a variação do atraso CDV passa a ter importância e deve ser considerada pela rotina de CAC. O ITU-T sugeriu um algoritmo a

ser utilizado no policiamento das conexões nas redes ATM: o algoritmo genérico de taxa de células GCRA (Generic Cell Rate Algorithm) [15]. Para conexões CBR atendidas por um enlace com capacidade $C$, o policiamento da taxa de pico é efetuado através de GCRA(1/PCR, CDVT). O padrão do tráfego de saída desta função de policiamento no pior caso é um processo periódico *on-off* com um comprimento máximo de surto $\mathrm{BS} = 1 + \lfloor \mathrm{CDVT}/(T-\delta) \rfloor$, onde $T$ = 1/PCR e $\delta$ = 1/$C$.

Para considerar o efeito da variação do atraso CDV uma restrição ao *buffer* é feita de forma que $\sum_i \mathrm{BS}_i \leq K$, onde $K$ é o tamanho do *buffer* e $\mathrm{BS}_i$ é o comprimento máximo de surto da conexão $i$. Esta abordagem assume perda de células nula no *buffer*, que é dimensionado para suportar a chegada simultânea de surtos com comprimentos máximos de todas as conexões ativas.

### 2.4.2 CAC para Tráfego de Taxa Variável

Foram feitas muitas propostas de algoritmos de CAC utilizando o conceito de banda efetiva. Esta consiste na taxa a ser alocada para uma conexão que é suficiente para garantir sua qualidade de serviço. A conexão é vista isoladamente, como se apenas ela estivesse ativa no enlace de saída.

Uma das primeiras propostas utilizando banda efetiva foi feita por Guérin, Ahmadi e Naghshineh [1]. A capacidade equivalente para uma fonte IFP é obtida a partir da distribuição da ocupação de um *buffer* finito. Uma aproximação é feita para permitir a obtenção de uma fórmula explícita para a capacidade equivalente. O método considera a possibilidade de o tráfego agregado necessitar de uma banda menor que a soma de todas as capacidades equivalentes das fontes conectadas. Para isso, é feita uma aproximação por distribuição normal e a capacidade equivalente escolhida é a de menor valor entre as duas alternativas.

Kesidis, Walrand e Chang [2] ampliaram a abrangência do conceito de banda efetiva ao mostrar sua validade para uma classe mais geral de fontes. O método é desenvolvido a

partir da suposição de que a probabilidade da ocupação de um buffer infinito ultrapassar um comprimento $K$ tem decaimento exponencial e da utilização de análise por teoria dos grandes desvios. Resultados são apresentados para fontes de taxa constante, Poisson, fontes markovianas discretas no tempo, fluidos de Markov e processos poissonianos modulados por Markov.

Liu, Petr e Braun [16] propuseram um esquema baseado em medidas do tráfego do usuário. Os parâmetros básicos de tráfego para serviços VBR são a taxa de pico PCR (*Peak Cell Rate*), a tolerância à variação do atraso CDVT (*Cell Delay Variation Tolerance*), a taxa média SCR (*Sustainable Cell Rate*), e a tolerância a surtos BT (*Burst Tolerance*). A proposta é dividida em duas partes. Na primeira parte, a largura de banda é alocada de acordo com os parâmetros SCR e BT. Na segunda parte, o desempenho da rotina é melhorado através da consideração da multiplexagem estatística. Existem várias combinações possíveis do conjunto (SCR, BT) que satisfazem os requisitos da conexão e ainda minimizam os recursos alocados. Para decidir qual combinação utilizar, o método usa duas técnicas: medições num *buffer* virtual e renegociação de parâmetros de UPC. A largura de banda efetivamente utilizada pelas conexões é estimada através de medidas para melhorar o desempenho da rotina.

Sohraby [17] propôs uma aproximação para tráfego pesado baseada no comportamento assimptótico da cauda da distribuição da ocupação do *buffer*. Resultados são apresentados para a superposição de fontes IBP e para fontes *on-off* onde os períodos de *on* e *off* têm distribuição arbitrária e são caracterizados pelos seus coeficientes quadrados de variação (SCV – *Squared Coefficient of Variation*). O autor afirma que a aproximação é eficiente para intensidade de tráfego $0.8 < r < 1$. Contudo, há uma grande dificuldade das redes ATM em operar nesta região devido aos requisitos de qualidade de serviço.

A alocação de banda utilizando os parâmetros descritores de tráfego fornecidos pelo usuário pode ser classificada como uma alocação estática. Como não há nenhuma verificação do estado real do tráfego na rede após a alocação, a banda reservada para a conexão estabelecida é decrementada da capacidade livre no enlace e só voltará a ser considerada disponível após o término da chamada. Se o tráfego contratado não se concretizar, a rede terá recursos ociosos. Saito e Shiomoto [4] propuseram um esquema de

alocação dinâmica, onde a aceitação é baseada nos parâmetros fornecidos pelo usuário e no tráfego efetivamente medido na rede. A distribuição da probabilidade de chegada de células num intervalo fixo $s$ é estimada através de medidas. A partir da distribuição e dos parâmetros de tráfego fornecidos pelo usuário é calculada a probabilidade de perda de células. Se o valor calculado for inferior que a qualidade solicitada, a conexão é efetivada.

Hsu e Walrand [8] propuseram dois esquemas de alocação dinâmica de banda para redes ATM. As conexões são divididas em classes e a banda é calculada para cada classe. O tráfego é medido durante um intervalo de amostragem $T$. Um contador é utilizado para representar a ocupação do *buffer*. Durante o período de amostragem $[(n\text{-}1)T, nT]$, o contador é decrementado a uma taxa fixa $\theta_{n-1}$, e incrementado a cada chegada de célula. O número de chegadas $A_n$ e o número de células perdidas $L_n$ durante este período são medidos usando outro conjunto de contadores. No instante $nT$, um novo valor de largura de banda é calculado para as conexões desta classe através da fórmula:

$$\theta_n = \theta_{n-1} + \frac{c}{n}(L_n - CLR \cdot A_n),$$

onde $c$ é uma constante positiva. Como a taxa de perda de células é um valor muito pequeno, o período de amostragem precisa ser longo. Por isso, o algoritmo é lento frente ao tráfego da rede. Para contornar esse problema um segundo método chamado monitoramento paralelo é proposto. O objetivo é determinar a menor largura de banda que satisfaz a taxa de perda de células desejada dentre um número finito de opções.

Dentre as propostas apresentadas acima, três foram escolhidas para estudo, duas de capacidade equivalente e uma de alocação dinâmica. Os algoritmos propostos por Guérin, Ahmadi e Naghshineh [1], Kesidis, Walrand e Chang [2] e Saito e Shiomoto [4] são detalhados no Cap. 3.

## 2.5 Conclusão

O objetivo deste capítulo foi o de apresentar as informações básicas para o estudo de controle de admissão de conexões em redes ATM. Os assuntos abordados são importantes para a compreensão das simulações discutidas no Cap. 4.

Os tipos de categorias de serviço especificados pelo ATM Forum foram detalhados. Os parâmetros descritores de tráfego e de qualidade de serviço utilizados em cada categoria foram ressaltados. Além disso, é indicado em cada categoria qual é o tipo de tráfego mais adequado para sua aplicação.

Foram discutidos os modelos matemáticos de fontes de tráfego utilizados na solução de problemas de enfileiramento e transbordamento de células em comutadores ATM.

Por último, várias propostas apresentadas na literatura para solucionar o problema de CAC foram discutidas brevemente. Os três algoritmos escolhidos para análise são detalhados no Cap. 3.

# CAPÍTULO 3

# ALGORITMOS DE CAC

## 3.1 Introdução

Os principais esquemas propostos na literatura para a implementação de CAC (Connection Admission Control) em redes ATM foram discutidos no Cap. 2. O ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) e o ATM Forum não definiram um método a ser utilizado e deixaram livre a escolha, pois cada configuração de comutador pode usufruir melhor de um ou outro esquema.

Neste capítulo, os três algoritmos escolhidos para comparação são apresentados detalhadamente. São discutidos dois algoritmos baseados em capacidade equivalente, ou banda efetiva, e um algoritmo baseado em alocação dinâmica.

## 3.2 Algoritmo 1: Modelo de Fontes Combinadas

Este algoritmo [1] utiliza o conceito de capacidade equivalente. A capacidade equivalente, ou a banda efetiva consiste na largura de banda necessária para atender o tráfego de uma conexão mantendo as qualidades de serviço da nova conexão e daquelas em andamento dentro de valores especificados.

O cálculo do valor necessário de banda é baseado unicamente nas informações dos parâmetros descritores de tráfego da conexão solicitada e nos recursos da rede, como tamanho do *buffer*. A grande vantagem do conceito de banda efetiva é que o cálculo total da banda em uso é a adição das capacidades equivalentes de todas as conexões ativas. Assim, a rede operará de maneira similar ao modo de comutação de circuitos. Isto permite

uma decisão rápida, que é fundamental em redes que trabalham com altas taxas de comutação e transporte de informações.

O conceito de capacidade equivalente é baseado na análise da taxa de decaimento assimptótica da distribuição estacionária da ocupação de um *buffer*. Essa taxa de decaimento dependerá do tipo de fonte que é submetido ao *buffer*.

O cálculo da capacidade equivalente apresentado em [1] utiliza a fonte do tipo fluido Markoviano de dois estados, ou *on-off*. Este tipo de fonte pode ser caracterizada pela sua taxa de pico $R$, a fração do tempo em que está ativa $r$ (carga) e o comprimento médio dos surtos gerados $b$.

O modelo de nó de comutação utilizado em [1] é baseado inicialmente em uma única fonte *on-off* e um simples *buffer* finito, mostrado na Fig. 3.1.

Seja $P_i(t, x)$ a probabilidade de que a fonte esteja no estado $i$ e o conteúdo do *buffer* seja $x$ no instante $t$.

Se i = 1 $\Rightarrow$ ativo

Se i = 0 $\Rightarrow$ silêncio

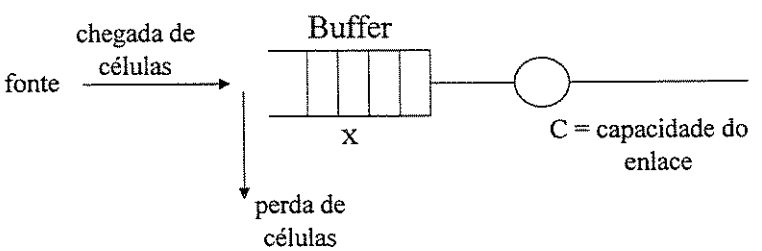


Figura 3.1: *Multiplexador ATM*

Utilizando o modelo de aproximação por fluxo de fluido, para análise de ocupação do *buffer*, as probabilidades $P_i(t,x)$ podem ser equacionadas e resolvidas [6].

Supondo o caso de uma única fonte conectada e invertendo a expressão da distribuição da ocupação de um buffer finito de tamanho $K$ e submetido ao tráfego de uma única fonte, a probabilidade de perda de células $\varepsilon$ é dada por,

$$\varepsilon = \beta \cdot \exp\left( -\frac{K(c-rR)}{b(1-r)(R-c)c} \right), \tag{3.1}$$

onde

$$\beta = \frac{(c - rR) + \varepsilon r(R - c)}{(1 - r)c}, \tag{3.2}$$

e $c$ é a capacidade equivalente da fonte em questão. O parâmetro $\varepsilon$ é também considerado a taxa de perda de células, CLR.

Mesmo no caso simples de uma única fonte, não há um resultado explícito para a capacidade equivalente e a equação precisa ser resolvida numericamente. A resolução numérica pode levar a tempos de resposta inadequados ao funcionamento da rede. É necessário utilizar alguma simplificação e $\beta$ é assumido ser igual a 1. Esta aproximação resulta num limitante superior para $c$.

Fazendo $\alpha = \ln(1/\varepsilon)$, temos

$$\alpha = \frac{K(c - rR)}{b(1 - r)(R - c)c}, \tag{3.3}$$

que resulta na equação de segundo grau

$$\alpha b(1 - r)c^2 - [\alpha b(1 - r)R - K]c - rRK = 0. \tag{3.4}$$

O resultado final para a capacidade é

$$c = \frac{a - K + \sqrt{(a - K)^2 + 4Kar}}{2a} R, \tag{3.5}$$

onde $a = \alpha b(1-r)R$.

Para o caso de $N$ fontes, a banda efetiva é calculada por

$$C_E = \sum_{i=1}^{N} \hat{c}_i \tag{3.6}$$

onde $\hat{c}_i$ é a banda efetiva da fonte $i$.

Tem sido observado que a taxa estacionária de *bits* do tráfego agregado de várias fontes se aproxima de uma distribuição normal com média $m$ e variância $\sigma^2$ [12]. Como na Eq. 3.6 as fontes não foram consideradas agregadas, a banda $C_E$ pode estar superestimada. Assim, é considerada também uma fonte agregada gaussiana. O valor da capacidade

equivalente para a fonte agregada pode ser obtido por aproximações para a inversa da distribuição normal de probabilidades.

Para a fonte gaussiana, a banda efetiva é dada por [1]

$$C_G = m + \alpha'\sigma, \tag{3.7}$$

onde

$\alpha' = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)}$,

$m = \sum_{i=1}^{N} m_i$ , é a taxa média total de *bits*, e

$\sigma^2 = \sum_{i=1}^{N} \sigma_i^2$ ; $\sigma_i^2$ é a variância da taxa de bits da i-ésima fonte.

A banda efetiva final será

$$C_F = \min\{C_E, C_G\}. \tag{3.8}$$

Se $C_F \leq C_T$, onde $C_T$ é a capacidade total do enlace, aceita-se a conexão; caso contrário, a chamada é rejeitada.

A aproximação gaussiana para a taxa estacionária de bits do tráfego agregado não é adequada em um caso bem específico e pode resultar num valor de capacidade equivalente subestimado. Isso ocorre para fontes com taxas de pico elevadas, longos períodos de surto e baixa utilização e que resultam num número pequeno de fontes atendidas pelo enlace. Nestes casos, deve-se desconsiderar o valor de capacidade obtido pela aproximação gaussiana, $C_G$, e utilizar o valor retornado pela aproximação de fluxo de fluido, $C_E$.

## 3.3 Algoritmo 2: Modelo de Fontes Gerais

Kesidis, Walrand e Chang [2] mostraram a existência de banda efetiva para uma classe mais geral de fontes. O modelo utilizado é baseado na suposição de que a probabilidade de ocupação de um *buffer* infinito, em estado estacionário, ultrapassar um limite $K$ obedece a um decaimento exponencial:

$$\mathrm{P}\{X > K\} \leq e^{-K\delta},$$

onde $\delta$ é uma constante positiva chamada taxa de decaimento assimptótica.

Além disso, o modelo é baseado na existência da função

$$h(\delta) = \lim_{t \to \infty} \frac{\ln \mathrm{E}\{\exp(A(t)\delta)\}}{t} \tag{3.9}$$

onde $A(t)$ é o número de células geradas no intervalo de tempo [0, t] e $\delta$ é um valor real.

A banda efetiva é dada por [2]

$$c(\delta) = \frac{h(\delta)}{\delta}. \tag{3.10}$$

O cálculo de $h(\delta)$ é, em geral, bastante complexo, e dificulta a sua utilização em CAC. Entretanto, resultados são apresentados em [2] para fontes de taxa constante, Poisson, fontes de Markov discretas no tempo, fluidos de Markov e processos poissonianos modulados por Markov. Para a fonte do tipo fluido markoviano *on-off*, as expressões acima podem ser calculadas e a banda efetiva é dada por [2]

$$c = \alpha + \sqrt{\alpha^2 + \beta^2} \tag{3.11}$$

onde

$$\alpha = \frac{1}{2}\left( R - \frac{1}{\delta b} - \frac{1}{\delta T_{off}} \right), \text{ e} \tag{3.12}$$

$$\beta = \frac{R}{\delta T_{off}}. \tag{3.13}$$

$T_{off}$ é o tempo médio dos períodos de silêncio da fonte,

$$T_{off} = \frac{b(1-r)}{r}. \tag{3.14}$$

O valor de $\delta$ é dado por

$$\delta = \frac{\ln\left(1/\varepsilon\right)}{K}, \tag{3.15}$$

onde $\varepsilon$ é a taxa de perda de células (CLR) especificada no contrato de tráfego.

## 3.4 Algoritmo 3: Alocação Dinâmica

A variedade de tipos de fontes de tráfego que uma rede ATM deve atender é bastante grande. Cada tipo de tráfego tem características e exigências muito diversas. Voz e vídeo interativos precisam de garantias rigorosas quanto ao tempo de atraso e variação desse atraso. Novos codificadores foram desenvolvidos e o que antes era transmitido em taxa constante, hoje é transmitido em taxa variável e em larguras de banda menores. Não há ainda um consenso de como representar de maneira simples e eficiente esses codificadores. Dados, muitas vezes, podem aceitar atrasos elevados, mas são intolerantes à perda de informações.

Modelos mais complexos de fontes permitem representar melhor a variedade de comportamento do tráfego gerado, mas aumentam muito a dificuldade do modelo matemático a ser solucionado e o resultado obtido, com freqüência, é inviável de ser implementado em um comutador ATM.

O tráfego contratado pelo usuário pode não se concretizar. Isso pode acontecer por uma dificuldade em prever o tráfego a ser gerado além da dificuldade em representá-lo utilizando um conjunto restrito de parâmetros descritores como os especificados pelo ITU-T (PCR, SCR, CDVT e IBT) ou ATM Forum (PCR, SCR, CDVT, MBS e MCR).

Outro problema relacionado à descrição do tráfego surge da modificação por ele sofrido ao longo dos nós da rede. Ao passar por um comutador, as células de uma conexão passam por um processo de enfileiramento em memória. Como as células de uma conexão ocupam a memória de maneira intercalada com células de outras conexões, o tráfego de cada conexão na saída do enlace é diferente do apresentado na entrada do comutador. Ao longo da rede, os parâmetros que descrevem o tráfego da conexão vão, progressivamente, perdendo sua representatividade. Como todos os nós da rede devem executar a rotina de CAC utilizando o mesmo conjunto de descritores de tráfego, a decisão de aceitação ou recusa torna-se mais difícil quando a conexão percorre um número maior de nós da rede. Uma alternativa é efetuar a conformação do tráfego (*traffic shaping*) em cada nó para tentar mantê-lo dentro das especificações contratadas.

Tentando contornar esses problemas, Saito e Shiomoto [4] propuseram um esquema de alocação dinâmica. Além dos parâmetros fornecidos pelo usuário, a rede baseia-se no tráfego efetivamente medido para decidir se pode ou não atender a uma nova conexão. A proposta tem a vantagem de não ser baseada em nenhum modelo específico de fonte de tráfego. O método é baseado na estimação da distribuição de probabilidade das fontes agregadas, através das contagens de células que chegam num intervalo fixo *s*.

O modelo consiste de $n$ conexões sendo atendidas por um enlace de capacidade $C'$ *bits/s*, um buffer de tamanho $K$, e $p_i^*(j)$ é a distribuição de probabilidade da chegada de $j$ células da conexão $i$ durante o intervalo de medição $s$. Assumindo $C = C's/L \leq K+1$, onde $L$ é o comprimento de uma célula ATM em *bits*, o limite superior para a probabilidade de perda de células, $\varepsilon$, é dado por

$$\varepsilon \leq \frac{\sum_{k=0}^{\infty}[k-C]^+ p_1^* * \cdots * p_n^*(k)}{\sum_{k=0}^{\infty} k p_1^* * \cdots * p_n^*(k)} \tag{3.16}$$

onde

$$[x]^+ = \begin{cases} x, & \text{se } x > 0 \\ 0, & \text{de outra forma} \end{cases}$$

e $*$ é a convolução.

Um usuário, ao solicitar conexão, deve fornecer a taxa de pico de bits $\xi$ e a taxa média de bits $\psi$. Para incluir a chamada solicitada no cálculo de $\varepsilon$ para a tomada de decisão é definido o vetor $\theta$ que representa o pior caso da distribuição de chegada de células da conexão solicitada em relação aos períodos de surto e silêncio [4]. Dessa forma,

$$\theta_{n+1}(j) = \begin{cases} a_{n+1}/R_{n+1}, & j = R_{n+1} \\ 1-(a_{n+1}/R_{n+1}), & j = 0 \\ 0, & \text{de outra forma} \end{cases} \tag{3.17}$$

onde $R_{n+1} = \lceil \xi_{n+1} s/L \rceil$ e $a_{n+1} = \psi_{n+1} s/L$. $\lceil x \rceil$ representa o menor inteiro maior ou igual a x. $R_{n+1}$ e $a_{n+1}$ representam o número máximo e o número médio de células que chegam,

durante o intervalo de medição $s$, da chamada n+1. Acrescentando a informação da conexão requisitada temos

$$\varepsilon \leq \frac{\sum_{k=0}^{\infty}[k-C]^{+} p_1^{*} * \cdots * p_n^{*} * \theta_{n+1}(k)}{\sum_{k=0}^{\infty} k p_1^{*} * \cdots * p_n^{*}(k) \quad + \quad a_{n+1}}. \tag{3.18}$$

Através de medição é possível obter uma estimativa para $p_1^{*} * \cdots * p_n^{*}(k)$. Usando a estimativa $\hat{\mathbf{p}} = (\hat{p}(0), \hat{p}(1), \cdots)$ pode-se obter o valor estimado da probabilidade de perda de células com a aceitação da nova conexão, $\hat{\varepsilon}_1$, que é

$$\hat{\varepsilon}_1 \leq \frac{\sum_{k=0}^{\infty}[k-C]^{+} \hat{p} * \theta_{n+1}(k)}{\sum_{k=0}^{\infty} k \hat{p}(k) \quad + \quad a_{n+1}} \tag{3.19}$$

Se os erros na estimativa $\hat{\mathbf{p}}$ forem suficientemente pequenos, o valor $\hat{\varepsilon}_1$ pode ser usado como o limite superior da probabilidade de perda. Se o valor encontrado for inferior ao CLR desejado, a conexão é aceita, caso contrário, recusada.

O método precisa usar um processo de contagem do número de células que chegam no intervalo de medição $s$ para estimar a distribuição de probabilidade do número de células que chegam nesse tempo. Um conjunto de $N$ intervalos de medição é chamado de período de renovação. A distribuição de freqüência medida nos $N$ intervalos do $t$-ésimo período de renovação é identificada por $\{q(k; t), k = 0, 1, \cdots\}$ e o valor médio do número de chegada de células é

$$b(t) = \sum_{k=0}^{\infty} k q(k;t) \tag{3.20}$$

Sempre que um período de renovação é completado as estimativas $\hat{\mathbf{p}}$ e $\hat{a}$ devem ser atualizadas. A forma de atualização depende da ocorrência de aceitação de novas conexões ou encerramento de conexões que já estavam ativas.

Se o número de conexões ativas durante um período de renovação não é alterado, as estimativas são atualizadas da seguinte forma:

$$\hat{p}(t+1) = \alpha q(t) + (1-\alpha)\hat{p}(t) \tag{3.21}$$

$$\hat{a}(t+1) = \alpha b(t) + (1-\alpha)\hat{a}(t) \tag{3.22}$$

onde $\alpha$ é um valor entre 0 e 1, que tem a função de regular o peso entre o tráfego medido no último período de medição e o resultado acumulado durante os períodos anteriores.

Quando uma nova conexão é estabelecida no $t$-ésimo período de renovação, deve-se fazer uso dos descritores de tráfego fornecidos pelo usuário que solicita a conexão e, por isso, é incluído no cálculo o vetor $\theta$.

$$\hat{p}(k; t+1) = \hat{p}(\cdot; t) * \theta_{n+1}(k), \quad (k = 0, 1, \cdots) \tag{3.23}$$

$$\hat{a}(t+1) = \hat{a}(t) + a_{n+1} \tag{3.24}$$

Quando uma conexão é encerrada é necessário subtrair seu tráfego dos valores estimados e a atualização dos parâmetros fica da seguinte forma:

$$\hat{p}(0; t+1) = \sum_{k=0}^{a'} \hat{p}(k; t) \tag{3.25}$$

$$\hat{p}(k; t+1) = \hat{p}(k + a'; t) \quad (k \geq 1) \tag{3.26}$$

$$\hat{a}(t+1) = \hat{a}(t) - a' - \sum_{k=0}^{a'} (k - a')\hat{p}(k; t) \tag{3.27}$$

onde $a' = \left\lceil \hat{a}(t) - \sum_{i=1}^{n} a_i \right\rceil$.

O vetor $\theta$ é calculado utilizando a taxa média e a taxa de pico da fonte que solicita a conexão. Esses valores de taxa média e de pico são referentes ao tempo do intervalo de medição $s$, ou seja, são calculados através do produto da taxa (em células/segundo) pelo intervalo $s$ (em segundos). O parâmetro $s$ é tomado a partir de um número inteiro de *slots* de tempo do enlace. Entenda-se por *slot* o tempo necessário para a transmissão de uma célula ATM pelo enlace. A largura de $s$ (em *slots*) é dividida pela velocidade do enlace (em células/segundo), para se obter a largura do intervalo de medição (em segundos).

As fontes que solicitam conexão, na grande maioria dos casos, têm taxas bem inferiores à do enlace. Tomando como exemplo o intervalo de medição de 80 *slots*, taxa de pico da fonte de 8 *Mbps* (18.867,92 células/segundo) e taxa do enlace de 155.52 *Mbps*

(366.792,45 células/segundo), a taxa de pico de cada fonte é de 4,11 células por intervalo *s*. Este valor é muito pequeno se comparado à largura de 80 *slots* do intervalo de medição. Como a taxa de pico é calculada através de $R_{n+1} = \lceil \xi_{n+1} s / L \rceil$, o menor valor assumido para $R_{n+1}$ será 1. No exemplo acima, $R_{n+1}$ assumiria valor 5.

Se, no exemplo acima, a taxa de pico da fonte não for tão alta (1 *Mbps*, por exemplo), a taxa de pico de cada fonte por intervalo *s* será oito vezes menor, ou seja, 0,51. Para fontes de 8 *Mbps*, se o intervalo de medição for de 19 *slots*, a taxa em *s* será 4,21 vezes menor, ou seja, será menor que 1 (80/19 = 4,21). Se tomarmos fontes com taxas mais baixas, 64 *Kbps*, por exemplo, o valor de $R_{n+1}$ estará muito sobrestimado. O valor encontrado para $R_{n+1}$ é de 0,033 célula por intervalo *s*, mas como é preciso usar o menor inteiro maior ou igual ao valor encontrado, faz-se $R_{n+1} = 1$, ou seja, 30 vezes maior.

Para obter uma estimativa suficientemente precisa para a distribuição de chegada de células o período de renovação precisa ser longo. Se ocorrer um número elevado de solicitações de conexões dentro de um único período de renovação, pode haver decisão errônea por parte do algoritmo devido ao acúmulo de capacidades sobrestimadas como discutido acima.

## 3.5 Conclusão

Os algoritmos escolhidos para a análise foram detalhados neste capítulo. Aspectos importantes de seus comportamentos, que podem provocar bom ou mau desempenho, foram ressaltados a partir da análise teórica. Os dois esquemas de capacidade equivalente utilizam o mesmo conjunto de descritores de tráfego (taxa de pico, carga e tamanho máximo do surto), enquanto o esquema de alocação dinâmica usa apenas dois (taxa de pico e taxa média). Os parâmetros utilizados pelas três propostas facilitam a comparação dos algoritmos a ser feita no Cap. 4.

# CAPÍTULO 4

# SIMULAÇÃO E RESULTADOS

## 4.1 Introdução

Neste capítulo, três algoritmos detalhados no Cap. 3 são analisados através de simulação de eventos discretos. É utilizado o modelo de um multiplexador ATM submetido ao tráfego de $N$ fontes idênticas do tipo *on-off*.

É discutida, inicialmente, a estrutura geral do modelo de simulação. A seguir, são apresentados os principais resultados da simulação e, por fim, é feita uma análise de confiabilidade dos resultados.

## 4.2 Modelo de Simulação

O modelo de nó utilizado é um multiplexador ATM submetido ao tráfego de um conjunto de fontes idênticas. O multiplexador é modelado por um *buffer* de tamanho $K$, com disciplina de atendimento do tipo FIFO (*First In-First Out*) e um único servidor com taxa de atendimento constante igual à capacidade do enlace. As fontes são do tipo fluido markoviano de dois estados (*on-off*) e são caracterizadas pela taxa de pico $R$, comprimento médio dos surtos gerados $b$ e carga $r$.

Para obter resultados mais próximos do que é esperado encontrar em sistemas reais, utilizou-se um enlace com taxa de transmissão de 155.52 *Mbps*. A taxa de perda de células, $\varepsilon$, foi fixada em $10^{-3}$. O tempo de tráfego simulado foi de 900 segundos, o que representa um fluxo de células significativo num multiplexador ATM operando à taxa escolhida. Há um período inicial da simulação (*warm-up*) para permitir que se atinja o regime estacionário, só após o qual o tempo de 900 segundos é contado e as estatísticas passam a ser obtidas.

## 4.3 Resultados

### 4.3.1 Variação do Tamanho do Buffer

O primeiro conjunto de simulações mostra o efeito da variação do comprimento do *buffer* na resposta dos algoritmos analisados. Utiliza fontes com taxa de pico de 8 *Mbps*, comprimento médio de surto de 100 células e carga de 0,2. Como as características do tráfego gerado por uma fonte multimídia ainda não são bem conhecidas, foram escolhidas, aleatoriamente, cargas que possibilitam maior ganho estatístico pelo multiplexador e, por isso, são as de maior interesse na determinação eficiente de sua banda efetiva.

Pelas curvas da Fig. 4.1 pode-se observar que o número de conexões aceitas pelos três algoritmos difere fundamentalmente na região em que o *buffer* é pequeno ($K < 1.000$ células). Os dois esquemas de capacidade equivalente apresentam o mesmo resultado para *buffers* de 700 ou mais células. O algoritmo de Guérin, Ahmadi e Naghshineh [1] tem melhor aproveitamento do enlace. Além disso, pode-se verificar que aumentos no tamanho do *buffer* além de 5.000 células têm efeito muito pequeno. Contudo, é importante ressaltar que este limite varia de acordo com o comprimento do surto; para valores maiores de surto a saturação acontecerá com valores maiores do *buffer*.

As propostas de banda efetiva só se aproximam do limite superior de aceitação para valores grandes de *buffer*, quando já se verifica a saturação. Nesta situação, o *buffer* é suficientemente grande para suportar a chegada simultânea de surtos da maioria das fontes conectadas e a capacidade alocada para cada uma delas está muito próxima de sua taxa média.

O método de alocação dinâmica não tira proveito do tamanho do *buffer* e apresenta bons resultados apenas para valores inferiores a 1.000 células. Este método pode considerar o *buffer* através da vinculação da largura do intervalo de medição ao tamanho deste. Contudo, isso é factível apenas para valores bem pequenos (algo abaixo de 500 células), caso contrário, o processo de estimação de tráfego ficaria demasiadamente lento.

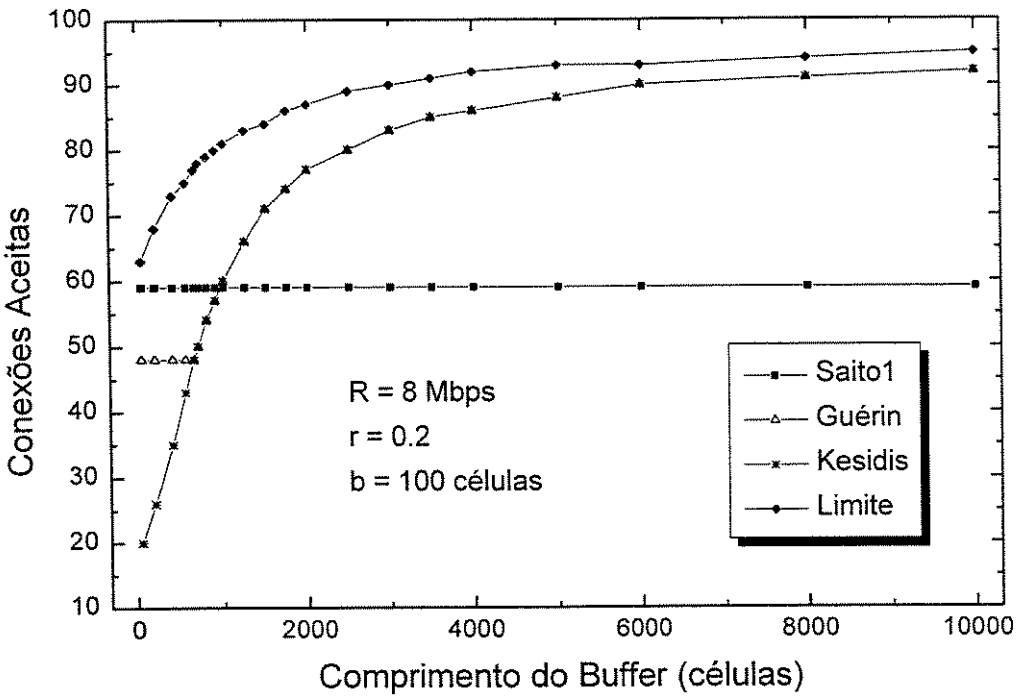


Figura 4.1: *Número de conexões aceitas em função do tamanho do buffer.*

A utilização obtida no enlace é expressiva apenas para valores grandes do *buffer* (Fig. 4.2). Tamanhos de *buffer* até 1.000 células não possibilitam utilização elevada do enlace.

Novamente, percebe-se a vantagem da aproximação gaussiana implementada no algoritmo de Guérin et al., pois o número de conexões aceitas é igual na região de capacidade equivalente e maior na região onde a aproximação gaussiana prevalece.

A capacidade de 2.000 células já é suficiente para atingir uma utilização do enlace de, aproximadamente, 0,8. A proposta de Saito et al. é superior as outras apenas quando o *buffer* é pequeno.

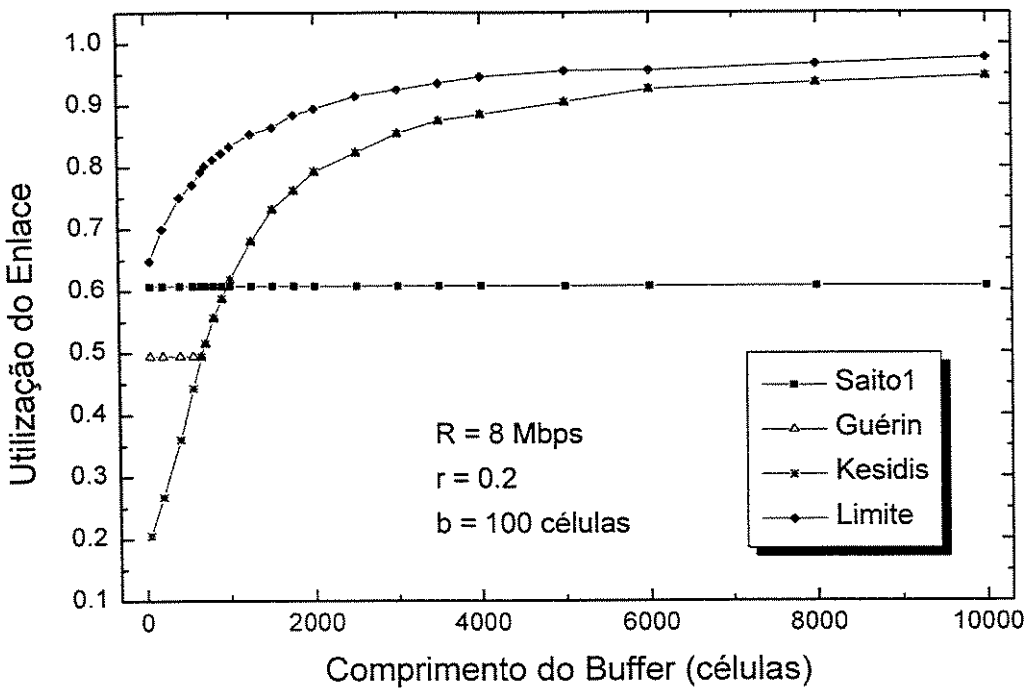


Figura 4.2: *Utilização do enlace em função do tamanho do buffer.*

O tempo médio de espera das células no *buffer* é um parâmetro importante, pois aplicações de tempo real (telefonia, por exemplo) não admitem atrasos elevados. O aumento no tamanho do *buffer* permite atingir uma utilização bastante alta do enlace, mas é preciso haver um compromisso que considere o atraso das células.

Os atrasos obtidos (Fig. 4.3) são muito pequenos apenas quando o *buffer* também é pequeno ($K < 2.000$ células). Valores maiores do *buffer* permitem alta utilização do enlace, contudo provocam tempos de espera muito elevados.

Verifica-se que utilizações da ordem de 0,6, como foi obtido pela alocação dinâmica, resultam em tempos de espera praticamente nulos, o que é desejável nas conexões de tempo real. Para utilizações superiores a 0,6, o tempo de atraso aumenta rapidamente e pode chegar a valores muito elevados, não suportados por tráfegos CBR e rt-VBR.

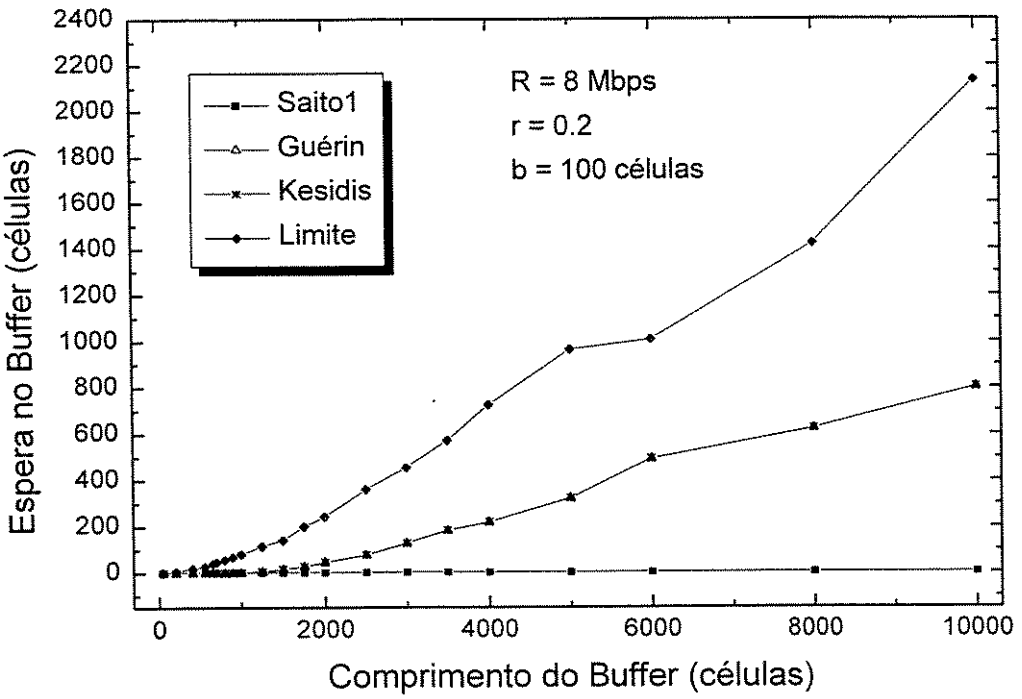


Figura 4.3: *Tempo médio de espera em função do tamanho do buffer.*

Na Fig. 4.4 pode-se constatar que todos os algoritmos apresentaram taxa de perda quase nula, com exceção dos dois primeiros pontos do método de alocação dinâmica (buffers de 50 e 200 células).

Na maior parte da Fig. 4.1, verifica-se que a diferença no número de conexões aceitas pelos algoritmos de capacidade equivalente e o limite estimado de aceitação é inferior a dez. No entanto, a perda foi quase nula nos primeiros enquanto que chegou bem próximo de $10^{-3}$ na curva do limite estimado. Isso demonstra que a taxa de perda de células pode crescer muito rapidamente com o aumento do número de conexões.

Choudhury, Lucantoni e Whitt [18] afirmam que o método de capacidade equivalente pode sobrestimar a probabilidade de perda de células por várias ordem de magnitude e que, em algumas situações, seria possível aceitar até duas vezes mais que o previsto. Neste grupo de simulações, verifica-se que a probabilidade de perda de células é sobrestimada e o limite de aceitações chegou a ser o triplo do valor encontrado por capacidade equivalente.

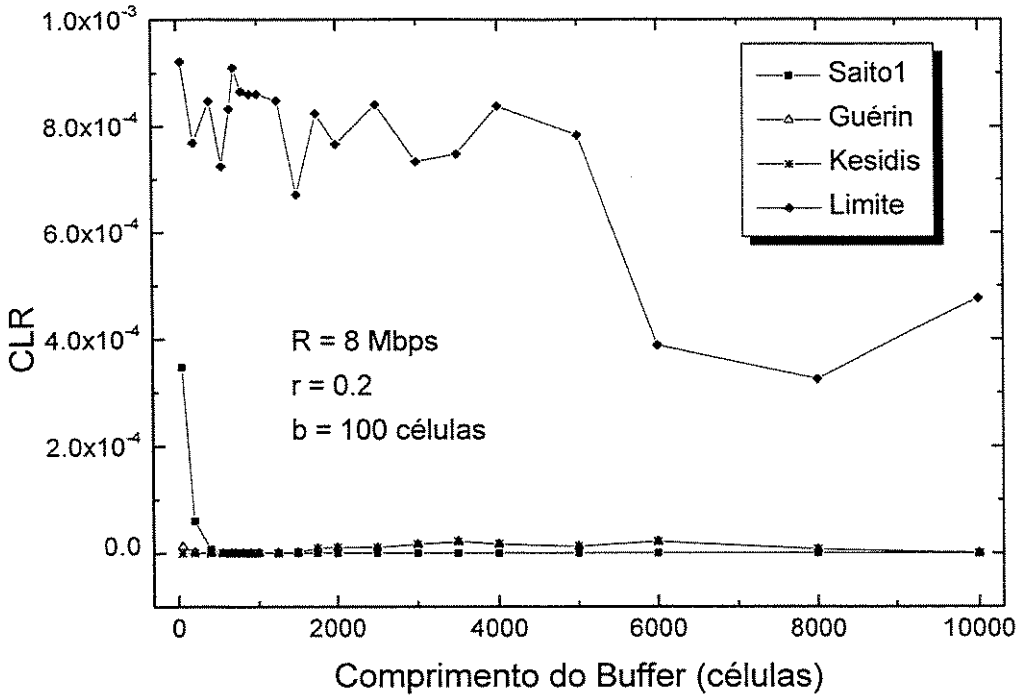


Figura 4.4: *Taxa de perda de células em função do tamanho do buffer.*

### 4.3.2 Variação do Comprimento do Surto

O segundo conjunto de simulações verifica o comportamento do algoritmo frente à variação do comprimento médio do surto. Utiliza fontes com taxa de pico de 8 *Mbps*, capacidade do *buffer* de 4.000 células e carga de 0,3. Verifica-se que, novamente, o algoritmo de Guérin et al. [1] tem melhores resultados (Fig. 4.5). Aqui pode-se deduzir que a diferença entre os algoritmos acontece quando o *buffer* é pequeno em relação ao tamanho médio do surto.

O esquema de alocação dinâmica mostra-se pouco sensível à variação do tamanho médio do surto. Apresenta resultado inferior aos outros métodos para surtos de até 1.000 células, mas é bem superior para valores maiores. Contudo, apresenta uma ligeira subestimação do tráfego quando o surto é muito grande (acima de 5.000 células) e aceita conexões além do limite de atendimento.

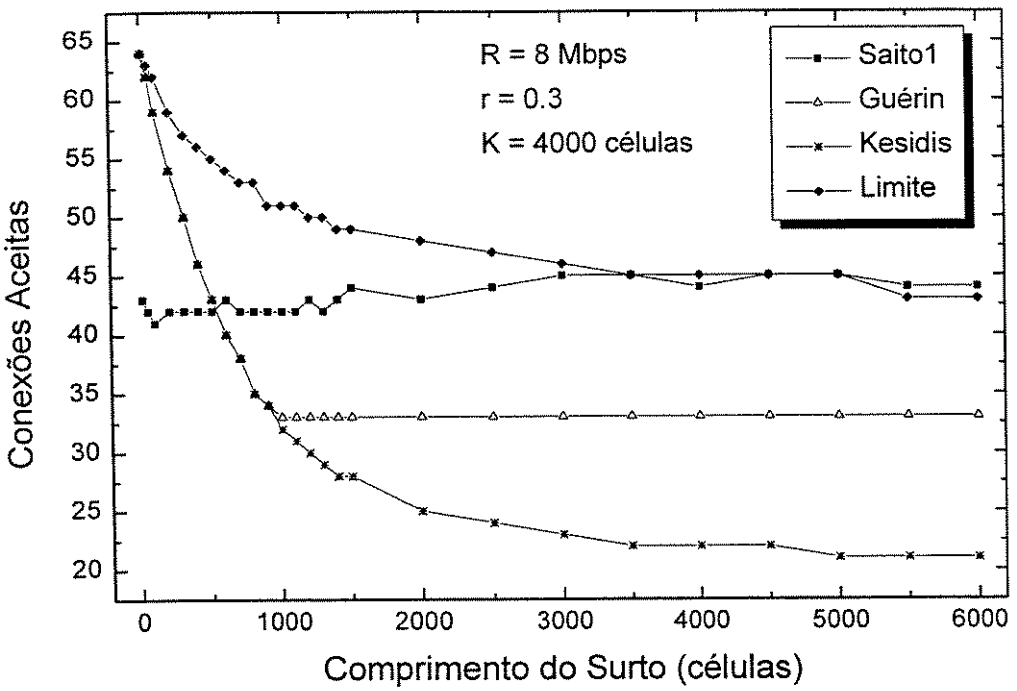


Figura 4.5: *Número de conexões aceitas em função do comprimento médio do surto.*

A utilização cai muito rapidamente com o crescimento do tamanho do surto (Fig. 4.6), mas pode ser bastante alta para surtos menores que 500 células. Obviamente, o tamanho do *buffer* influi fortemente nesta queda abrupta; valores maiores permitiriam um decaimento mais suave.

Ainda assim, o método de alocação dinâmica consegue uma utilização do enlace de 0,7, enquanto o de Guérin et al. fica em 0,5. O limite estimado apresenta uma utilização de aproximadamente 0,7. Isso mostra que o desempenho da alocação dinâmica foi bom, apesar de preocupar por estar excessivamente próximo e até ultrapassar o limite em algumas situações.

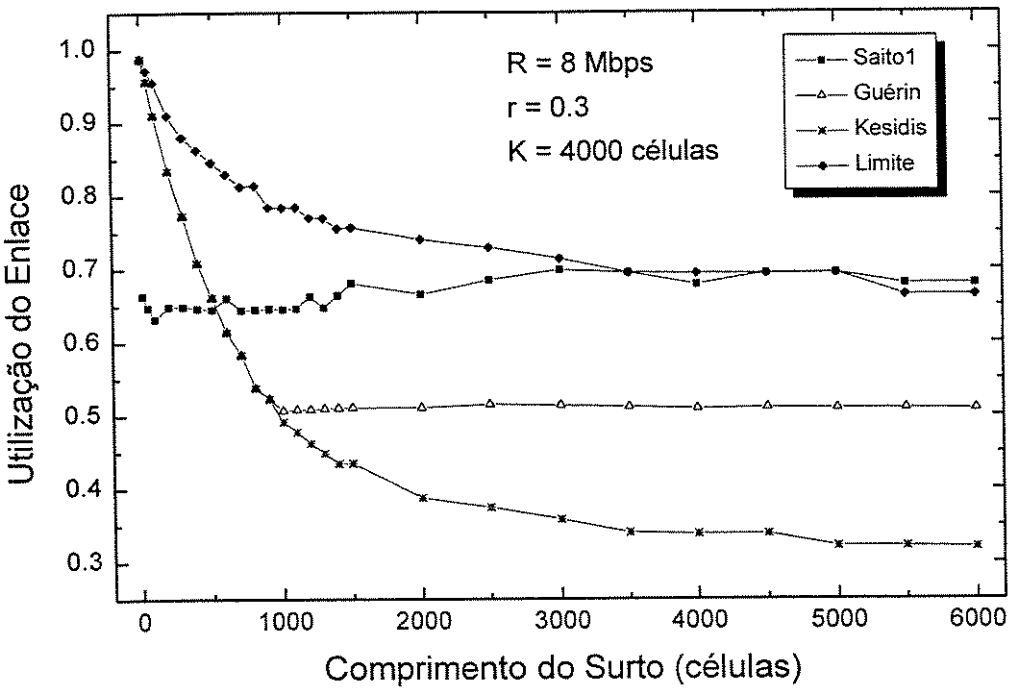


Figura 4.6: *Utilização do enlace em função do comprimento médio do surto.*

O tempo de espera é elevado nos pontos em que o enlace tem alta utilização (Fig. 4.7). Se o tempo for um parâmetro crítico, é necessário optar por operar o multiplexador em níveis de utilização menores.

Na maior parte da região estudada, os tempos de atraso são muito pequenos, atingindo o limite em torno de 100 células, que é um atraso baixo.

Na Fig. 4.8, verifica-se que os algoritmos de capacidade efetiva são muito conservadores. A alocação dinâmica aceita mais conexões que, por sua vez, resultam numa taxa de perda de células maior.

Como antecipado pelo número de conexões, o limite de aceitação é ultrapassado para surtos em torno de 5.000 células. Conseqüentemente, a taxa de perda de células resultante é superior ao padrão especificado, violando o contrato de tráfego.

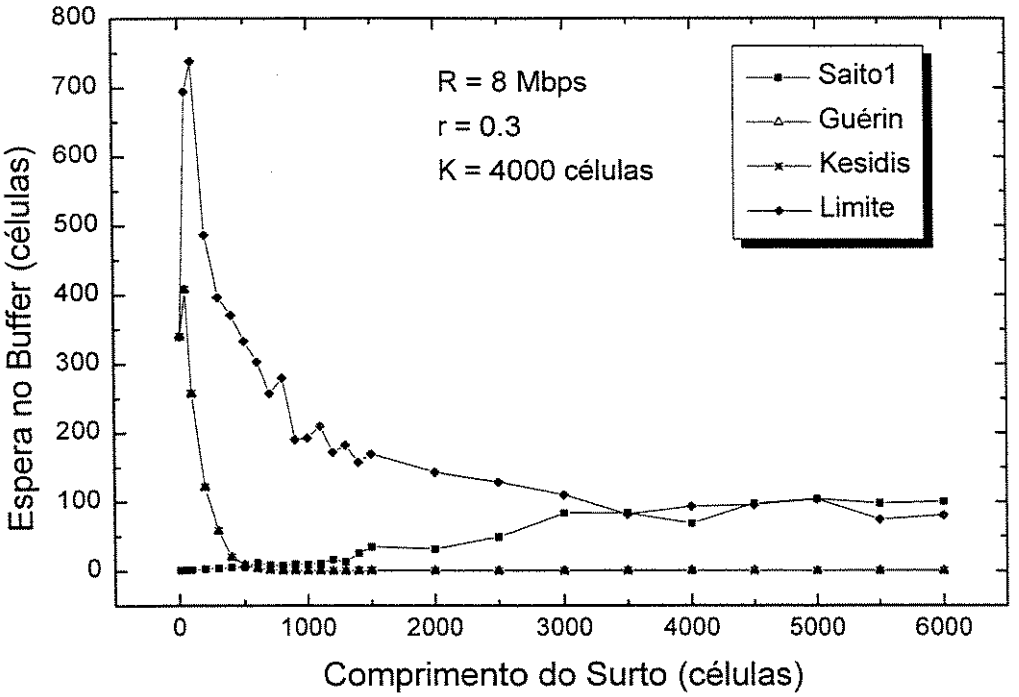


Figura 4.7: *Tempo médio de espera no buffer em função do comprimento médio do surto.*

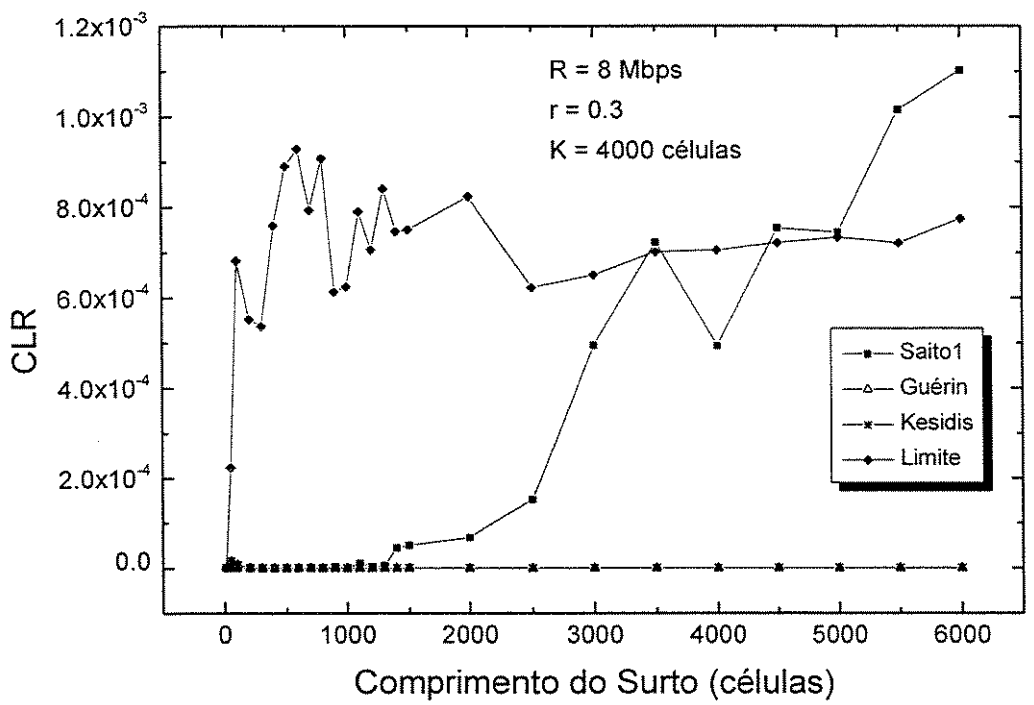


Figura 4.8: *Taxa de perda de células em função do comprimento médio do surto.*

### 4.3.3 Variação da Carga das Fontes

O terceiro conjunto de simulações mostra como a variação da carga, $r$, das fontes afeta os resultados dos esquemas estudados. Utiliza fontes com taxa de pico de 8 *Mbps*, capacidade do *buffer* de 1.000 células e comprimento médio dos surtos de 200 células. As curvas, novamente, confirmam que os modelos são bastante conservadores.

Em especial, a proposta de Kesidis, Walrand e Chang [2] é excessivamente conservadora quando a carga das fontes é pequena (Fig. 4.9). Comprovamos, através de simulações adicionais, que esta diferença diminui na medida em que a razão *K/b* é maior. Em outras palavras, o desempenho deste algoritmo se aproxima do outros a medida que o tamanho do *buffer* é maior em relação ao tamanho médio dos surtos.

A proposta de Saito et al. tem melhores resultados para cargas até 0,4, mas é superado pelos outros esquemas nas cargas mais altas.

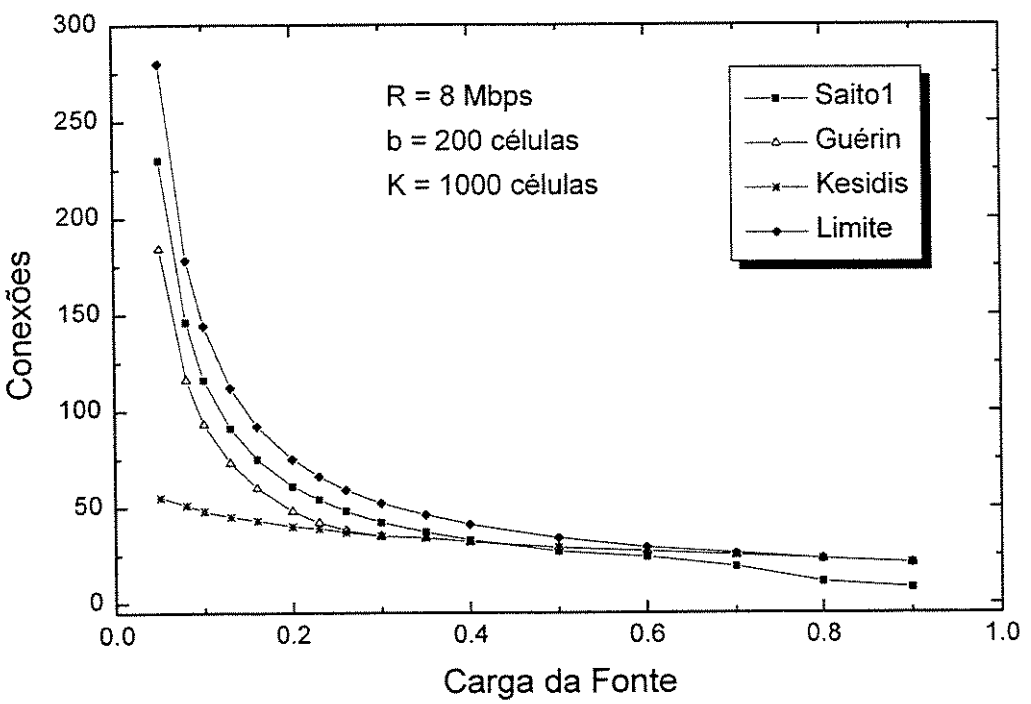


Figura 4.9: *Número de conexões aceitas em função da carga das fontes.*

Para obter uma boa utilização do enlace é necessário um *buffer* suficientemente grande para armazenar as eventuais seqüências longas de surtos agregados. Pode-se ver

que a utilização foi baixa na região em que a carga é pequena (Fig. 4.10). Isso significa que o *buffer* utilizado não foi bastante grande e havia espaço para aceitação de mais conexões com um *buffer* maior. Contudo, verificamos que isso limitou os atrasos de células em valores pequenos (Fig. 4.11).

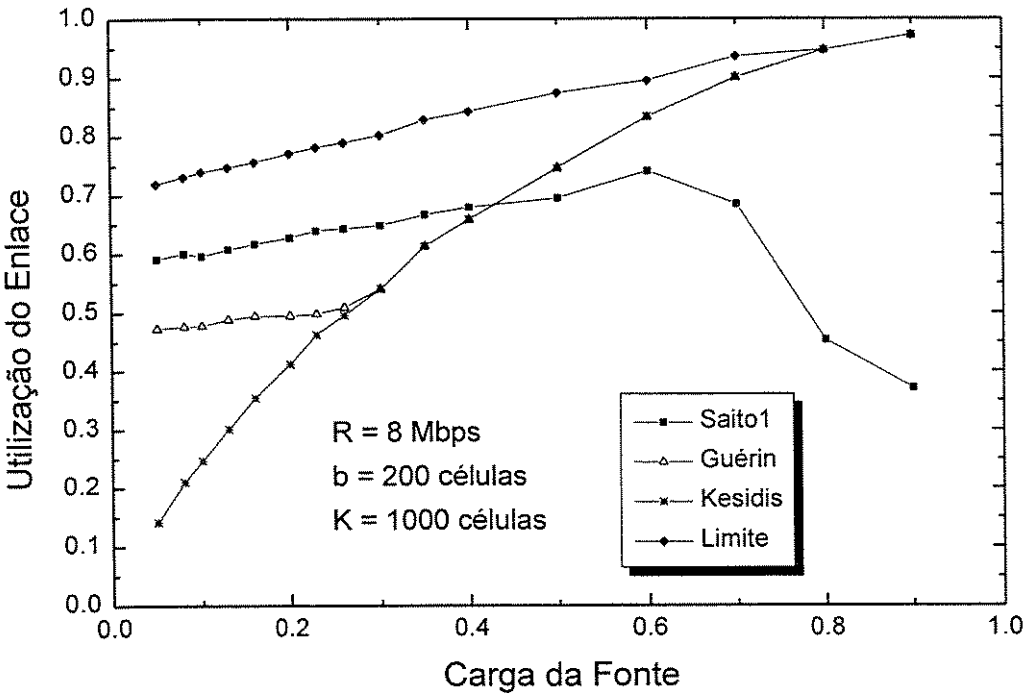


Figura 4.10: *Utilização do enlace em função da carga das fontes.*

A taxa de perda de células praticamente nula confirma o caráter conservador dos algoritmos (Fig. 4.12). A folga para aceitação de novas conexões é bastante grande nas cargas mais baixas.

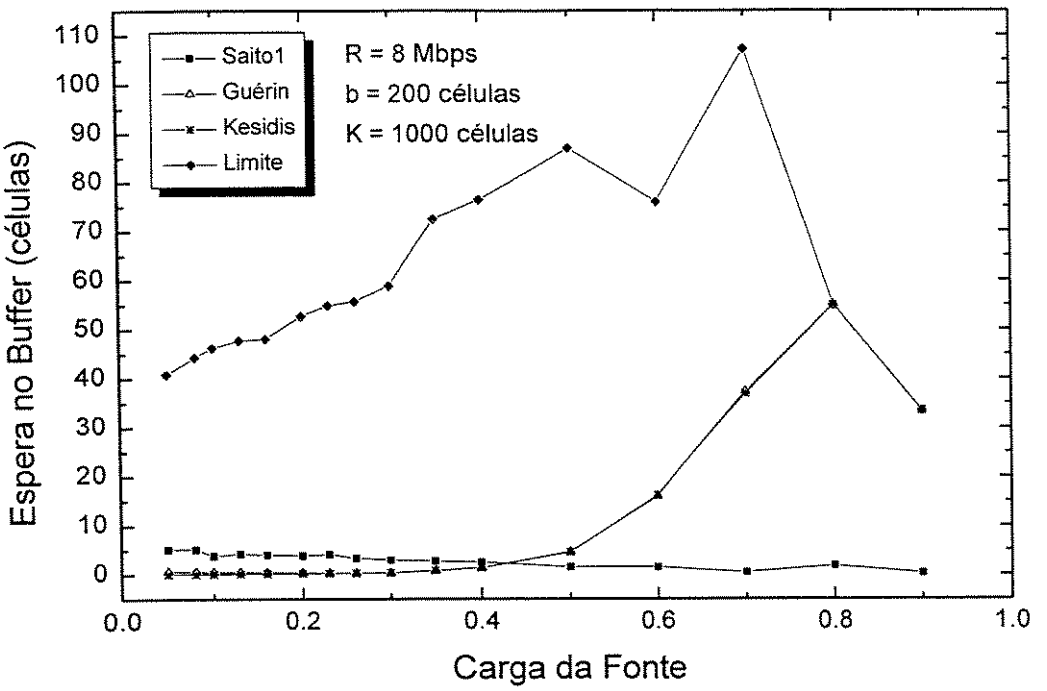


Figura 4.11: *Tempo médio de espera no buffer em função da carga das fontes.*

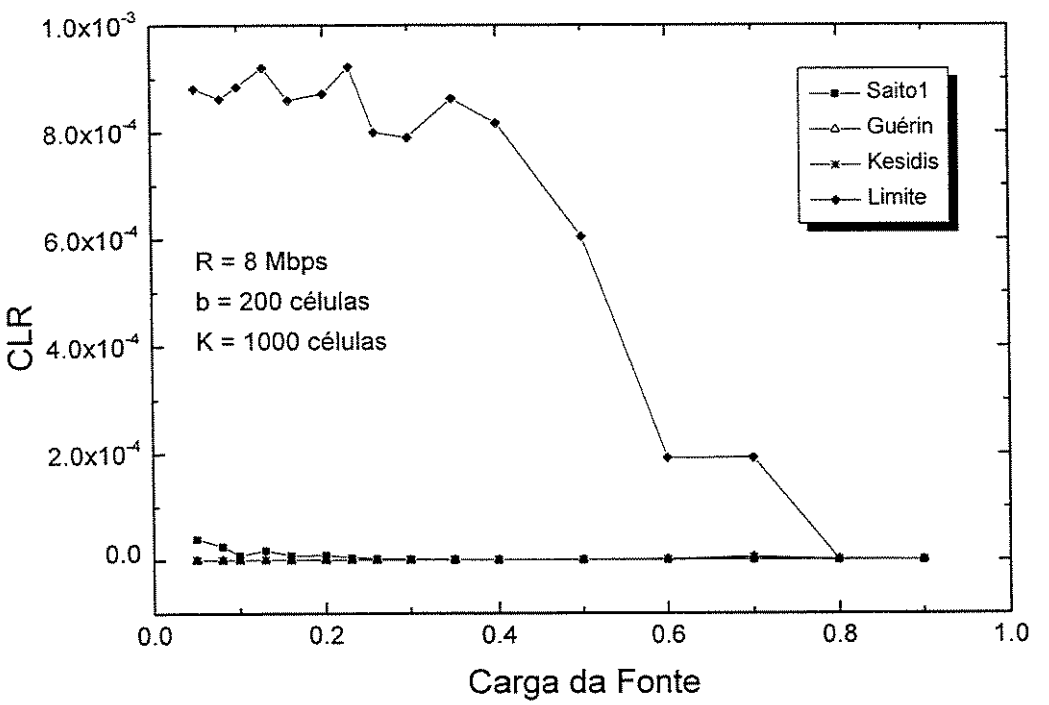


Figura 4.12: *Taxa de perda de células em função da carga das fontes.*

### 4.3.4 Variação dos Parâmetros de Alocação Dinâmica

O quarto conjunto de simulações tem por objetivo analisar o efeito dos parâmetros que controlam a estimação do tráfego no método de alocação dinâmica. Os parâmetros são a largura do intervalo de medição (em slots de células do enlace), a largura do período de renovação (em número de intervalos de medição) e o regulador da previsão exponencial $\alpha$. As combinações de valores utilizadas estão indicadas na Tab. 4.1.

| | Intervalo de Medição | Período de Renovação | $\alpha$ |
|---|---|---|---|
| saito1 | 80 | 10.000 | 0,2 |
| saito2 | 80 | 10.000 | 0,4 |
| saito3 | 80 | 1.000 | 0,2 |
| saito4 | 40 | 10.000 | 0,2 |
| saito5 | 160 | 10.000 | 0,2 |

Tabela 4.1

Como o objetivo não é analisar o desempenho, mas verificar que valores dos parâmetros resultam em boa estimativa de tráfego, são apresentados apenas os gráficos com o número de conexões aceitas e a taxa de perda de células resultante em cada uma das três variações estudadas anteriormente: *buffer*, surto e carga.

Verifica-se, pela Fig. 4.13, que a combinação Saito2 obteve maior número de aceitações. O parâmetro $\alpha$ provocou esse comportamento mais agressivo do algoritmo. O valor maior de $\alpha$ significa que o tráfego recentemente medido tem maior peso no cálculo final do tráfego estimado. Dessa forma, a estimação fica mais sujeita a oscilações momentâneas, mas também responde de maneira rápida a tendências do tráfego.

As combinações mais conservadoras foram Saito1 e Saito4. Aparentemente, a largura do período de renovação foi mais importante na medição do tráfego. As combinações Saito3 e Saito5 tiveram resultado intermediário. Saito5 utiliza intervalo de medição e período de renovação maiores que Saito3. A combinação Saito5 pode possibilitar uma melhor estimativa do tráfego. A CLR foi respeitada em todos os casos (Fig. 4.14).

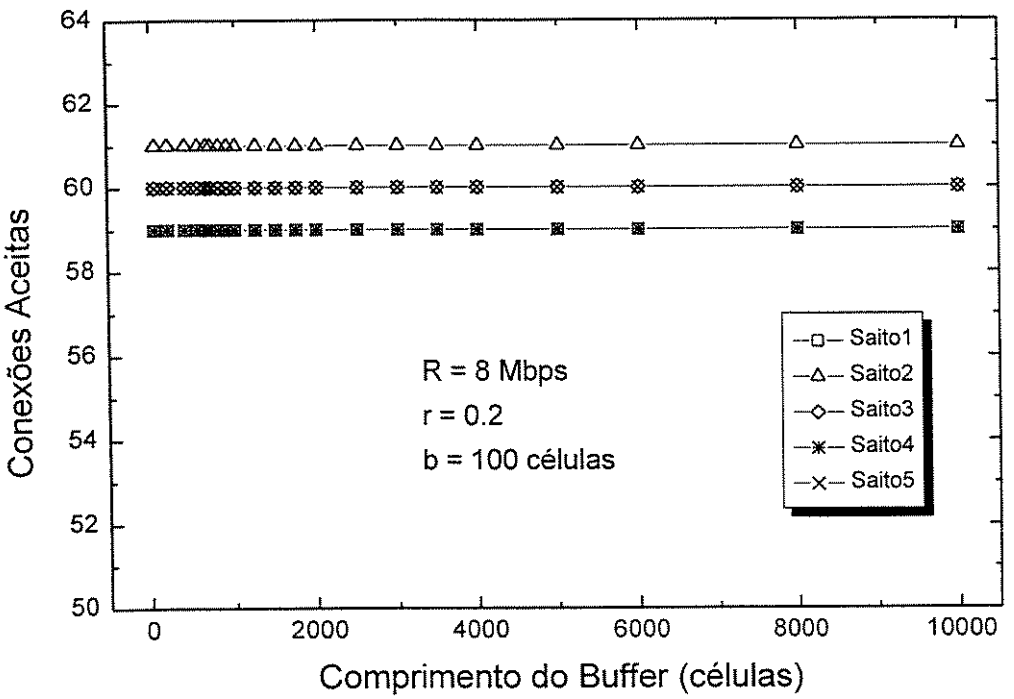


Figura 4.13: *Número de conexões aceitas em função do tamanho do buffer.*

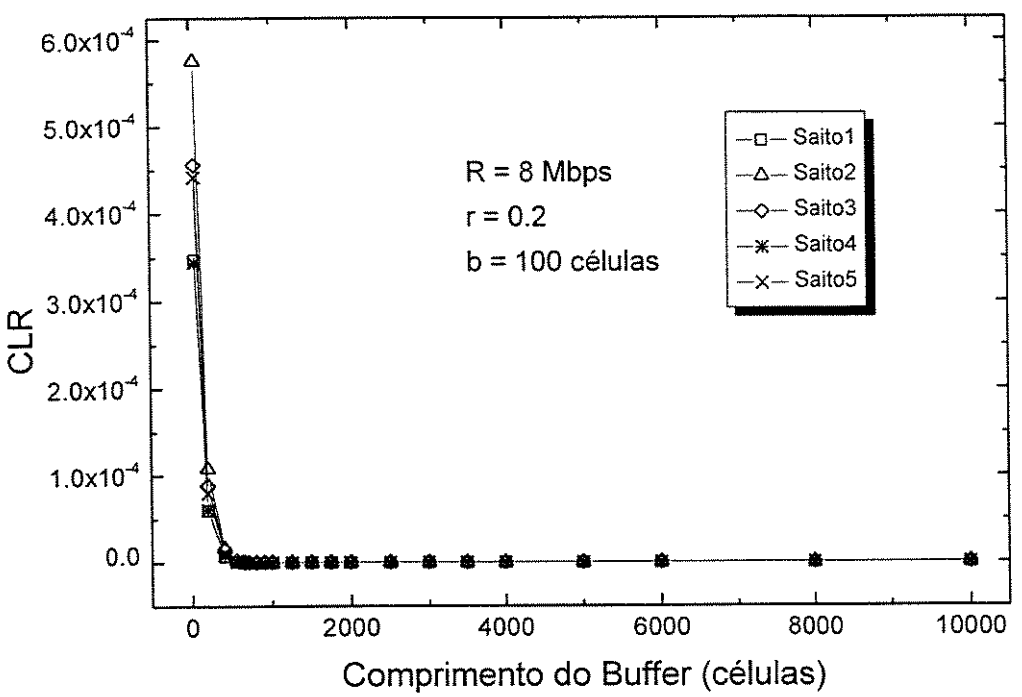


Figura 4.14: *Taxa de perda de células em função do tamanho do buffer.*

Na Fig. 4.15 estão exibidas apenas três curvas. A inclusão de todas as cinco opções resulta num gráfico de difícil interpretação visual. Novamente, a combinação que mais aceitou conexões foi Saito2, ressaltando o forte efeito que $\alpha$ possui.

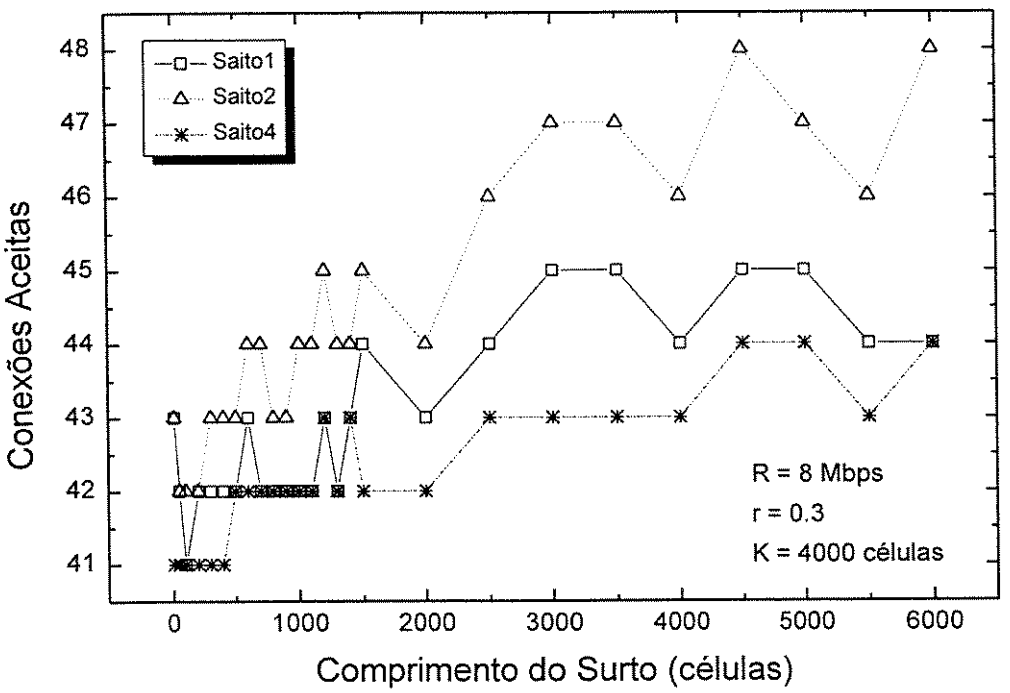


Figura 4.15: *Número de conexões aceitas em função do comprimento médio do surto.*

A taxa de perda de células está na Fig. 4.16. A aparente vantagem de Saito2 revela-se um problema, pois a CLR não é respeitada em vários pontos. Saito3 também viola a CLR, mas apenas em dois pontos. Com isso, o número de 1.000 amostras do período de renovação se mostra insuficiente para uma boa estimação do tráfego. As outras três combinações ultrapassam a CLR muito discretamente no último ponto, surto de 6.000 células. Logo, verifica-se que $\alpha$ precisa ser realmente pequeno para garantir a estabilidade necessária a uma boa decisão.

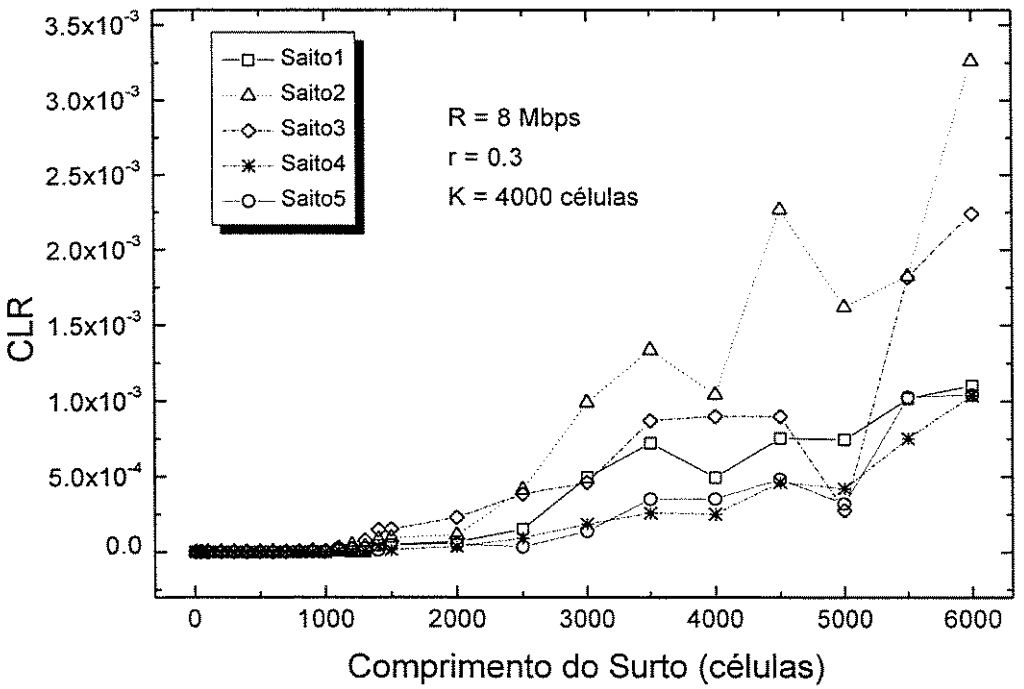


Figura 4.16: *Taxa de perda de células em função do comprimento médio do surto.*

No estudo da variação da carga (Fig. 4.17), todas as combinações tiveram resultados muito próximos, não havendo quase distinção entre eles. A taxa de perda de células (Fig. 4.18) obtida foi pequena e não houve violação do valor especificado.

Após as análises apresentadas, constata-se que o método é muito sensível a surtos longos. Nestes casos, as estimativas podem não ser precisas e o tráfego pode ser subestimado, provocando violação da CLR contratada. Para obter uma medição mais precisa do tráfego seria necessário aumentar o intervalo de medição e o período de renovação. O tamanho do *buffer* e a carga das fontes não comprometeram a segurança do algoritmo em nenhuma das combinações estudadas.

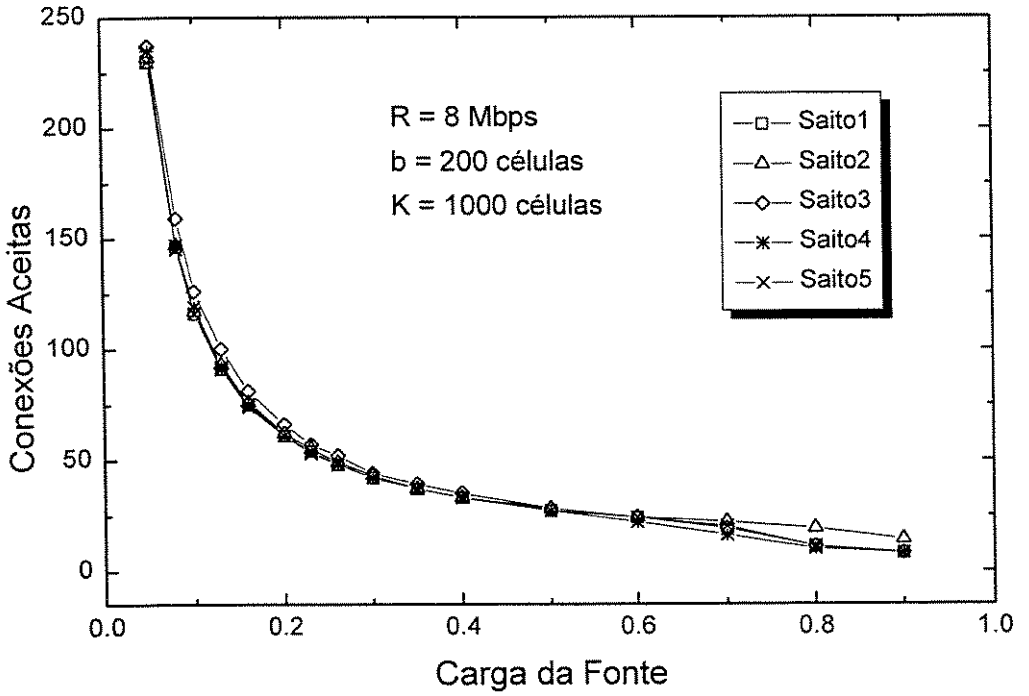


Figura 4.17: *Número de conexões aceitas em função da carga das fontes.*

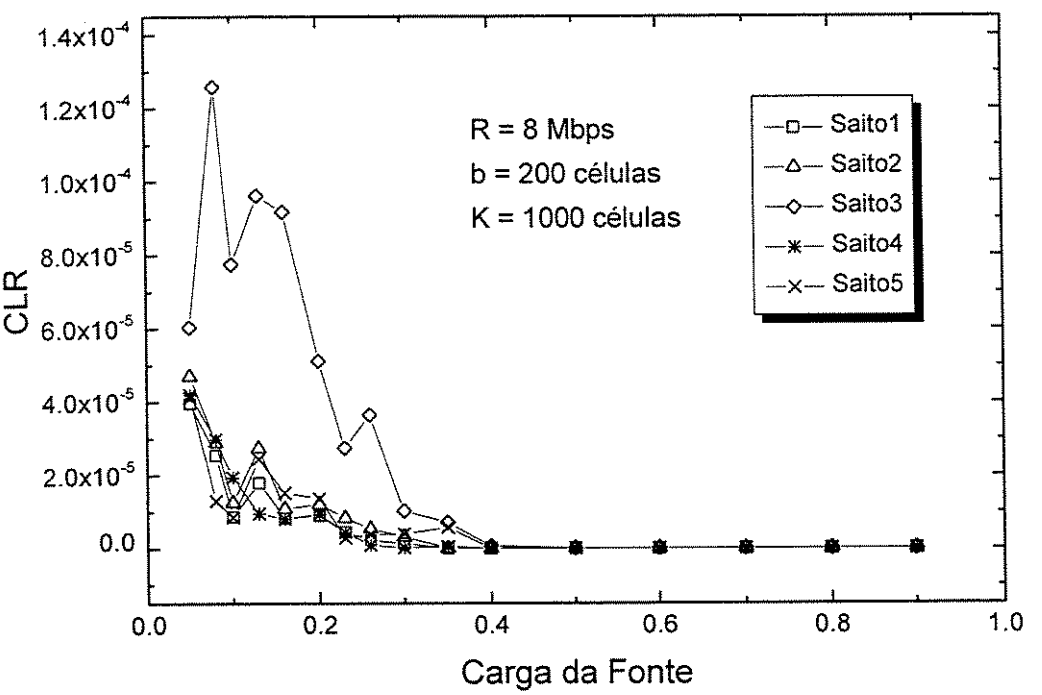


Figura 4.18: *Taxa de perda de células em função da carga das fontes.*

## 4.4 Confiabilidade dos Resultados

Num estudo de inferência estatística como o aqui apresentado, costuma-se validar os resultados através da apresentação do intervalo de confiança resultante. Os gráficos mostrados nas seções anteriores não contém a indicação do intervalo de confiança para evitar o excesso de informações que poderia prejudicar a boa interpretação das curvas.

O intervalo de confiança quantifica a confiança que pode ser atribuída aos resultados de uma simulação, num sentido estatístico. Por exemplo, um intervalo de confiança de 90% significa que em 90% das execuções da simulação, o valor real da medida está dentro do intervalo calculado. Nos outros 10% de ocasiões, o valor real está fora do intervalo calculado. A proporção de amostras onde o valor real está dentro do intervalo calculado é chamado de *cobertura*.

Se o número de amostras $n$ for suficientemente grande, um intervalo de confiança percentual aproximado $100(1-\beta)$ é dado por [19]

$$\overline{X}(n) \pm t_{n-1,\,1-\beta/2}\sqrt{\frac{s^2(n)}{n}}$$

onde,

$$s^2(n) = \frac{\sum_{i=1}^{n}\left[X_i - \overline{X}(n)\right]^2}{n-1}; \qquad \overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n};$$

$t_{n-1,1-\beta/2}$ é o ponto crítico $(1-\beta/2)$ superior para a distribuição $t$ com $n$-1 graus de liberdade.

As simulações foram divididas em lotes e as informações necessárias para calcular o intevalo de confiança dos tempos de espera e da utilização foram coletados. Não foi calculado o intervalo de confiança para a taxa de perda de células, pois este valor tem apenas uma amostra por simulação. Para calcular o intervalo de confiança da perda de células, seria necessária a execução de várias simulações idênticas com exceção das sementes do gerador de números pseudo-aleatórios.

O valor escolhido para o intervalo de confiança foi de 95% ($\beta = 0.05$). Os valores médios dos intervalos obtidos e seus respectivos desvios padrão estão nas tabelas 4.2 a 4.4.

A média indicada nas tabelas refere-se à razão entre o valor do intervalo de confiança para mais ou para menos do valor medido e o próprio valor medido, ou seja, representa em termos percentuais a dimensão do intervalo unilateral em relação ao valor medido.

Por exemplo, para uma utilização medida de 0,5 e intervalo de confiança unilateral de 0,05, a faixa onde o valor real pode ser encontrado em 95% dos casos vai de 0,45 a 0,55. Fazendo a razão (0,05/0,5), obtém-se o valor de 10%.

| | Utilização | | Tempo de Espera | |
|---|---|---|---|---|
| | Média (%) | Desvio Padrão | Média (%) | Desvio Padrão |
| Saito1 | 0,15 | $5{,}5 \cdot 10^{-4}$ | 5,7 | 0,84 |
| Limite | 0,12 | 0,02 | 2,6 | 0,27 |
| Guérin | 0,14 | 0,02 | 3,8 | 1,12 |
| Kesidis | 0,16 | 0,04 | 3,5 | 1,5 |

Tabela 4.2: *Variação do Buffer*

| | Utilização | | Tempo de Espera | |
|---|---|---|---|---|
| | Média (%) | Desvio Padrão | Média (%) | Desvio Padrão |
| Saito1 | 0,4 | 0,18 | 15,7 | 5,8 |
| Limite | 0,37 | 0,19 | 9,4 | 5,8 |
| Guérin | 0,45 | 0,22 | 35,7 | 32,1 |
| Kesidis | 0,5 | 0,28 | 8,4 | 8,1 |

Tabela 4.3: *Variação do Surto*

| | Utilização | | Tempo de Espera | |
|---|---|---|---|---|
| | Média (%) | Desvio Padrão | Média (%) | Desvio Padrão |
| Saito1 | 0,17 | 0,07 | 6,61 | 3,67 |
| Limite | 0,15 | 0,07 | 3,23 | 0,87 |
| Guérin | 0,18 | 0,09 | 5,97 | 3,03 |
| Kesidis | 0,23 | 0,15 | 3,1 | 2,8 |

Tabela 4.4: *Variação da Carga*

Verifica-se pelas tabelas que a utilização média calculada nas simulações é bastante exata. O tempo de espera, em alguns casos, apresentou intervalos de confiança muito grandes, notadamente no estudo de variação do surto. De maneira geral, os intervalos indicam uma boa confiabilidade nos resultados.

## 4.5 Conclusão

As simulações implementadas foram discutidas neste capítulo. Vários gráficos, para várias combinações de parâmetros, foram mostrados. Uma análise específica do algoritmo de alocação dinâmica foi apresentada. A confiabilidade dos resultados obtidos foi discutida a partir da análise do intervalo de confiança.

Os algoritmos de capacidade equivalente mostraram-se bastante conservadores na maioria dos casos. O método de alocação dinâmica obteve bons resultados, mas mostrou baixa confiabilidade em tráfego com surtos longos.

# CAPÍTULO 5

# CONCLUSÃO GERAL

Os principais aspectos que afetam o controle de conexões em rede ATM foram apresentados no Cap. 2. As categorias de serviço definidas pelo ATM Forum foram discutidas e o uso dos parâmetros descritores de tráfego em cada categoria foi ressaltado. O capítulo foi encerrado com uma discussão de várias das propostas existentes para a rotina de CAC (Connection Admission Control).

As propostas escolhidas para estudo foram apresentadas detalhadamente no Cap. 3. Duas das propostas estudadas são baseadas em capacidade equivalente, ou banda efetiva, e a outra consiste num método de alocação dinâmica, onde a decisão é baseada no tráfego efetivamente medido na rede.

O desempenho de três algoritmos de controle de admissão de conexões foi analisado no Cap. 4. Simulações em linguagem C foram implementadas para modelar o comportamento de um multiplexador ATM submetido ao tráfego agregado de várias fontes. Esta estrutura foi utilizada com cada um dos três algoritmos de CAC. A taxa de perda de células foi o parâmetro de qualidade de serviço utilizado. Para avaliar se as propostas são eficientes ou conservadoras foi obtido o limite de aceitação de conexões em cada caso através de tentativas exaustivas.

Todos os algoritmos mostraram-se conservadores. A proposta de Kesidis, Walrand e Chang [2] teve o pior desempenho dentre os três. O método de Saito e Shiomoto [4] conseguiu bons resultados em várias situações, mas teve baixo desempenho com *buffers* grandes mostrando-se insensível ao seu tamanho.

O algoritmo de Saito e Shiomoto merece um estudo mais aprofundado dos problemas inerentes ao método de alocação dinâmica. O estudo desenvolvido neste trabalho não investigou o efeito da taxa de chegada de chamadas no desempenho deste método. É

preciso investigar se um número elevado de chamadas num curto período de tempo compromete o bom desempenho do algoritmo.

## Trabalhos Futuros

O uso de fontes correlacionadas como MMBP, MMFP ou auto-similares pode mostrar um comportamento diferente dos algoritmos. Como várias fontes reais de tráfego apresentam correlação, esse estudo será importante.

A avaliação do efeito da taxa de chegada de chamadas no desempenho do método de alocação dinâmica de Saito e Shiomoto [4] se faz necessária, pois num comutador ATM a taxa de conexão e desconexão deve ser elevada devido à combinação de muitas fontes com tempos de serviço variados.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] R. Guérin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," IEEE J. Select. Areas Commun., vol. 9, pp. 968-981, 1991.

[2] G. Kesidis, J. Walrand, and C.-S. Chang, "Effective bandwidths for multiclass Markov fluids and other ATM sources," IEEE Trans. Networking, vol. 1, no. 4, pp. 424-28, Aug. 1993.

[3] H. Michiel, and K. Laevens, "Teletraffic engineering in a broad-band era," Proceedings of the IEEE, vol. 85, no. 12, pp. 2007-2033, Dec. 1997.

[4] H. Saito, and K. Shiomoto, "Dynamic call admission control in ATM Networks," IEEE J. Select. Areas Commun., vol. 9, no. 7, pp. 982-989, 1991.

[5] A. W. Berger, and W. Whitt, "Effective Bandwidths with priorities," IEEE/ACM Transactions on Networking, vol. 6, no. 4, pp. 447-460, 1998.

[6] D. Anick, D. Mitra, and M. M. Sondhi, "Stochastic theory of a data-handling system with multiple sources," Bell Sys. Tech. J., vol. 61, pp. 1871-1894, 1982.

[7] C. Courcoubetis, G. Kesidis, A. Ridder, J. Walrand, and R. Weber, "Admission control and routing in ATM networks using inferences from measured buffer occupancy," IEEE Trans. Comm., Feb/Mar/Apr, 1995.

[8] I. Hsu, and J. Walrand, "Dynamic bandwidth allocation for ATM switches," J. Appl. Prob., 33, pp. 758-771, 1996.

[9] R. Bolla, F. Davoli, and M. Marchese, "Bandwidth allocation and admission control in ATM networks with service separation," IEEE Comm. Magazine, pp. 130-137, May, 1997.

[10] J.-Y. Le Boudec, and R. Nagarajan, "A CAC algorithm for VBR connections over a VBR trunk," Proceedings of 15th International Teletraffic Congress – ITC, vol. 1, pp. 59-70, Washington, DC, USA, 1997.

[11] J. C. Oliveira, I. S. Bonatti, P. L. D. Peres and A. K. Budri, "Cell level performance approach for link dimensioning of ATM networks," Proceedings of ITS 98, vol.1, pp. 189-194, São Paulo, SP, Brazil, 1998.

[12] H. G. Perros and K. M. Elsayed, "Call admission control schemes: a review," IEEE Communications Magazine, November 1996.

[13] W. E. Leland, W. Willinger, M. S. Taqqu and D. V. Wilson, "On the self-similar nature of Ethernet-traffic (extended version)," IEEE/ACM Trans. Networking, vol.2, pp. 1-15, 1994.

[14] ITU_T, "B-ISDN ATM layer cell transfer performance Recommendation I.356," 1996.

[15] ITU_T, "Traffic control and congestion control in B-ISDN Recommendation I.371," 1995.

[16] K. Liu, D. W. Petr, C. Braun, "A measurement-based CAC strategy for ATM networks," Proc. ICC'97, IEEE, vol. 3, pp. 1714-18, 1997.

[17] K. Sohraby, "On the asymptotic behavior of heterogeneous statistical multiplexer with applications," Proc. INFOCOM 92, pp. 839-847.

[18] G. L. Choudhury, D. M. Lucantoni and W. Whitt, "Squeezing the most out of ATM," IEEE Trans. on Commun., vol. 44, no. 2, February 1996.

[19] A. M. Law and W. D. Kelton, *Simulation modeling and analysis*. McGraw-Hill, 1982.

[20] ATM Forum, *Traffic Management Specification, Version 4.0*. April, 1996.

[21] N. Giroux, S. Ganti, *Quality of Service in ATM Networks*. NJ: Prentice Hall PTR, 1999.

# Anexo

O código fonte desenvolvido para a simulação do algoritmo de alocação dinâmica está listado neste anexo. A simulação dos outros algoritmos utiliza o mesmo código com exceção da exclusão das rotinas de medição de tráfego, CAC, cálculo da distribuição de probabilidade de chegada de células e o uso de varíaveis específicas de cada método.

```
/*****************************************************************************

   programa para testar o calculo da probabilidade de erro usando o
   metodo proposto por Saito, implementando o CAC e simulando com
   trafego agregado.

   Geracao de trafego on-off, submetido a um buffer, calculo do CLR e
   comparacao com o valor retornado pelo algoritmo.

   22/09/99 - Niudomar Siqueira

*****************************************************************************/

#include <math.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>

/****************************** CONSTANTES ***************************************/

#define N (short int) 600   // comp. do vetor p; vai de 0 a (N-1)
              // prob. de chegar 0 ate´ prob. de chegar (N-1) = N possibilidades

  // quando N_SLOTS = 80, 100 fontes implica a taxa maxima de 412 celulas por
  // intervalo de medicao s.
  // R1 = N_SLOTS*8/155.52

#define DURACAO 900.0   // duracao da simulacao apos a chegada de todas as fontes

#define N_FONTES (int) 120       // no. de fontes que vao solicitar conexao
#define EPSILON 1e-3          // CLR desejado
#define TAXA_ENLACE 155.52e6  // em bps

#define TAXA_CELULAS (TAXA_ENLACE/(53.0*8.0)) // taxa do enlace em celulas/segundo
#define SERVICO (1.0/(TAXA_CELULAS))      // tempo de transmissao de 1 celula no enlace
#define N_CELULAS (DURACAO*TAXA_CELULAS) // n total de slots de tempo do enlace na simul
#define N_AMOSTRAS 100000L     // janela de 100.000 slots p/ calcular media
#define INV_N_AMOSTRAS (1.0/(N_AMOSTRAS))
#define TAM_VETOR (unsigned int) ((N_CELULAS)/(N_AMOSTRAS)) //N de medias da simul.

#define ALPHA 0.2          // parametro da previsao exponencial
#define N_SLOTS 80          // tempo de medicao do trafego em celulas do enlace
#define RENOVACAO 10000  // comprimento do periodo de renovacao

#define s (N_SLOTS*SERVICO) // duracao do periodo de medicao - em unidade de tempo
#define T_RENOVACAO (RENOVACAO*s)  // duracao em seg do periodo de renov.

/*****************************************************************************/

#define NOMEARQ "saitol"   // arquivo de saida: NOMEARQ.rel

#define K 50          // tamanho do buffer em celulas

double R = 8e6;          // taxa de pico da fonte

double r = 0.2;          // utilizacao da fonte

double b2 = 100;         // comprimento medio do surto


/*****************************************************************************/

#define PI 3.1415926535897
#define seedA 534        // sementes do gerador de numeros pseudo-aleatorios
#define seedB 972
#define seedC 249

#define LIMITE 4294967295U  // limite de uma variavel unsigned int

/*****************************************************************/


time_t hora;       // para registrar dia e hora da simulacao no relatorio
FILE *arq;
char arq_rel[13] = NOMEARQ;     // arquivo de relatorio
```

```
int n_conexoes = 0;    // no. de conexoes efetivadas
int fontes_analisadas = 0;    // para registrar e permitir a deteccao de quando
                              // todas as fontes ja´ passaram pelo CAC.
int conexoes_ativas[N_FONTES+10];    // registro de quais fontes estao conectadas

unsigned long int cel_ger = 0, cel_per = 0;  // n de celulas geradas e perdidas pelas
                                             // fontes conectadas
double vetor_tempo[TAM_VETOR+10];            // vetor p/ armazenar as medias do buffer
double vetor_util[TAM_VETOR+10];     //p/ armazenar as medias de utilizacao do enlace
double vetor_buffer[TAM_VETOR+10];     //p/ armazenar as medias de tam. do buffer

unsigned int num_janela = 0;    // indexa dos vets que armaz. medias de buffer e util.
unsigned long int enlace_ocupado = 0;  // p/ calcular a utilizacao
unsigned long int tam_buffer_acumulado = 0;  // p/ calcular o buffer medio
unsigned long int num_slots = 0;//contador interno a jan. de cada media buffer e util.
unsigned long int total_cel_atendidas = 0; //total de cel. atendidas em toda a simul.

unsigned int medias_tempo = 0;  // indexador do vetor que armazena as medias de tempo
unsigned long int tempo_espera_acumulado = 0; // p/ tempo medio de espera
unsigned long int total_celulas_enfileiradas = 0;  // p/ tempo medio de espera


char recusa = 0;           // registra se pedido de conexao foi recusado
double  t_atual = 0.0, tempo_reinicio;    // instante de tempo atual da simulacao

unsigned int tam_buffer = 0;     // tamanho atual do buffer;
unsigned int max_buffer = 0;  // comprimento maximo atingido pelo buffer

double  T_celula, P, Q;          // variaveis descritoras da fonte on-off

double  p[N+10], q[N+10];         // distribuicao de prob. de chegada de celulas em s
                                  // p - resultado da composicao com previsao exponencial
                                  // q - resultado medido
double  a = 0.0, b = 0.0;    // a = media da cheg. de celulas em s (exponencial)
                             // b = media medida
double  theta[N+8], ultimo_B1 = 0.0;   // theta - calculado usando R1 e a1 da fonte
unsigned int num_amostras_B1 = 0;    // B1 e' a CLR prevista pelo algoritmo

int R1;    // taxa de pico em s da conexao solicitada
           // R1 = N_SLOTS*8/155.52
double a1;  // taxa media em s da conexao solicitada

struct evento {
   double   tempo;         // instante de tempo em que o evento deve ocorrer
   int no;                 // no. do no' que provoca o evento. Enlace = N_FONTES+1
   struct evento  *prox;  // aponta para o proximo evento da lista encadeada
} *entrada = NULL, *fim = NULL;


/*****************************  PROTOTIPOS  ****************************************/

double calcula_CLR(void);
void simula(void);
void store(struct evento *i);
void qstore(void /* short int *celula_no */);
double ran(void);
double intervalo(void);
void inicializa(void);
void cac(struct evento *conexao);
void enfileira(struct evento *celula);
void transmite(struct evento *celula);
void relatorio(void);
void main(void);
void mede_trafego(void);
void atualiza_p(void);
double convolucao(short int k);

/*********************************************************************************/


void qstore()
{
   if (tam_buffer != 0) {
      tempo_espera_acumulado += (tam_buffer);
   }
   if (tempo_espera_acumulado >= (LIMITE - K)) {  /* limite do tipo unsigned int */
```

```
      printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
      fprintf(arq, "ERRO!  Ultrapassou o limite de uma variavel inteira\n");
      fprintf(arq, "Erro na rotina: qstore\n");
      fclose(arq);
      exit(1);
   }
   tam_buffer++;
   total_celulas_enfileiradas++;
   if (tam_buffer > max_buffer) max_buffer = tam_buffer;
   if (total_celulas_enfileiradas == N_AMOSTRAS) {     // completou um ciclo de amostras
          medias_tempo++;
          vetor_tempo[medias_tempo-1] = tempo_espera_acumulado*INV_N_AMOSTRAS;
          tempo_espera_acumulado = 0;
          total_celulas_enfileiradas = 0;
   }
}

/*************************************************************/


double ran(void)
{

/*
       Algoritmo de Wichmann-Hill para geracao de numeros aleatorios.
       Periodo de aproximadamente 7E12.
*/

   int k = 30269, m = 30307, n = 30323;
   double u;
   static long int x = seedA, y = seedB, z = seedC;
   x = (x*171) % k;
   y = (y*172) % m;
   z = (z*170) % n;
   u = (double) x/30269 + (double) y/30307 + (double) z/30323;
   return (u - (int) u);
}

/*************************************************************/

double intervalo(void)
{
   double tempo;
   tempo = T_celula;
   if (ran() <= P) do
      tempo += T_celula;
   while (ran() > Q);
   return (tempo);
}

/*************************************************************/

void main(void)
{
   inicializa();
   simula();
   relatorio();
}

/*************************************************************/

void inicializa(void)
{
   struct evento *ev = NULL;
   int i;     // deve ser do mesmo tipo de n_conexoes e evento.no


   strcat(arq_rel, ".rel");
   arq = fopen(arq_rel, "w");
   if (!arq) {
      printf("ERRO - ERRO - ERRO - ERRO - ERRO \n \
              Nao foi possivel abrir arquivo de relatorio - Erro posicao 6\n\n \
              Verificar rotina: inicializa\n\n");
      exit(1);
   }
   hora = time(NULL);
   fprintf(arq, ctime(&hora));
```

```
   fprintf(arq, "\n");

   if (T_RENOVACAO > DURACAO) {
      printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
      fprintf(arq, "ERRO!  Tempo de simulacao insuficiente para a medicao de \
                    trafego\n\n");
      fprintf(arq, "Verificar constantes DURACAO, N_SLOTS e RENOVACAO\n");
      fclose(arq);
      exit(1);
   }
   if (N_CELULAS < N_AMOSTRAS) {
      printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
      fprintf(arq, "ERRO!  Tempo de simulacao insuficiente para a medicao de \
trafego\n\n");
      fprintf(arq, "Verificar constantes N_CELULAS e N_AMOSTRAS\n");
      fclose(arq);
      exit(1);
   }

   R /= (53.0*8.0);         // taxa de pico em celulas/seg
   T_celula = 1.0/R;    // comprimento em tempo da celula na fonte
   P = 1.0/b2;          // P = prob. de sair do surto e entrar em silencio
   Q = r/(b2*(1-r));    // Q = prob. de sair do silencio e entrar no surto
   for(i=0; i < N; i++) {
      p[i] = 0.0;
      q[i] = 0.0;
      theta[i] = 0.0;
   }
   for(i=0; i <= N_FONTES; i++) {
      conexoes_ativas[i] = 0;
   }
   R1 = ((int) ceil(R*s));      // R1 ja' foi convertido para celulas/segundo.
   a1 = R1*r;                  // adaptado ao intervalo de medicao s
   theta[0] = 1-(a1/((double) R1));
   theta[R1] = a1/((double) R1);

   if (!(ev = malloc(sizeof(struct evento)))) {
      printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
      fprintf(arq, "ERRO!  Sem memoria para novo evento - Erro posicao 2\n\n");
      fprintf(arq, "Verificar rotina: inicializa\n");
      fclose(arq);
      exit(1);
   }
   ev->tempo = t_atual;     // t_atual e' inicializada com zero
   ev->no = N_FONTES + 1;  // agenda o primeiro servico do enlace
   ev->prox = NULL;
   store(ev);
   for(i=1; i <= N_FONTES; i++) {    // cria a 1a. celula de todas as fontes
      if (!(ev = malloc(sizeof(struct evento)))) {
         printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
         fprintf(arq, "ERRO!  Sem memoria para novo evento - Erro posicao 3\n\n");
         fprintf(arq, "Verificar rotina: inicializa\n");
         fclose(arq);
         exit(1);
      }
      ev->tempo = T_RENOVACAO*(i-1);

   // ATENCAO: deve estar de acordo com a constante DURACAO1
   // 1 chegada de nova conexao por periodo de renovacao
   // para permitir que a medicao do trafego possa surtir efeito no CAC

      ev->no = i;
      ev->prox = NULL;
      store(ev);
   }
}

/*************************************************************/

void simula(void)
{
   struct evento *ev;
   unsigned int k;
   double tempo_reinicio = 0.0; // para mudar referencia de tempo do inicio da simulacao

   do {
      ev = entrada;
```

```
      entrada = entrada->prox;
      if (ev->no <= N_FONTES) {
         cac(ev);
         if (!recusa) enfileira(ev);

      }
      else transmite(ev);
      free(ev);
   } while (fontes_analisadas < N_FONTES);
   tempo_reinicio = t_atual;

// limpa todas as estatisticas para comecar tudo novamente a partir do estado
// estacionario.

   cel_per = 0;
   for (k=0; k < TAM_VETOR; k++) {
      vetor_util[k] = vetor_buffer[k] = vetor_tempo[k] = 0.0;
   }
   total_cel_atendidas = 0;
   medias_tempo = 0;
   total_celulas_enfileiradas = tempo_espera_acumulado = 0;
   num_slots = enlace_ocupado = num_janela = 0;
   cel_ger = tam_buffer_acumulado = tam_buffer = 0;

   do {
      ev = entrada;
      if (ev->tempo > (DURACAO + tempo_reinicio)) return;
      entrada = entrada->prox;
      if (ev->no <= N_FONTES) enfileira(ev);
      else transmite(ev);
      free(ev);
   } while (1);
}

/*************************************************************/

void cac(struct evento *conexao)
{
   int k;

   recusa = 0;
   for (k = 0; k < n_conexoes; k++) {
      if ((conexoes_ativas[k]) == conexao->no) return;
   }
   if (calcula_CLR() > EPSILON) {
      recusa = 1;
      fontes_analisadas++;
      return;
   }
   else {
      conexoes_ativas[n_conexoes] = conexao->no;
      n_conexoes++;
      atualiza_p();
      fontes_analisadas++;

      cel_per = 0;
      for (k=0; k < TAM_VETOR; k++) {
         vetor_util[k] = vetor_buffer[k] = vetor_tempo[k] = 0.0;
      }
      total_cel_atendidas = 0;
      medias_tempo = 0;
      total_celulas_enfileiradas = tempo_espera_acumulado = 0;
      num_slots = enlace_ocupado = num_janela = 0;
      cel_ger = tam_buffer_acumulado = tam_buffer = 0;
           // para evitar que alguma variavel ultrapasse o limite TAM_VETOR
           // depois de todas as conexoes solicitadas, ha' nova limpeza de
           // variaveis e a simulacao comeca a tomar as estatisticas pra valer.
   }
}


/*************************************************************/

void enfileira(struct evento *celula)
{
   struct evento *p;
```

```
   t_atual = celula->tempo;
   cel_ger++;
   mede_trafego();
   if (tam_buffer == K) {
      cel_per++;    // transbordo
   }
   else qstore(/* &(celula->no) */);  // informacao nao utilizada por enquanto
   if (!(p = malloc(sizeof(struct evento)))) {
      printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
      fprintf(arq, "ERRO!  Sem memoria para novo evento - Erro posicao 4\n\n");
      fprintf(arq, "Erro na rotina: enfileira\n");
      fclose(arq);
      exit(1);
   }
   p->tempo = t_atual + intervalo();
   p->prox = NULL;
   p->no = celula->no;
   store(p);
}

/*************************************************************/

void transmite(struct evento *celula)
{
   struct evento *p;
   t_atual = celula->tempo;

   num_slots++;
   if (tam_buffer > 0) {    // ha' pelo menos uma celula para ser atendida
      tam_buffer_acumulado += tam_buffer;
      tam_buffer--;      //  atualiza a ocupacao do buffer
      enlace_ocupado++;
   }
   if (num_slots == N_AMOSTRAS) {  // completou um ciclo de amostras
      num_janela++;
      vetor_buffer[num_janela-1] = tam_buffer_acumulado*INV_N_AMOSTRAS;
      vetor_util[num_janela-1] = enlace_ocupado*INV_N_AMOSTRAS;
      tam_buffer_acumulado = 0;
      total_cel_atendidas += enlace_ocupado;
      enlace_ocupado = 0;
      num_slots = 0;
   }
   if (!(p = malloc(sizeof(struct evento)))) {
      printf("ERRO - ERRO - ERRO. VERIFICAR ARQUIVO DE RELATORIO\n\n");
      fprintf(arq, "ERRO!  Sem memoria para novo evento - Erro posicao 5\n\n");
      fprintf(arq, "Erro na rotina: transmite\n");
      fclose(arq);
      exit(1);
   }
   p->tempo = t_atual + SERVICO;      // agenda proximo slot do enlace
   p->no = N_FONTES + 1;
   p->prox = NULL;
   store(p);
}

/*************************************************************/

void relatorio(void)
{
   unsigned int k;
   double   perda;
   double   util_enlace = 0.0;        // utilizacao media do enlace
   double   var_util = 0.0;       // variancia da utilizacao do enlace
   double   buffer_medio = 0.0;
   double      var_buffer = 0.0;     // variancia do tamanho medio do buffer
   double   tempo_espera = 0.0;       // tempo medio de espera das celulas no buffer
   double   var_tempo = 0.0;      // variancia do tempo medio de espera


   hora = time(NULL);
   fprintf(arq, "\n");
   fprintf(arq, ctime(&hora));
   fprintf(arq, "\n");

   total_cel_atendidas += enlace_ocupado;  // o ultimo ciclo de amostras e' incompleto
   fprintf(arq, "\nEpsilon %.0e\n", EPSILON);
   fprintf(arq, "Numero de Fontes %d\n", N_FONTES);
```

```
   fprintf(arq, "Tempo de Simulacao   %.1f segundos\n\n", DURACAO);
   fprintf(arq, "*****************************************************************\n\n");
   fprintf(arq, "R  %.0e  bps\n", R*424);
   fprintf(arq, "r  %.2f\n", r);
   fprintf(arq, "b2  %.0f  celulas\n\n", b2);
   fprintf(arq, "Buffer  %d\n\n", K);
   fprintf(arq, "*****************  Estimacao do Trafego  *********************\n\n");
   fprintf(arq, "Periodo de Medicao = %d  Periodo de Renovacao = %d  ALPHA = %G\n\n"\
                , N_SLOTS, RENOVACAO, ALPHA);
   fprintf(arq, "Ultimo CLR calculado = %e \n\n", (ultimo_B1/* /num_amostras_B1 */));
   fprintf(arq, "Numero de CLR's calculados: %d\n\n", num_amostras_B1);

   fprintf(arq, "*****************************************************************\n\n");
   perda = ((double) cel_per)/((double) cel_ger);
   fprintf(arq, "Numero de conexoes efetivadas = %d\n\n", n_conexoes);

   fprintf(arq, "\nTotal de Celulas Geradas  Total de Celulas Perdidas  CLR Geral\n");
   fprintf(arq, "%u                      %u              %e\n\n\n", cel_ger, cel_per, perda);

   fprintf(arq, "   ****************      Estatisticas do Enlace     ************\n\n");
   fprintf(arq, "Utilizacao     Variancia    Num. Amostras      Celulas Atendidas\n");
   util_enlace = 0.0;
   for (k=0; k < TAM_VETOR; k++) {
      util_enlace += vetor_util[k];
   }

   util_enlace /= TAM_VETOR;
   var_util = 0.0;
   for (k=0; k < TAM_VETOR; k++) {
      var_util += pow((vetor_util[k] - util_enlace), 2);
   }
   var_util /= (TAM_VETOR-1);
   fprintf(arq, "%G       %G        %u", util_enlace, var_util, TAM_VETOR);
   fprintf(arq, "              %u\n\n\n\n", total_cel_atendidas);

   fprintf(arq, "   ****************      Estatisticas do Buffer     ************\n\n");
   fprintf(arq, "Compr. Medio  Variancia    Num. de Amostras   Compr. Maximo\n");
   buffer_medio = 0.0;
   for (k=0; k < TAM_VETOR; k++) {
      buffer_medio += vetor_buffer[k];
   }

   buffer_medio /= TAM_VETOR;
   var_buffer = 0.0;
   for (k=0; k < TAM_VETOR; k++) {
      var_buffer += pow((vetor_buffer[k] - buffer_medio), 2);
   }
   var_buffer /= (TAM_VETOR-1);
   tempo_espera = 0.0;
   for (k=medias_tempo; k > 0; k--) {
      tempo_espera += vetor_tempo[k-1];
   }

   tempo_espera /= medias_tempo; // tempo medio de espera no buffer em celulas
   var_tempo = 0.0;
   for (k=medias_tempo; k > 0; k--) {
      var_tempo += pow((vetor_tempo[k-1] - tempo_espera), 2);
   }
   var_tempo /= (medias_tempo-1);
   fprintf(arq, "%G   %G       %u                %u\n\n\n\n", buffer_medio, \
                var_buffer, TAM_VETOR, max_buffer);
   fprintf(arq, "Tempo de Espera     Variancia       Num. de Amostras\n");
   fprintf(arq, "%G            %G              %u\n\n\n\n", tempo_espera, var_tempo, \
                medias_tempo);
   fprintf(arq, "Comprimento Atual = %u\n\n\n\n", tam_buffer);
   if ((tam_buffer + total_cel_atendidas + cel_per) != cel_ger) {
      fprintf(arq, "ERRO: numero de celulas atendidas, perdidas e restantes no \
buffer nao \n esta de acordo com o numero de celulas geradas.");
   }
   fclose(arq);
}

/*********************************************************************/

void store(struct evento *i)          /* novo elemento a armazenar  */
{
   struct evento *velho, *p;
```

```
  if (!fim) {            /* lista estava vazia */
     i->prox = NULL;
     entrada = i;
     fim = i;
     return;
  }
  p = entrada;

  velho = NULL;
  while(p) {
     if (i->tempo > p->tempo) {
        velho = p;
        p = p->prox;
     }
     else {
        if(velho) {    /* fica no meio */
           velho->prox = i;
           i->prox = p;
           return;
        }
        i->prox = entrada;     /* novo primeiro elemento */
        entrada = i;
        return;
     }
  }
  fim->prox = i;  /* coloca no final */
  i->prox = NULL;
  fim = i;
}

/*****************************************************************/


double calcula_CLR(void)
{

   short int      k;
   double  B1 = 0.0, soma = 0.0, convol = 0.0;

   for(k = N_SLOTS + 1; k <= N; k++) {
      convol = 0.0;
      soma += (k - N_SLOTS)*convolucao(k);
   }
   B1 = soma/(a1 + a);
   ultimo_B1 = B1;
   num_amostras_B1++;
   return(B1);
}

/*****************************************************************/

void mede_trafego(void)
{
   static int contador = 0;
   int pula_slot, sobra, completa_renovacao;
   static int p1[N+2];
   static double inicio_slot = 0.0, inicio_renovacao = 0.0;
   short int i, pula_n_renovacoes;

   if ((t_atual - inicio_renovacao) <= T_RENOVACAO) {
      if ((t_atual - inicio_slot) <= s) {
         contador++;
      }
      else {
         pula_slot = (int) (((t_atual - inicio_slot)/s) - 1.0);
         p1[0] += pula_slot;  // slot(s) inteiro(s) sem nenhuma chegada;
         if (contador >= N) {
            printf("ERRO - ERRO - ERRO - VERIFICAR ARQUIVO DE RELATORIO\n\n");
            fprintf(arq, "ERRO! Contagem de celulas dentro de um slot de \
medicao ultrapassou\n");
            fprintf(arq, "o limite fixado em %d celulas. Rotina mede_trafego. \
Programa Abortado.\n\n", N);
            printf("Numero de conexoes ativas = %d\n\n", n_conexoes);
            printf("Tempo atual = %f\n\n", t_atual);
            fclose(arq);
            exit(1);
         }
```

```
         p1[contador]++;
         contador = 1;  // a chegada da 1a. celula no slot seguinte
         inicio_slot += s*(pula_slot + 1);
      }

   }
   else {
      pula_slot = (int) (((t_atual - inicio_slot)/s) - 1.0);
      completa_renovacao = (int) ((inicio_renovacao + T_RENOVACAO - inicio_slot)/s);
      sobra = pula_slot - completa_renovacao;
      p1[0] += completa_renovacao;    // slot(s) inteiro(s) sem nenhuma chegada;
      if (contador >= N) {
            printf("ERRO - ERRO - ERRO - VERIFICAR ARQUIVO DE RELATORIO\n\n");
            fprintf(arq, "ERRO! Contagem de celulas dentro de um slot de \
medicao ultrapassou\n");
            fprintf(arq, "o limite fixado em %d celulas. Rotina mede_trafego. \
Programa Abortado.\n\n", N);
            fclose(arq);
            exit(1);
         }
      p1[contador]++;
      for(i=0; i < N; i++) {
         q[i] = ((double) p1[i])/((double) RENOVACAO);
      }
      b = 0.0;
      for(i=0; i < N; i++) {
         b += i*q[i];
         // atualiza o vetor p e a media a
         p[i] = ALPHA*q[i] + (1-ALPHA)*p[i];
         p1[i] = 0;
      }
      a = ALPHA*b + (1-ALPHA)*a;
      pula_n_renovacoes = sobra/RENOVACAO;
      if (pula_n_renovacoes >= 1) {
         q[0] = 1.0; // nao chegou nenhuma celula em todo o periodo
         for (i=1; i < N; i++) {
            q[i] = 0.0;
         }
         for(i=1; i <= pula_n_renovacoes; i++) {
            for(i=0; i < N; i++) {    // atualiza o vetor p e a media a
               p[i] = ALPHA*q[i] + (1-ALPHA)*p[i];
            }
            a = /* ALPHA*b  + */ (1-ALPHA)*a;
         }
      }
      contador = 1;  // a chegada da 1a. celula no slot seguinte
      inicio_renovacao += (1 + pula_n_renovacoes)*T_RENOVACAO;
      sobra -= pula_n_renovacoes*RENOVACAO;
      inicio_slot = inicio_renovacao + sobra*s;
   }
}

/*****************************************************************/

void atualiza_p(void)
{
   short int k;

   a += a1;
   for(k=0; k < N; k++) {
      p[k] = convolucao(k);
   }
}

/*****************************************************************/

double convolucao(short int k)
{
   double convol = 0.0;

   convol = p[k]*theta[k-k];
   if  ((k-R1) >= 0) {
      convol+= p[k-R1]*theta[R1];
   }
   return(convol);
}
```