Estudo de um Método para Análise Transiente de Modelos Markovianos de Sistemas de Comunicação [1]

João A.N. da Silva[1] Rosa Leão[2]

[1] Instituto Militar de Engenharia - RJ
Praça General Tibúrcio, 80
Rio de Janeiro - RJ

[2] Universidade Federal do Rio de Janeiro
Coppe/Sistemas
Cx.P. 68511, Rio de Janeiro, RJ 21945-970

E-mail:{abdalla@ime.eb.br, rosac@cos.ufrj.br}

Resumo

Os avanços crescentes na tecnologia tem possibilitado o aparecimento de sistemas de comunicação cada vez mais sofisticados. A modelagem e análise é portanto uma etapa essencial na concepção e desenvolvimento desses sistemas. O método da uniformização tem sido amplamente utilizado para análise transiente, contudo ele apresenta custo computacional alto para modelos *stiff*. Neste trabalho foi avaliado o desempenho de um método recentemente proposto para a obtenção de medidas transientes. Foram estudados dois modelos na área de redes de computadores que apresentam como características: *stiffness* e grande cardinalidade do espaço de estados. Foi mostrada a eficiência computacional do método, que é bastante superior a da uniformização, e também como medidas importantes para modelos de sistemas de comunicação podem ser facilmente obtidas.

Abstract

The rapid advances in the technology have resulted in sophisticated communication systems. Modeling and analysis are of major importance during the conception and development of that systems. One of the most widely used technique to transient analysis is the uniformization method. However, the computational cost required to calculate transient probabilities is very large for stiff models. In our work we evaluate the performance of a method recently proposed to calculate transient measures. We study two models in the domain of computer networks which are stiff and have a large state space cardinality. It was shown that the method is computationally very efficient when compared with uniformization and also that important transient measures for communication systems can be easily obtained.

[1] Este trabalho tem o suporte parcial do CNPq (ProTeM e auxílio integrado).

1 Introdução

Modelos markovianos tem sido amplamente usados para avaliar o quanto um sistema é eficiente, confiável ou o quanto está disponível. Algumas medidas tipicamente calculadas são: a probabilidade do sistema estar em um estado ou conjunto de estados (que representem uma situação particular), a fração de tempo que o sistema passou em determinado conjunto de estados durante $(0, t)$, tempo esperado em que o sistema passará pela primeira vez por um conjunto de estados.

Em todas as medidas descritas acima, quando o período de observação t é grande em relação ao tempo entre a ocorrência de eventos do sistema, a solução em estado estacionário é uma boa aproximação. Entretanto, para muitos casos, o intervalo de interesse é relativamente curto, não sendo válidos os resultados da análise em estado estacionário. Para estes casos, é necessário fazer a análise transiente.

Existem vários métodos para a obtenção de medidas transientes em modelos markovianos. O método mais popular de todos é o método da uniformização [7]. Sua popularidade se deve principalmente à sua robustez numérica, sua interpretação probabilística e à simplicidade de implementação.

O custo computacional da uniformização é proporcional a Λt [3], onde Λ é no mínimo igual ao maior valor absoluto dos elementos da diagonal do gerador infinitesimal irredutível Q e $(0, t)$ é o intervalo de observação. Devido a isto, seu custo pode ser alto para modelos onde as taxas de transição dos eventos diferem de diversas ordens de grandeza (modelos *stiff*) [6].

Um método aproximado para o cálculo das probabilidades de estado em um intervalo finito foi proposto por Ross [9]. Nele, as probabilidades para um tempo t são aproximadas pelas probabilidades calculadas para um tempo aleatório com distribuição erlangiana. No método é necessário fazer a inversão de uma matriz cuja ordem é igual ao número de estados do modelo. Embora esta matriz (que é derivada da matriz de transição de estados do sistema) seja tipicamente esparsa, a sua inversão, além de ser computacionalmente cara, destrói esta característica. Em consequência, o método se torna de pouca utilidade para modelos com grande cardinalidade do espaço de estados.

Em [4] foi proposto um método de solução alternativo que evita a inversão de matrizes. Este método usa uma técnica iterativa de solução de equações lineares [5]. Foi mostrado através de pequenos exemplos que o custo computacional exigido na solução de modelos *stiff* foi menor do que o custo do método da uniformização, sem perda significativa de precisão. Posteriormente, em [2], foi proposto um método direto de solução. Foi observado um desempenho significativamente melhor do método direto para modelos *stiff* ou modelos que possuam uma estrutura especial da matriz de transição de estados, quando comparado à uniformização.

O objetivo deste artigo é avaliar a eficiência dos métodos propostos em [4, 2] quando usados na modelagem de sistemas de comunicação. Serão explorados dois exemplos representativos da área de redes de computadores. Através dos exemplos pode-se ter uma idéia da relevância de medidas transientes para os modelos estudados e comprovar a eficiência dos métodos. O primeiro exemplo modela um canal de comunicação multiplexador servindo duas filas distintas segundo um esquema de *polling*. É um modelo *stiff*, o que leva a uma solução de alto custo com a técnica da uniformização. O segundo exemplo é um modelo construído com a finalidade de estudar o comportamento do tráfego gerado por uma fonte TCP sendo transmitido sobre um nó ATM. Trata-se de um modelo de grande magnitude, que serve ao propósito de comprovar a eficiência dos métodos para modelos com grande

cardinalidade do espaço de estados.

O artigo está organizado da forma descrita a seguir. Na seção 2 é feita uma descrição sumária do método de Ross e são apresentados os métodos propostos em [2]. Os resultados do modelo do canal multiplexador estão na seção 3. Na seção 4 os resultados do modelo do protocolo TCP sobre uma rede ATM são descritos. Finalmente, apresentamos as conclusões e os trabalhos futuros na seção 5.

2 Métodos para Análise Transiente

Nesta seção introduziremos primeiramente o método proposto por Ross. Em seguida apresentamos os métodos que desejamos avaliar a eficiência quando usados nos exemplos das seções subsequentes.

2.1 O método de Ross

A aproximação de Ross [9] baseia-se no cálculo de $\pi_i(t)$ (probabilidade do processo $\mathcal{X}$ estar no estado i no instante t) para um intervalo de tempo aleatório com distribuição erlangiana.

Suponha que o sistema modelado pela cadeia de Markov de tempo contínuo $\mathcal{X}$ é observado a intervalos que são variáveis aleatórias $Y_1, Y_2, \cdots$ com taxa exponencial λ e independentes de $\mathcal{X}$. Seja o evento associado à observação de $\mathcal{X}$ designado por $\mathcal{E}$. Pode-se dizer que, se $\mathcal{X}$ está no estado i, uma transição em $\mathcal{X}$ ocorrerá antes de um novo evento $\mathcal{E}$ (a próxima observação) com probabilidade $q_i/(q_i + \lambda)$. Da mesma forma, um evento $\mathcal{E}$ ocorrerá antes de uma transição em $\mathcal{X}$ com probabilidade $\lambda/(q_i + \lambda)$. Estas são repectivamente as probabilidades de o sistema sair do estado i e de permanecer nele antes de uma nova observação ($\mathcal{E}$). No primeiro caso pode-se ainda dizer que o sistema fará uma transição para um estado j com probabilidade q_{ij}/q_i.

Seja a variável aleatória erlangiana definida por $Y(r) = Y_1 + \cdots + Y_r$. A equação recursiva seguinte pode ser obtida, condicionando-se no fato de $\mathcal{X}$ fazer ou não uma transição antes de $\mathcal{E}$.

$$\pi_{ij}(Y(r)) = \frac{1}{(q_i + \lambda)} \sum_{k \neq i} \pi_{kj}(Y(r)) q_{ik} + \frac{\lambda}{(q_i + \lambda)} \pi_{ij}(Y(r-1)) \qquad (1)$$

onde $\pi_{ij}(Y(0)) = \delta_{ij}$.

A mesma equação na forma matricial fica:

$$\mathbf{\Pi}(r) = (\mathbf{I} - \mathbf{Q}/\lambda)^{-1} \mathbf{\Pi}(r-1) \qquad (2)$$

onde $\mathbf{\Pi}(r) = [\pi_{ij}(Y(r))]$.

Pode-se aplicar esta equação um número R de vezes, chegando-se às probabilidades no instante $Y(R)$ em função das probabilidades de estado iniciais. Observe que $Y(R)$ é uma variável aleatória erlangiana de parâmetros (λ, R), cuja média é dada por $E[Y(R)] = R/\lambda$. Como se desejam as probabilidades no instante t, escolhe-se $\lambda = R/t$, o que garante $E[Y(R)] = t$. Além disso, a variância de $Y(R)$ diminui com o aumento de R, tendendo a 0 quando $R \to \infty$, fazendo com que $Y(R) \to t$. Portanto R pode ser escolhido, empiricamente, grande o suficiente para se obter a precisão desejada.

2.2 Os métodos iterativo e direto

No método de Ross, o uso da equação 2 requer a inversão da matriz $(\mathbf{I}-\mathbf{Q}/\lambda)$, o que é uma operação cara. Os métodos descritos nesta seção obtém a solução desejada, procurando preservar a característica esparsa e as estruturas especiais presentes na matriz citada, reduzindo os seus custos de processamento e de armazenamento.

A nova abordagem [2] consiste em inicialmente uniformizar-se o processo original obtendo-se um processo equivalente com matriz de probabilidades de transição $\mathbf{P}$ onde o (i,j)-ésimo elemento p_{ij} é a probabilidade do sistema passar do estado i para o estado j. É importante lembrar que o tempo decorrido em um estado antes de qualquer transição é exponencialmente distribuído com taxa Λ.

Seguindo o mesmo procedimento usado para obter a equação 1, condicionando-se em $\mathcal{X}$ fazer ou não uma transição antes de $\mathcal{E}$ durante o último intervalo exponencial, tem-se:

$$\pi_{ij}(Y(r)) = \frac{\Lambda}{(\Lambda+\lambda)} \sum_k \pi_{kj}(Y(r))p_{ik} + \frac{\lambda}{(\Lambda+\lambda)}\pi_{ij}(Y(r-1)) \tag{3}$$

onde $\pi_{ij}(Y(0)) = \delta_{ij}$.

Seja $\mathbf{1}_j$ o vetor cujo j-ésimo elemento é 1, e todos os demais são 0. Seja $\mathbf{z}(j,r) = \Pi(r)\mathbf{1}_j$. A partir da equação 3 obtemos:

$$\left(\mathbf{I} - \frac{\Lambda}{(\Lambda+\lambda)}\mathbf{P}\right)\mathbf{z}(j,r) = \frac{\lambda}{(\Lambda+\lambda)}\mathbf{z}(j,r-1) \tag{4}$$

Fazendo-se $\lambda = R/t$, $\mathbf{z}(j,R)$ é uma boa aproximação da j-ésima coluna de $\mathbf{\Pi}(t)$. Ou seja, o i-ésimo elemento do vetor $\mathbf{z}(j,R)$ é uma boa aproximação de $\pi_{ij}(t)$. Daqui em diante $\mathbf{z}(j,r)$ será escrito simplesmente por $\mathbf{z}(r)$ para simplificar a notação.

A equação 4 pode ser generalizada para obter-se a probabilidade do sistema estar em um conjunto de estados $\mathcal{D}$, em um instante t (partindo de qualquer um dos estados). Basta definir $\mathbf{1}_\mathcal{D}$ como o vetor que possui 1 nas posições correspondentes aos estados contidos em $\mathcal{D}$, e 0 nas demais posições. Fazendo $\mathbf{z}(\mathcal{D},r) = \mathbf{\Pi}(r)\mathbf{1}_\mathcal{D}$ e substituindo $\mathbf{z}(j,r)$ por $\mathbf{z}(\mathcal{D},r)$ (e $\mathbf{z}(j,r-1)$ por $\mathbf{z}(\mathcal{D},r-1)$) na equação 4, obtém-se a equação desejada. A equação é válida, pois $\mathbf{z}(\mathcal{D},r)$ é igual a soma de todas as colunas de $\mathbf{\Pi}(r)$ correspondentes aos estados contidos em $\mathcal{D}$.

Uma outra medida que pode ser obtida a partir da equação 4, é o valor esperado de uma variável aleatória do sistema sendo modelado. Seja o vetor $\mathbf{y}$ um vetor de N elementos tal que y_i $(i = 1,2,\ldots,N)$ contém o valor da variável y quando o sistema se encontra no estado i. Pode-se ver que:

$$\mathbf{\Pi}(r).\mathbf{y} = \overline{\mathbf{y}}(r) \tag{5}$$

onde $\overline{\mathbf{y}}(r)$ é tal que seu i-ésimo elemento $\overline{y}_i(r)$ é igual ao valor esperado de y no passo r dado que o sistema partiu do estado i no instante inicial.

Portanto, temos:

$$\left(\mathbf{I} - \frac{\Lambda}{(\Lambda+\lambda)}\mathbf{P}\right)\overline{\mathbf{y}}(r) = \frac{\lambda}{(\Lambda+\lambda)}\overline{\mathbf{y}}(r-1) \tag{6}$$

que possui a mesma forma da equação 4.

A solução do problema passa pela resolução (R vezes) da equação linear 4, que é da forma $\mathbf{Ax} = \mathbf{b}$. Na seção 2.2.1 será abordado o uso de um método iterativo para a resolução das referidas equações lineares. A seção 2.2.2 aborda o método direto elaborado para resolver estas equações.

2.2.1 O método iterativo

É sabido que se a matriz de taxas de transição do modelo sendo estudado for esparsa, então a matriz $(\mathbf{I} - \frac{\Lambda}{(\Lambda+\lambda)}\mathbf{P})$ também será esparsa, o que faz do modelo um candidato natural ao uso da técnica iterativa. Devido ao seu bom desempenho, o método SOR [5] foi escolhido para resolver a equação 4. Esta equação será resolvida R vezes (com $r = 1, 2, \cdots, R$). Supondo que serão necessárias N iterações, em média, para cada valor de r, a solução final terá um custo de aproximadamente RN multiplicações vetor $\times$ matriz.

O lado direito da equação 4 é o produto de uma constante pela solução no passo anterior. É sabido que o número de iterações de um método iterativo pode ser reduzido se a solução inicial é uma aproximação da solução desejada. Logo, se $\mathbf{z}(r-1)$ é uma boa aproximação de $\mathbf{z}(r)$, o número total de iterações pode ser reduzido. Pode-se mostrar [2, 4], que $\mathbf{z}(r-1)$ é uma aproximação razoável de $\mathbf{z}(r)$, principalmente para os maiores valores de r, o que comprovadamente reduz o número total de iterações.

2.2.2 O método direto

Muitos modelos de sistemas de computação e comunicação apresentam matrizes com estruturas especiais como por exemplo, matrizes banda em geral e as matrizes bloco-Hessenberg em particular. Se o método preserva a estrutura da matriz, evitando seu enchimento, o espaço de armazenamento necessário e o número de operações com blocos serão reduzidos.

No método direto descrito a seguir, considera-se a matriz $\mathbf{A} = \mathbf{I} - \frac{\Lambda}{(\Lambda+\lambda)}\mathbf{P}$ particionada em blocos como a seguir:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1K} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{K1} & \mathbf{A}_{K2} & \cdots & \mathbf{A}_{KK} \end{bmatrix} \tag{7}$$

Esta matriz é inicialmente submetida à técnica da redução, descrita adiante, que manipula os blocos, transformando-os, o que requer algumas multiplicações de blocos e a inversão dos blocos diagonais. Após esse procedimento, os blocos da matriz modificada são usados na determinação de $\mathbf{z}(r)$.

Considere o vetor $\mathbf{z}(r)$ particionado também em K blocos $\mathbf{z}_1(r), \mathbf{z}_2(r), \cdots, \mathbf{z}_K(r)$, com dimensões correspondendo ao particionamento de $\mathbf{A}$. A equação 4 pode ser reescrita como:

$$\sum_{j=1}^{K} \mathbf{A}_{ij}^{(0)} \mathbf{z}_j(r) = \mathbf{w}_i^{(0)}(r) \qquad i = 1, 2, \cdots, K \tag{8}$$

onde,

$$\mathbf{A}_{ij}^{(0)} = \mathbf{A}_{ij} \quad \text{e} \quad \mathbf{w}_i^{(0)}(r) = \frac{\lambda}{(A+\lambda)}\mathbf{z}_i(r-1) \tag{9}$$

Pode-se mostrar [2] que os blocos da diagonal $\mathbf{A}_{ii}$ são inversíveis e, portanto, pode-se resolver a primeira equação de 8 (com $i = 1$) para $\mathbf{z}_1$, obtendo a equação 10. O índice r será omitido nos vetores $\mathbf{z}$ e $\mathbf{w}_i$ por conveniência da notação.

$$\mathbf{z}_1 = -(\mathbf{A}_{11}^{(0)})^{-1} \sum_{j=2}^{K} \mathbf{A}_{1j}^{(0)}\mathbf{z}_j + (\mathbf{A}_{11}^{(0)})^{-1}\mathbf{w}_1^{(0)} \tag{10}$$

substituindo $\mathbf{z}_1$ em 8, tem-se:

$$\sum_{j=2}^{K} \underbrace{[\mathbf{A}_{ij}^{(0)} - \mathbf{A}_{i1}^{(0)}(\mathbf{A}_{11}^{(0)})^{-1}\mathbf{A}_{1j}^{(0)}]}_{\mathbf{A}_{ij}^{(1)}}\mathbf{z}_j = \underbrace{[\mathbf{w}_i^{(0)} - \mathbf{A}_{i1}^{(0)}(\mathbf{A}_{11}^{(0)})^{-1}\mathbf{w}_1^{(0)}]}_{\mathbf{w}_i^{(1)}} \qquad i = 2, 3, \ldots, K \tag{11}$$

Esta equação é a base do procedimento da redução. Note que, como ela é computada para todo $2 \leq i, j \leq K$, pode ser escrita como $\mathbf{A}^{(1)}\mathbf{z}^{(1)} = \mathbf{w}^{(1)}$, onde $\mathbf{z}^{(1)}$ é idêntico a $\mathbf{z}$, porém sem o primeiro bloco $\mathbf{z}_1$, $\mathbf{A}^{(1)}$ possui $(K-1) \times (K-1)$ blocos numerados $\mathbf{A}_{22}^{(1)}, \mathbf{A}_{23}^{(1)}, \ldots, \mathbf{A}_{KK}^{(1)}$, e $\mathbf{w}^{(1)}$ tem $K-1$ blocos numerados de $\mathbf{w}_2^{(1)}$ a $\mathbf{w}_K^{(1)}$.

Aplica-se este procedimento progressivamente para $k = 2, 3, \ldots, K-1$. Após $K-1$ passos obtemos:

$$\mathbf{A}_{KK}^{(K-1)}\mathbf{z}_K = \mathbf{w}_K^{(K-1)} \tag{12}$$

Resolvendo a equação acima obtemos $\mathbf{z}_K$. Os outros blocos do vetor $\mathbf{z}$ podem ser obtidos através de:

$$\mathbf{z}_i = (\mathbf{A}_{ii}^{(i-1)})^{-1}\left[\mathbf{w}_i^{(i-1)} - \sum_{j=0}^{(K-i-1)} \mathbf{A}_{i,K-j}^{(i-1)}\mathbf{z}_{K-j}\right] \tag{13}$$

Em resumo, a redução é inicialmente aplicada à matriz $\mathbf{A}$, o que só é feito uma única vez. Em seguida, para cada passo r, cada bloco do vetor $\mathbf{w}$ *reduzido* é calculado usando blocos da matriz $\mathbf{A}$ *reduzida*. Calcula-se finalmente, em cada passo r, o vetor $\mathbf{z}$ usando o recém-calculado vetor $\mathbf{w}$ e a matriz $\mathbf{A}$ *reduzidos*.

O procedimento completo pode ser esquematizado da seguinte maneira:

1. Inicialização

$$\mathbf{A}^{(0)} = \left(\mathbf{I} - \frac{\Lambda}{(A+\lambda)}\mathbf{P}\right) \tag{14}$$

2. Redução da matriz $\mathbf{A}$

- Para $1 \leq k \leq K-1$ e $k+1 \leq i, j \leq K$:

$$\mathbf{A}_{ij}^{(k)} = \mathbf{A}_{ij}^{(k-1)} - \mathbf{A}_{ik}^{(k-1)}(\mathbf{A}_{kk}^{(k-1)})^{-1}\mathbf{A}_{kj}^{(k-1)} \tag{15}$$

3. Cálculo dos vetores $\mathbf{z}(r)$

   Para $1 \leq r \leq R$:

   - Atualização de $\mathbf{w}^{(0)}(r)$

$$\mathbf{w}^{(0)}(r) = \mathbf{z}(r-1) * \frac{\lambda}{(\Lambda + \lambda)} \tag{16}$$

     Lembrando que para o primeiro passo $\mathbf{z}(0) = \mathbf{1}_j$ (ou $\mathbf{1}_{\mathcal{D}}$).

   - Cálculo de $\mathbf{w}(r)$:
     - para $1 \leq k \leq K-1$ e $k+1 \leq i \leq K$:

$$\mathbf{w}_i^{(k)}(r) = \mathbf{w}_i^{(k-1)}(r) - \mathbf{A}_{ik}^{(k-1)}(\mathbf{A}_{kk}^{(k-1)})^{-1}\mathbf{w}_k^{(k-1)}(r) \tag{17}$$

   - Cálculo de $\mathbf{z}(r)$:
     - para $1 \leq i \leq K$:

$$\mathbf{z}_i(r) = (\mathbf{A}_{ii}^{(i-1)})^{-1}\left[\mathbf{w}_i^{(i-1)}(r) - \sum_{j=0}^{(K-i-1)} \mathbf{A}_{i,K-j}^{(i-1)}\mathbf{z}_{K-j}(r)\right] \tag{18}$$

A figura 1 ilustra o algoritmo.

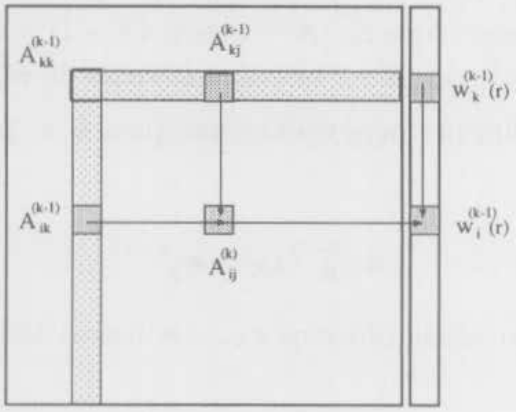


Figura 1: Uso dos blocos pelo algoritmo

Supondo a matriz $\mathbf{A}$ cheia, com $K \times K$ blocos todos com a mesma dimensão $(B \times B)$, o cálculo dos blocos $\mathbf{A}_{ij}^{(k)}$ no passo 2 terá um custo de $O((BK)^3)$ operações. No entanto admite-se [2] que este custo seja reduzido para estruturas especiais como matrizes-banda. Para matrizes que não possuem uma estrutura especial, pode ocorrer o enchimento da matriz. A partir da figura 1, podemos ver que um bloco $\mathbf{A}_{ij}^{(k)}$ nulo, pode tornar-se não nulo se os blocos $\mathbf{A}_{ik}^{(k-1)}$ e $\mathbf{A}_{kj}^{(k-1)}$ são não nulos. O cálculo para se obter $\mathbf{w}_k^{(k-1)}(r)$ requer $K(K+1)/2$ multiplicações de um vetor de tamanho B por uma matriz $B \times B$.

3 Um Canal Multiplexador

Neste exemplo é modelado um canal multiplexador servindo duas filas, q_1 e q_2, sujeitas a tráfegos distintos. A especificação do modelo e a geração da matriz de transição de estados $\mathbf{P}$ foram realizados usando-se a ferramenta TANGRAM-II [1]. É dada ênfase à aplicação do método direto, mostrando-se como estruturas especiais podem ser usadas para aumentar sua eficiência. Compara-se o desempenho do método com o do método da uniformização.

3.1 O modelo

O sistema consiste em um único servidor visitando cada fila de maneira cíclica. A fila q_i é servida enquanto houver pacotes ou quando expirar o seu tempo máximo de serviço (ou *time-out*) to_i. Quando as duas filas se encontram vazias, o servidor entra em um *estado de espera*, passando a servir imediatamente a primeira fila em que chegar um pacote.

Define-se como tc_i o tempo de comutação ou *switchover*, decorrido entre o fim do serviço à fila i e o começo do serviço à fila seguinte. Assume-se que os tempos entre chegadas de pacotes, tempos de serviço, bem como to_i e tc_i são variáveis aleatórias exponencialmente distribuídas.

Os parâmetros do sistema são mostrados na figura 2. Note que o valor de $tc_1 = tc_2 = 0{,}1\mu s$, muito menor que os demais tempos médios do modelo, o torna *stiff*, ou seja, suas taxas de transição diferem em várias ordens de grandeza. A cadeia de Markov resultante possui 1721 estados, e sua matriz de transição possui 7557 elementos não-nulos sendo portanto esparsa com *densidade* de aproximadamente $2{,}55*10^{-3}$.

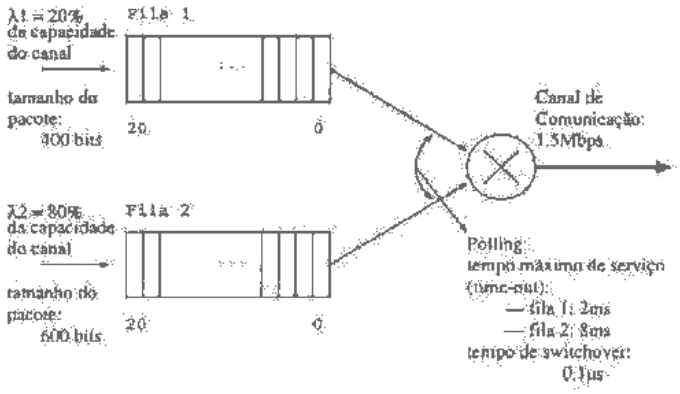


Figura 2: Canal multiplexador, com duas filas, operando por *polling*

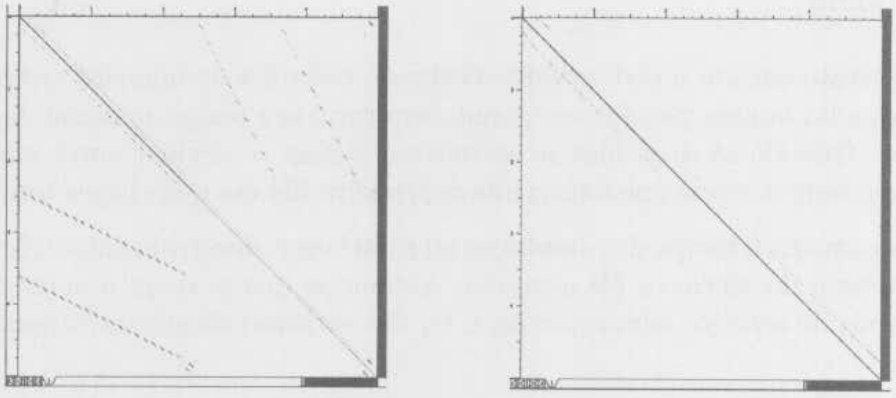
3.2 Identificação de estruturas especiais

Como citado na seção 2.2.2, se a matriz A possui uma estrutura especial, o custo computacional do método direto pode ser reduzido. Utilizamos um módulo da ferramenta TANGRAM-II que permite a reordenação dos estados do modelo possibilitando que seja verificado se a matriz de transição possui uma estrutura especial adequada ao método.

A matriz de transição de estados inicialmente obtida para o modelo apresenta a estrutura mostrada na figura 3-a. Nesta figura, os pontos escuros representam elementos não-nulos na matriz. Através de uma reordenação dos estados, foi possível obter uma matriz de transição de estados com a estrutura mostrada na figura 3-b.

De forma a não haver enchimento de blocos, foi feito um particionamento em 42 grupos onde o primeiro grupo possui 22 estados, seguido por 20 grupos de 41 estados intercalados com 20 grupos de 42 estados mais o último grupo de 39 estados, resultando numa matriz pentadiagonal por blocos.

Para comprovar o aumento de eficiência proporcionado por uma ordenação de estados e um particionamento adequados, foram computados os custos quando o modelo foi resolvido usando-se a matriz original (3-a) e quando o modelo foi resolvido usando-se a matriz reordenada (3-b) e o particionamento escolhido como descrito acima. Em ambos os casos usou-se um intervalo de tempo $t = 100$, um número de instantes de interesse $n = 20$ e



a) matriz original b) matriz reordenada

Figura 3: Matriz de transição de estados: a) original; b) após reordenação

número de passos $R = 10$ para o cálculo das medidas apresentadas na figura 4-c referentes à fila q_1.

No caso da matriz original, a redução consumiu 174.001.098 operações de multiplicação enquanto os cálculos de **w** e **z** consumiram ao todo 269.146.905 operações. Com a matriz reordenada e o particionamento escolhido, o custo da redução caiu para 7.863.606, e os cálculos de **z** e **w** para 34.489.824. Ou seja, o custo da redução diminuiu em *22 vezes*, e o custo dos cálculos de **w** e **z** foi dividido por um fator de *7,8*. A redução no tempo de computação total foi de 157s para 15,1s, uma queda de cerca de *10 vezes*.

3.3 As medidas realizadas

A probabilidade de perda de pacote é uma medida importante para se avaliar o desempenho do sistema. Para conhecer a evolução dessa probabilidade em certas situações, procura-se calcular a probabilidade de perda em um instante de tempo t a partir de uma *condição inicial*.

Neste exemplo foram usadas quatro condições iniciais. Na primeira, ambas as filas estão vazias, como no início da operação do sistema, ou após a interrupção e retomada do tráfego nas duas filas. Na segunda e terceira condições há uma fila cheia e outra vazia, o que pode decorrer de uma anormalidade em um dos tráfegos, em uma das filas ou no próprio servidor. A quarta condição é aquela em que as duas filas se encontram cheias, o que mostrará a reação do sistema após uma condição extremamente desfavorável.

Foram calculadas as probabilidades de perda nas duas filas, partindo das condições iniciais descritas, para dois intervalos, de 0 a 10ms e 0 a 100ms. Dentro de cada intervalo foram feitas medidas em 20 instantes de interesse, para cada condição inicial, com número de estágios $R = 10$.

A probabilidade de perda em uma dada fila em um determinado instante t é igual a soma das probabilidades dos estados (calculadas para o instante t) onde a fila se encontra cheia. Então identificou-se os estados em que cada fila se encontra cheia e calculou-se a *probabilidade desse conjunto de estados* para cada caso, da forma descrita na seção 2.2. Os resultados são apresentados nos gráficos da figura 4.

Para se entender o que acontece nas filas nestas quatro situações, foi calculado o número esperado de pacotes em cada fila, para os mesmos instantes descritos. Para tanto identificou-se, para cada fila, o número de pacotes presentes no *buffer* em cada um dos

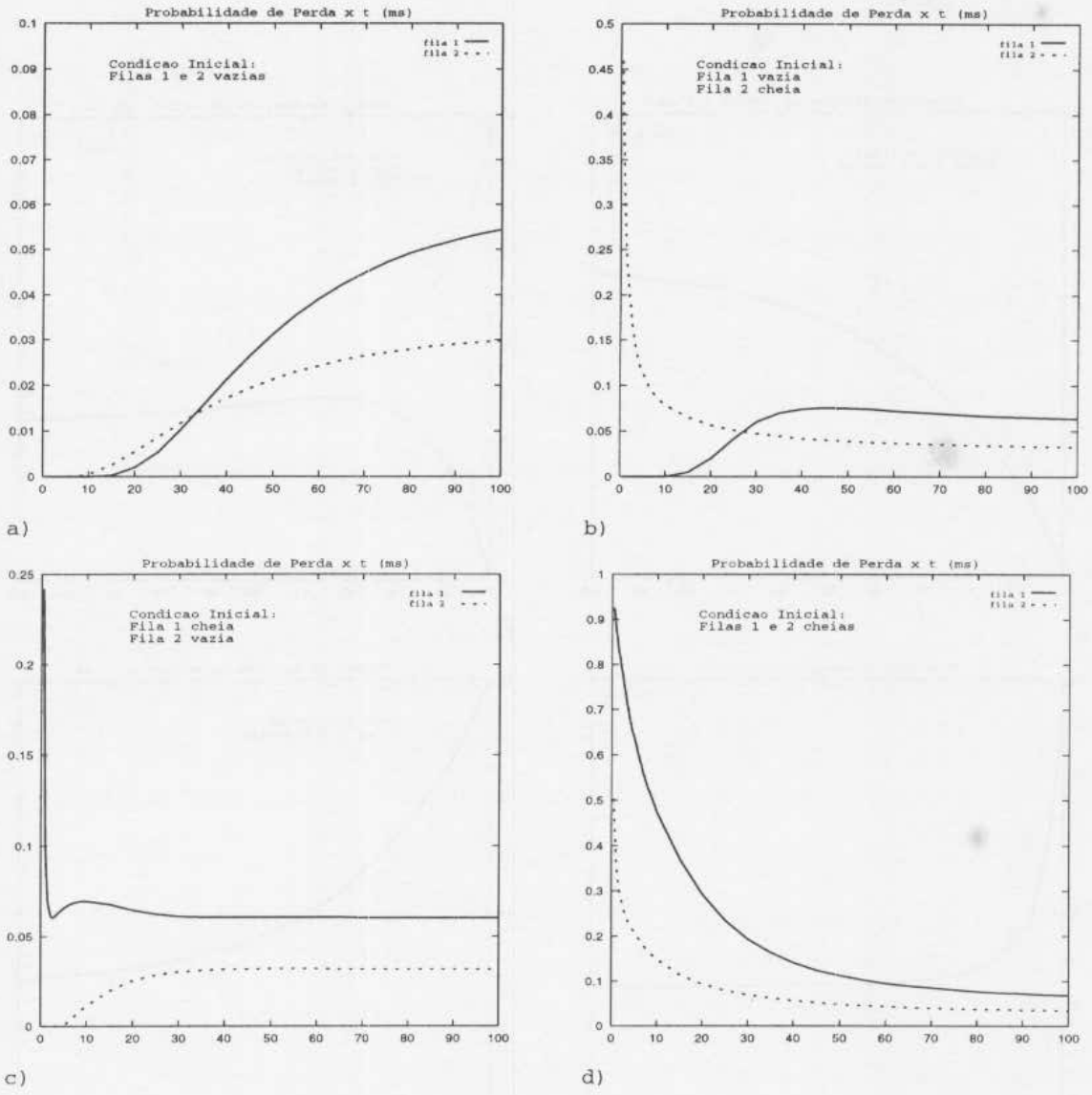


Figura 4: Probabilidades de perda nas filas q_1 e q_2

estados. Então definiu-se o vetor *número de pacotes na fila por estado* (vetor **y** da seção 2.2) e obteve-se o *valor esperado* do número de pacotes em cada fila. Nestes cálculos também foi usado $R = 10$. A partir dos resultados foram gerados os gráficos da figura 5.

Nas duas figuras observa-se que as probabilidades e o valores médios calculados tendem para os seus valores de estado estacionário independente da condição inicial, conforme esperado. Os valores de probabilidade de perda conferem com os valores de estado estacionário calculados usando o método GTH implementado na ferramenta TANGRAM-II, ou seja, cerca de $6{,}0 * 10^{-2}$ para a fila q_1 e $3{,}1 * 10^{-2}$ para a fila q_2.

Pela figura 5-a, pode-se ver que para $t < 30$ms o número de pacotes nas filas q_1 e q_2 está tipicamente abaixo de 4 e 6 respectivamente. Uma consequência disso é que a disputa das duas filas pelo canal é pequena e influi pouco no *throughput* (em pacotes/ms), que neste caso é maior na fila q_1 (por ter pacotes menores). Isto faz com que neste período a probabilidade de perda na fila q_2 seja maior que na fila q_1, como pode ser visto na figura 4-a.

Comparando-se os gráficos a e c da figura 5, vê-se que a fila q_2 se enche muito mais rápido no segundo caso, refletindo a influência do tráfego da outra fila sobre o seu *through-*

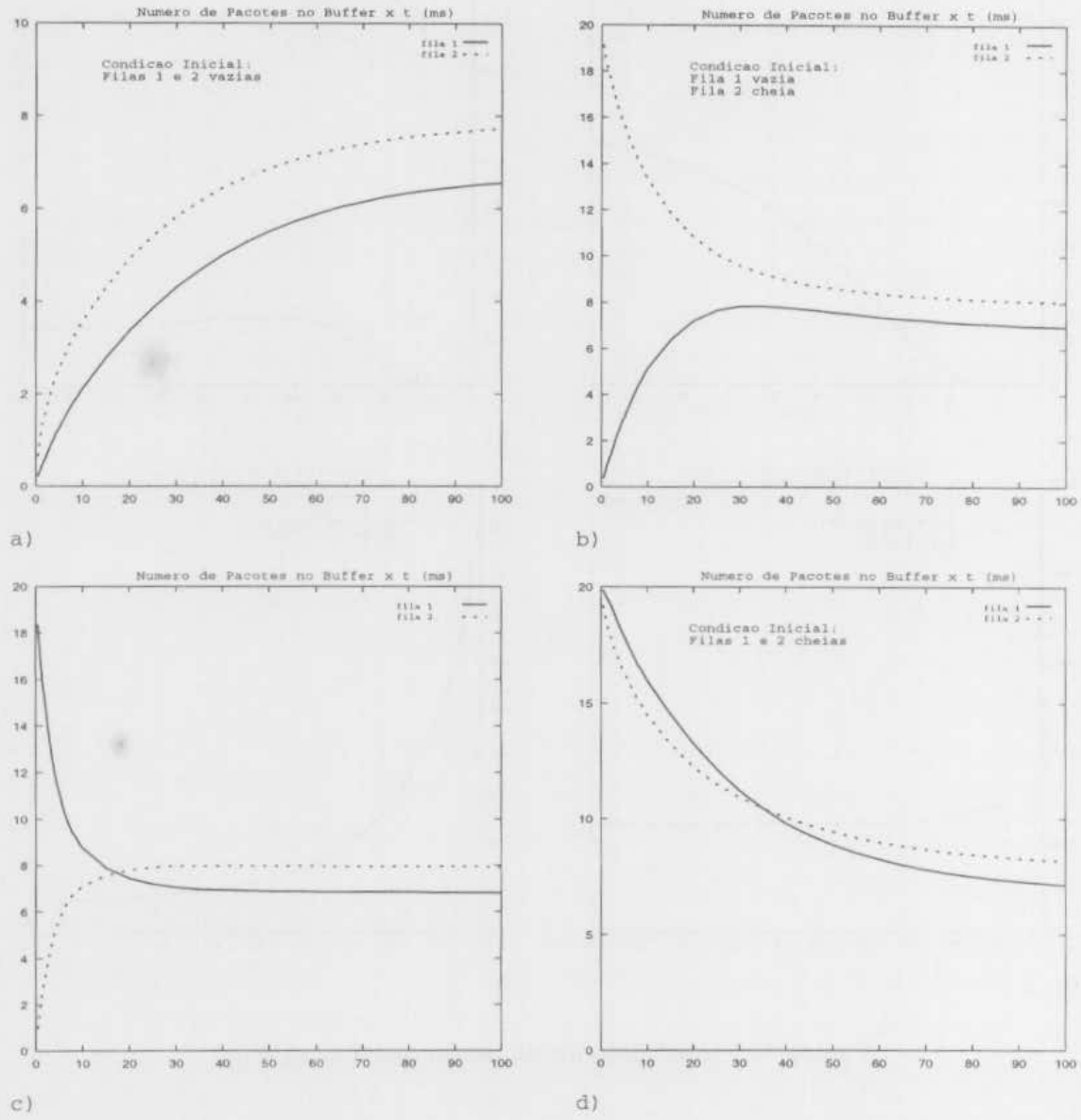


Figura 5: Número esperado de pacotes nas filas q_1 e q_2

put. O mesmo pode ser dito para a fila q_1 comparando-se os gráficos a e b.

A figura 4-c mostra um fato interessante decorrente da influência mútua entre as duas filas. Vê-se que a probabilidade de perda na primeira fila cai bruscamente nos primeiros instantes após q_2 se encontrar vazia. Esta probabilidade experimenta um leve aumento, se estabilizando em seguida. Isto se dá porque após cerca de 5ms o *buffer* de q_2 já se encontra com mais de 6 pacotes em média (conforme 5-c), diminuindo consideravelmente o *throughput* de q_1.

Por fim, analisando-se a figura 4-d, vê-se que a probabilidade de perda da fila q_2 cai bem mais rápido do que a de q_1, chegando a menos de 0,1 em menos de 20ms, enquanto naquela isto só acontece após 50ms. A figura 5-d também mostra o número esperado de pacotes na segunda fila aproximando-se mais depressa de seu valor de estado estacionário ($\sim 7,9$), enquanto o número na primeira fila cai mais lentamente em direção ao seu valor estacionário ($\sim 6,8$). Isto ocorre pois o tempo de serviço da fila q_2 é maior que o de q_1.

3.4 Comparação com a uniformização

Nesta seção, comparam-se os resultados obtidos através do método direto com os mesmos obtidos usando-se a técnica da uniformização, cujo erro pode ser fixado *a priori*, com o objetivo de computar o erro do método direto. Adicionalmente, comparamos o custo do método direto com o requerido pela uniformização para os mesmos níveis de precisão.

Devido ao grande número de iterações necessário, limitou-se a resolução do presente modelo pelo método da uniformização até $t = 2{,}5$ms. Foram feitas medidas de probabilidade de perda, para as condições iniciais da figura 4-c, em 5 instantes dentro do intervalo de 0 a 2,5 ms, com precisão de 10^{-6}. Calculou-se o erro absoluto fazendo-se o módulo da diferença entre os valores obtidos pelo método da uniformização e pelo método direto. Dividindo-se este erro pelo valor da medida em cada instante, obtém-se o erro relativo.

A figura 6 mostra a variação desses erros. Pode-se ver que tanto o erro absoluto quanto o relativo caem com o tempo. Observa-se que o erro é sempre menor que 10% do valor da probabilidade real, caindo a menos de 1% para $t \geq 2{,}5$ms. Para aumentar-se a precisão da probabilidade de perda para $t \leq 2{,}5$ms, bastaria aumentarmos o valor de R.

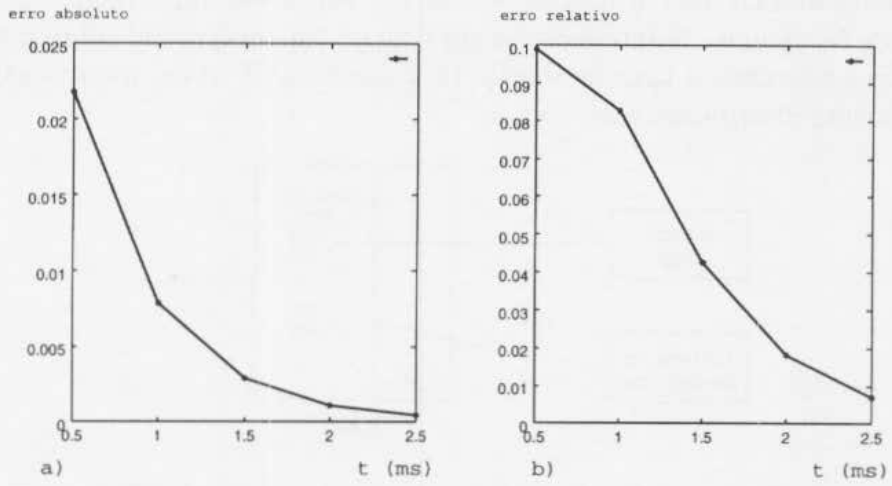


Figura 6: Erros: a) valor absoluto; b) valor relativo

Para comparar os custos do método direto e o da uniformização neste exemplo, executou-se novamente a uniformização, desta vez usando precisão equivalente a obtida pelo método direto para cada um dos instantes considerados. O custo total requerido pelo método da uniformização foi de cerca de $6{,}7 * 10^7$ multiplicações, enquanto o método direto exigiu apenas $2{,}5 * 10^7$ multiplicações, uma redução de cerca de 63%.

Estes resultados mostram a importância deste método, sem o qual seria altamente custosa a realização das medidas apresentadas. Uma vantagem prática adicional decorre do cálculo direto, baseado na equação 4, da probabilidade de um conjunto de estados e do valor esperado de uma variável aleatória sem necessidade de calcular as probabilidades de estado envolvidas.

A título de comparação de custos, foram realizadas algumas das medidas descritas na seção 3.3 usando-se o método iterativo. Foram usados os parâmetros $t = 100$, $n = 5$ e $R = 10$ para o cálculo de probabilidades de perda na fila q_1 partindo-se da condição inicial onde q_1 está cheia e q_2 vazia. O método direto teve um custo 26,6% menor que o iterativo.

4 Tráfego TCP sobre Nó ATM

Nesta seção modela-se a transmissão de um tráfego, gerado segundo o protocolo TCP, através de um nó ATM. Esta modelagem foi concebida originalmente em [8]. A estrutura de sua matriz de transição é desfavorável ao método direto, provocando o indesejável enchimento de sua matriz **A**, por isso, apenas o método iterativo é utilizado, tendo seu desempenho comparado com o do método da uniformização.

4.1 Descrição do sistema e seu modelo

Apresentaremos nessa seção uma descrição sumária do modelo, para maiores detalhes ver [8]. O sistema aqui modelado representa um comutador ATM recebendo um fluxo de células oriundo de uma fonte operando sob o protocolo TCP conforme a figura 7. Este fluxo é direcionado a um determinado canal de saída que também recebe tráfego de algumas conexões de tempo real. Este tráfego adicional em *background* tem o efeito de diminuir a banda disponível para a fonte TCP. Enquanto aguardam transmissão, as células da conexão TCP são armazenadas em um *buffer* dedicado (*buffer* TCP) com capacidade para N_b células. A interferência do tráfego em *background* sobre o tráfego TCP é modelada considerando a taxa de serviço da conexão TCP como uma variável aleatória exponencialmente distribuída com taxa μ.

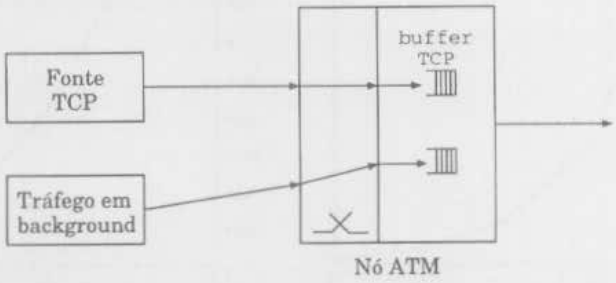


Figura 7: Fonte TCP transmitindo sobre nó ATM

Abaixo descreveremos como os mecanismos do TCP foram modelados.

Crescimento e redução da janela TCP: Inicialmente o crescimento da janela é exponencial até $W_{max}/2$ (fase *slow start*), após esse limite o crescimento é linear (fase *congestion avoidance*). No caso de perda, se a janela corrente é menor que a janela máxima (W_{max}), a janela é reduzida para 1 e o protocolo entra na fase *congestion avoidance*. Caso contrário, a janela é reduzida para 1 e o protocolo entra na fase *slow start*. O tempo decorrido entre a geração de um segmento e a chegada do seu sinal de reconhecimento (ACK), o chamado *tempo de ida e volta*, é modelado como uma variável aleatória exponencialmente distribuída de média $1/\delta$. Logo o crescimento da janela é proporcional a δ.

Expiração do time-out TCP: No modelo, considera-se o tempo de *time-out* do TCP como uma variável aleatória exponencialmente distribuída com taxa τ.

Geração de segmentos TCP: O tempo médio entre a transmissão de um segmento e o recebimento de seu ACK é $1/\delta$ e w segmentos podem ser transmitidos neste tempo. Portanto, o tempo entre a geração de dois segmentos consecutivos é modelado como uma variável aleatória exponencial de taxa $w\delta$.

Considera-se que os segmentos gerados pela fonte TCP têm tamanho fixo, sendo portanto divididos em um número constante, R_{TCP}, de células ATM. Neste modelo assume-

se que os segmentos têm todos o tamanho máximo recomendado, que corresponde a $R_{TCP} = 192$ células ATM.

Ocupação do buffer TCP: Considera-se que a chegada de células ao nó ATM ocorre em lotes de R_{TCP} células, com taxa igual a da geração de segmentos. Células serão perdidas sempre que um novo segmento é gerado e houver espaço para menos de R_{TCP} células no *buffer* TCP. Quando isto ocorre significa que a fonte não receberá o ACK correspondente àquele segmento, o que implica que após um intervalo exponencial de média $1/\tau$ a janela será reduzida voltando a crescer depois com taxa δ.

Tendo em vista estas considerações, notam-se dois parâmetros importantes para a dinâmica do modelo: $1/\delta$ e μ. O primeiro representa o atraso de ida e volta e reflete principalmente a distância entre fonte e destino, também sendo influenciado pela carga na rede e pelo tamanho do *buffer*. O segundo parâmetro, μ, modela a taxa de serviço para uma célula no nó ATM, levando em conta a taxa de transmissão e a carga imposta pelo tráfego em *background*. A matriz $\mathbf{P}$ do modelo foi gerada com o mesmo programa usado em [8].

No sistema aqui modelado, a taxa disponível para a conexão TCP no canal de saída do comutador ATM é de cerca de 200 Mbps, o que é representado no modelo por $\mu = 500$ células/ms (que corresponde precisamente a 212 Mbps). A distância entre fonte e destino é de 1600 km, causando um *atraso de ida e volta* de aproximadamente 16 ms, modelado por $\delta = 0{,}06$. Em coerência com este valor, fez-se $\tau = 0,012$. O valor ideal para o tamanho máximo da janela TCP, considerando um *throughput* máximo requerido pela aplicação de cerca de 130 Mbps é $W_{max} = 30$. O tamanho do *buffer* é de $N_b = 1000$ células, valor usual em comutadores ATM. Em consequência, o presente modelo possui $N = 68068$ estados.

4.2 Medidas realizadas

Para estudar o desempenho do protocolo TCP/IP operando sobre uma rede ATM, algumas medidas importantes são as probabilidades de perda de células e segmentos e o *throughput* alcançado pela conexão. Outras medidas interessantes são a carga oferecida pela conexão TCP à rede, o número esperado de células no *buffer*, o tamanho esperado da janela TCP e as probabilidades do sistema estar na fase *slow start* ou *congestion avoidance*. Estas medidas foram calculadas baseadas na probabilidade de um conjunto de estados e no valor esperado de uma variável aleatória do modelo conforme descrito na seção 2.2. As variáveis de estado do modelo usadas para a obtenção das medidas são b (número de células no buffer) e w (tamanho da janela). Um estado é considerado *com perda* se o número de células no *buffer* $b > N_b - R_{TCP}$, e *sem perda* em caso contrário.

- *Throughput* no receptor:

  Obtido através do cálculo do valor esperado da *taxa de chegada de segmentos úteis*, ou seja segmentos que não terão nenhuma célula perdida. Isto é feito compondo-se o vetor $\mathbf{y}$ (seção 2.2) com $w\delta$ para estados *sem perda* e 0 para estados *com perda*.

- Carga oferecida a rede:

  Calcula-se o valor esperado da *taxa de chegada de segmentos*, compondo-se o vetor $\mathbf{y}$ com o valor de $w\delta$ em cada estado.

- Probabilidade de perda de célula:

  Esta probabilidade é igual a *taxa de perda de células* dividida pela carga oferecida a rede. Consequentemente, esta medida pode ser obtida encontrando-se o valor es-

perado da *taxa de perda de células* atribuindo o valor $(w\delta.(R_{TCP} - N_b + b)/R_{TCP})$ às entradas do vetor **y** correspondentes aos estados *com perda*, e o valor 0 às demais entradas. Em seguida, divide-se este valor médio pela carga oferecida a rede, obtendo-se a probabilidade desejada.

- Probabilidade de perda de segmento:

  De forma semelhante a medida anterior, esta probabilidade será igual a *taxa de perda de segmentos* dividida pelo valor da carga oferecida a rede. Assim, para o cálculo do valor esperado da *taxa de perda de segmentos*, atribui-se o valor de $w\delta$ ao vetor **y** nas posições correspondentes aos estados *com perda* e 0 nas demais. Para se obter a probabilidade desejada, divide-se esta taxa pela carga oferecida a rede.

- Número esperado de células no *buffer*:

  Calcula-se o valor esperado da variável de estado b, inicializando-se **y** com o valor desta variável em cada estado.

- Tamanho esperado da janela:

  Calcula-se o valor esperado da variável de estado w, atribuindo-se às entradas de **y** o seu valor em cada estado.

- Probabilidade de estar na fase *slow start*:

  Calcula-se a probabilidade do sistema estar no conjunto dos estados em que a fase de crescimento da janela TCP é *slow start*.

Foram executadas duas baterias de medidas. Na primeira, foram realizadas medidas em 3 instantes de interesse para um intervalo de 0 a 15ms. Na segunda bateria, foram feitas medidas em 3 instantes num intervalo de 0 a 150ms. Foram usadas duas condições iniciais relevantes. A primeira corresponde à situação em que o *buffer* está vazio ($b = 0$), a janela está em seu valor mínimo ($w = 1$), e a sua fase de crescimento é *slow start*. Na segunda condição inicial, têm-se $b = 0$, como na primeira, porém a janela está em sua máxima amplitude ($w = 30$) e na fase de *congestion avoidance*. Todas as medidas para estado estacionário foram obtidas com o algoritmo proposto em [8].

A figura 8-a mostra a evolução do tamanho da janela TCP após a primeira condição inicial. Nela, vê-se que w tende a crescer lentamente, principalmente após atingir metade de seu valor máximo. Em $t = 150$ms seu valor ainda está por volta de $w = 21$, longe de seu valor em estado estacionário de cerca de 29,6. Na figura 8-b vê-se que o valor inicial da janela de $w = 30$ cai em direção ao valor de 29,6, em consequência das perdas de segmentos que provocam sua redução.

Observando-se a figura 9, pode-se entender melhor o crescimento da janela apresentado na figura 8-a. Pela curva correspondente à mesma condição inicial, percebe-se que *slow start* é a fase de crescimento mais provável até cerca de 70ms. Na curva em que *congestion avoidance* é a fase inicial, nota-se que a probabilidade do sistema mudar de fase é extremamente pequena nos primeiros instantes. Após decorridos 150ms, seu valor está praticamente estabilizado e vê-se que a probabilidade do sistema estar na fase *slow start* é menor que 10^{-2}.

Na figura 10-a vê-se que a carga oferecida e o *throughput* crescem lentamente após a condição inicial com $w = 1$, chegando o *throughput* a cerca de 90Mbps em $t = 150$ms, ainda longe de seu valor de estado estacionário, 128Mbps. Este comportamento é uma consequência direta da evolução do tamanho esperado da janela mostrado na figura 8-a.

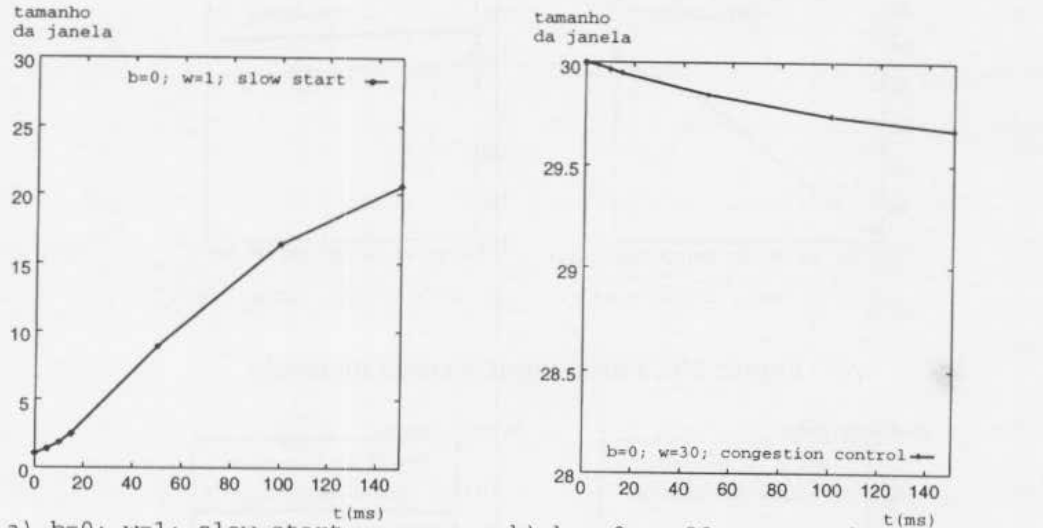


Figura 8: Tamanho esperado da janela TCP

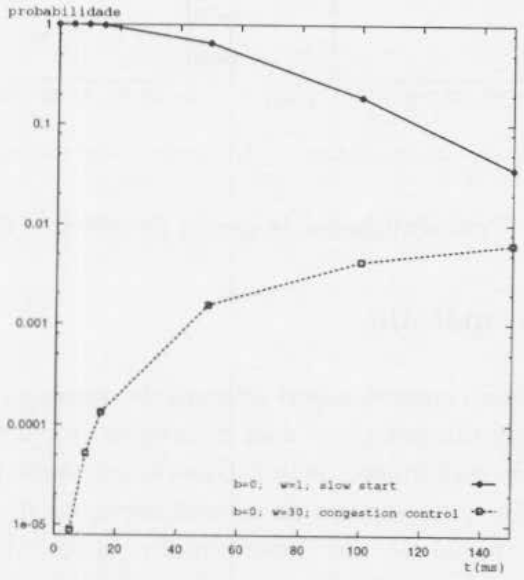


Figura 9: Probabilidade do sistema estar na fase *slow start*

O gráfico 10-b mostra essas medidas partindo da condição de janela máxima ($w = 30$). Nota-se que o *buffer* vazio garante o aproveitamento de todos os segmentos transmitidos inicialmente, proporcionando um alto *throughput* inicial, que cai gradualmente para seu valor médio final.

A figura 11 mostra os resultados das medidas de probabilidades de perda de célula e de segmento. Observa-se que estes valores são coerentes com o comportamento da carga oferecida e do *throughput* descritos acima.

A figura 12 ressalta a diferença de velocidade com que o *buffer* se enche quando a janela TCP está inicialmente com seu valor mínimo ($w = 1$) e quando está inicialmente com seu valor máximo ($w = 30$). No primeiro caso, após 150ms o *buffer* acumula cerca de apenas 91 células, ao passo que no segundo caso este número atinge seu valor estacionário de cerca de 190 células em menos de 10ms.

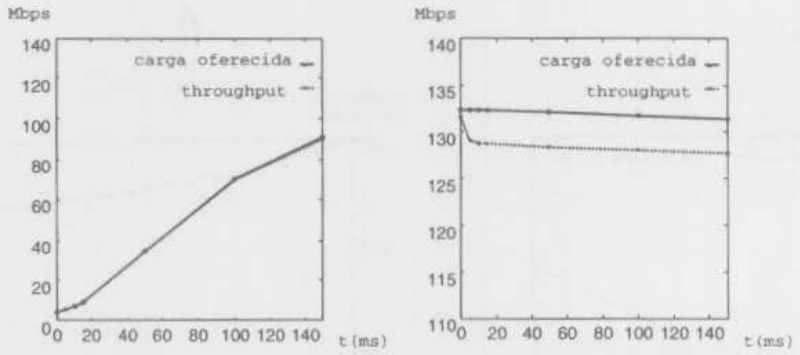


Figura 10: Throughput e carga oferecida

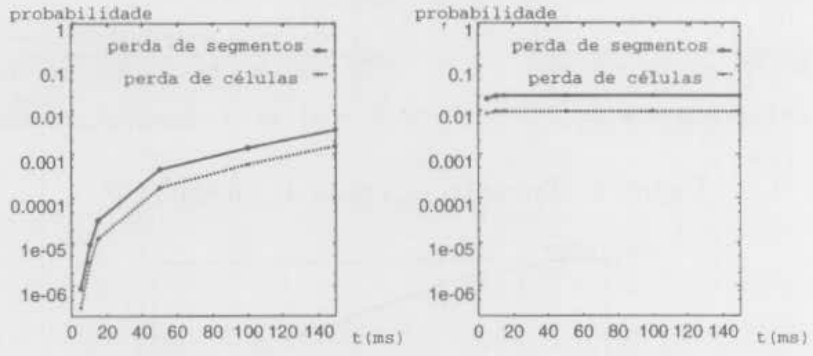


Figura 11: Probabilidades de perda de célula e de segmento

4.3 Desempenho do método

Em cada medida realizada foi computado o número de iterações necessário para a solução. A figura 13 apresenta estes números nos dois intervalos considerados. Pode-se constatar, de maneira geral, a redução do número de iterações de um passo r para outro e também que o intervalo menor ($t = 15$ms) necessitou de menos iterações do que o maior ($t = 150$ms). Isto ocorre pois o vetor inicial de um determinado passo r torna-se uma aproximação melhor do vetor final com o aumento do passo e quanto menor for o valor de t.

Para comparar os custos, o método da uniformização foi executado, calculando-se as probabilidades de estado transientes do modelo a partir de uma dada condição inicial. Devido ao excessivo número de iterações necessário sua execução foi realizada somente até $t = 50$ms. A figura 14 apresenta os custos totais necessários para as medidas em todos os instantes até t.

Os valores para o método iterativo correspondem às medidas que apresentaram os custos mínimo e máximo. Das curvas entre 5 e 15ms pode-se ver que para cada instante acrescentado à medida, o custo total do método iterativo cresce muito mais lentamente do que o da uniformização. Para os três instantes considerados, o custo total da uniformização é de 949.447.552, enquanto o do método iterativo, no pior caso, é de 253.988.724 multiplicações, ou seja, 73,2% abaixo daquele valor. Os valores relativos à medida em $t = 50$ms podem ser comparados ao custo da uniformização para este instante, constatando-se uma redução de 66% no pior caso.

Note que, devido às diferenças de ordem prática entre os métodos implementados e o

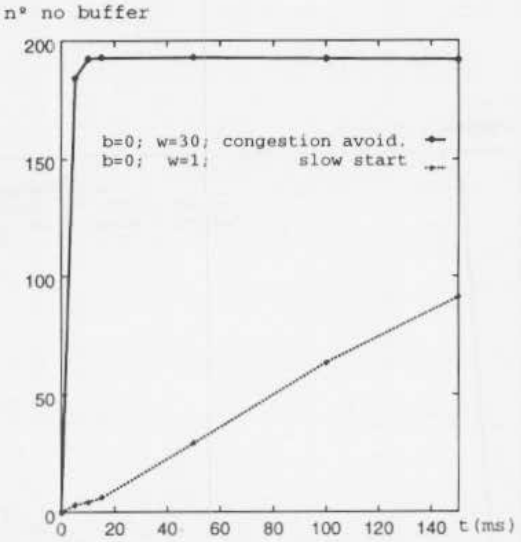


Figura 12: Número esperado de células no *buffer*

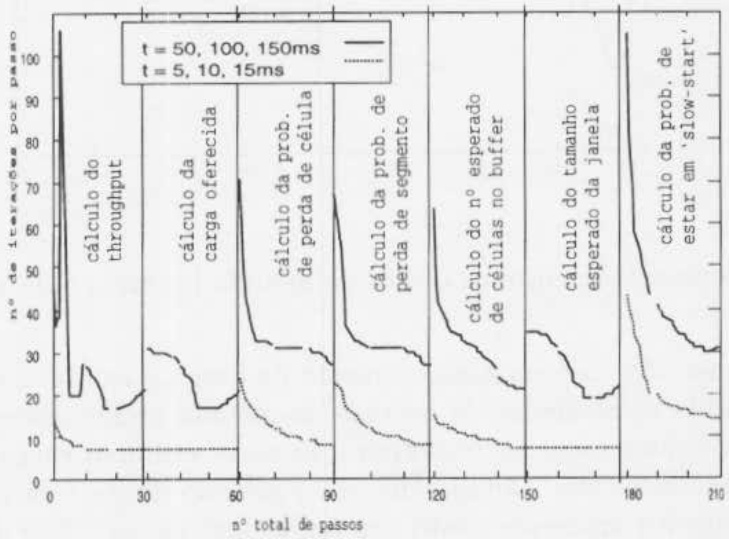


Figura 13: Número de iterações por passo em cada medida

da uniformização, este último seria executado duas vezes, uma para cada condição inicial e cada intervalo, enquanto o método iterativo foi executado sete vezes, uma para cada medida. Nestas condições, para as medidas até $t = 15$ms, o método iterativo teria um custo 41,6% inferior.

5 Conclusão

Neste artigo avaliamos o desempenho de métodos recentemente propostos para a obtenção de medidas transientes em modelos markovianos.

Dois exemplos foram apresentados. O primeiro deles, o modelo do multiplexador, permitiu avaliar duas características que levam a vantagens computacionais em relação à uniformização. Uma delas é o fato de sua matriz de transição de estados possuir uma estrutura especial, o que reduz significativamente o custo computacional do método direto. A outra característica se deve ao fato do modelo ser *stiff*. Foi observado através dos

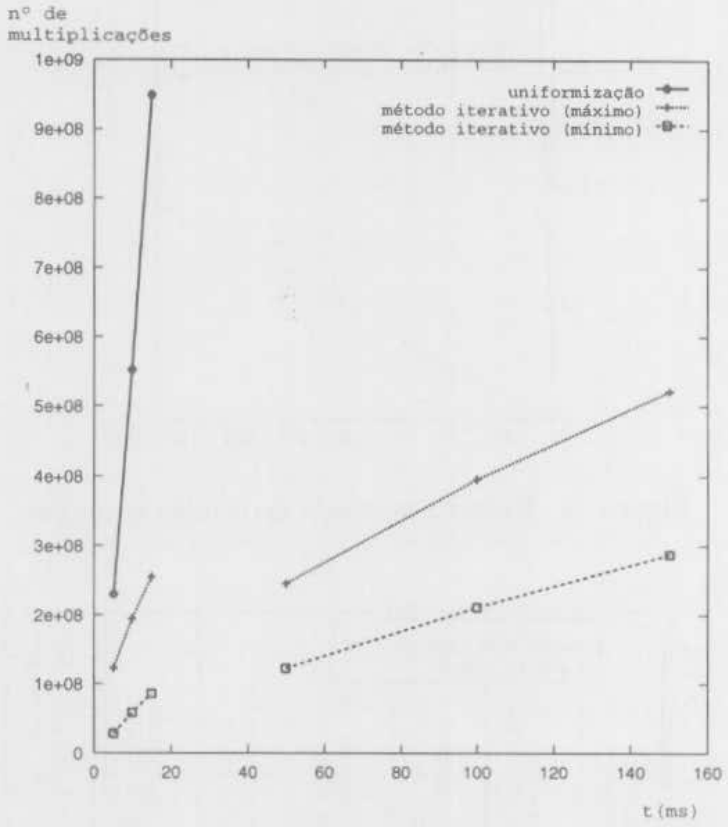


Figura 14: Comparação entre os custos do método iterativo e da uniformização

resultados que o custo da uniformização aumenta de forma quadrática com o tempo total para o qual a medida é calculada. Já no caso do método implementado esse aumento é linear. O custo computacional da uniformização neste exemplo chega a ser 63% maior que o custo do método direto. Comparando-se o método direto com o iterativo pôde-se observar que o primeiro apresenta custo computacional menor. Isto deve-se ao fato da matriz de transição apresentar uma estrutura especial que favorece o método direto.

No segundo exemplo, o modelo do tráfego TCP sobre ATM foi observado um desempenho melhor do método iterativo em relação ao método direto. Isto deve-se ao fato da matriz de transição do modelo não apresentar uma estrutura especial adequada ao método direto. Comparando-se o custo do método iterativo com o da uniformização, observa-se que podem ser obtidos ganhos computacionais de até 79%.

Nos dois exemplos, a precisão obtida com os métodos, é bastante satisfatória quando comparada à uniformização. Quanto ao custo computacional, foi observado um crescimento linear com o número de passos R (para os dois métodos) e que o custo do método direto é independente de t.

Futuramente podem ser realizados diversos trabalhos. Pretendemos estender a atual implementação para permitir a obtenção de medidas de recompensa acumulada e também buscar limites ou aproximações para o erro computacional dos métodos tratados.

O método será integrado à ferramenta de domínio público TANGRAM-II, tornando-se uma importante opção para obtenção de medidas transientes nesta ferramenta. Por outro lado, as facilidades de construção, manipulação e resolução de modelos estarão disponíveis auxiliando a realização das medidas desejadas.

Referências

[1] R.M.L.R. Carmo, L.R. de Carvalho, E. de Souza e Silva, M.C. Diniz, and R.R. Muntz. TANGRAM-II: A performability modeling environment tool. In *Lecture Notes in Computer Science 1245: Proceedings of the 9th International Conference of Performance Evaluation Modelling Techniques and Tools*, volume 1245, pages 6–18, St. Malo, França, junho 1997. Springer.

[2] Rosa M. L. R. Carmo, Edmundo de Souza e Silva, and Raymond Marie. Efficient solutions for an approximation technique for the transient analysis of markovian models. Technical report, Institut National de Recherche en Informatique et en Automatique, Centre de Diffusion, INRIA, BP 105 – 78153 Le Chesnay Cedex, France, 11 1996.

[3] Edmundo de Souza e Silva and H. Richard Gail. The uniformization method in performability analysis. Technical report, IBM Research Division, Thomas J. Watson Research Center - Yorktown Heights, NY 10598, 2 1996.

[4] Rosa M. L. R. Carmo e Edmundo de Souza e Silva. Uma solução aproximada para análise transiente de modelos markovianos. In *XVI Congresso da Sociedade Brasileira de Computação - XXIII SEMISH*, pages 333–344, Recife, agosto 1996.

[5] G. H. Golub and C. F. van Loan. *Matrix Computation.* 2nd edition. The John Hopkins University Press, 1989.

[6] W.K. Grassmann. Finding transient solutions in Markovian event systems through randomization. In W. J. Stewart, editor, *Numerical Solution of Markov Chains*, pages 357–371. Marcel Dekker, Inc., 1991.

[7] A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavsk Aktuarietidskrift*, 36:87–91, 1953.

[8] M. Meo, E. de Souza e Silva, and M. Ajmone Marsan. Efficient solution for a class of markov chain models of telecommunication systems. *Performance Evaluation*, 27-28:603–625, 1996.

[9] S. M. Ross. Approximating transition probabilities and mean occupation times in continuous-time markov chains. *Probability in the Engineering and Informational Sciences*, pages 251–264, 1987.