SIMULAÇÃO DE REDES ATM NO NÍVEL DE CÉLULAS

ANTÔNIO M. ALBERTI, ERNESTO L. ANDRADE NETO, LEONARDO DE S. MENDES

Departamento de Comunicações - Universidade Estadual de Campinas - UNICAMP
Caixa Postal 6101, CEP: 13081-970, Campinas SP – Brasil
Tel: 0xx19-7883703, Fax: 0xx19-7881395, alberti, ernestoa, lmendes@decom.fee.unicamp.br

RESUMO

Neste artigo apresentaremos um conjunto de modelos para a simulação de redes ATM no nível de células desenvolvido para o ambiente de simulação Hydragyrum, que é um simulador de redes de comunicações expansível desenvolvido em C++ para o sistema operacional *Windows* 95/98/NT™. O conjunto de modelos desenvolvido abrange o transporte de células ATM, o roteamento, estabelecimento e remoção de conexões ATM, e as funções de gerenciamento de tráfego ATM. Apresentaremos também um exemplo de aplicação do conjunto de modelos para a simulação de uma rede ATM.

1. INTRODUÇÃO

O ATM (*Asynchronous Transfer Mode*) [1][2] é uma tecnologia de transmissão, multiplexação e chaveamento de pequenos pacotes de tamanho fixo, chamados de células, que permite a integração e o transporte de voz, vídeo, imagens e dados sobre uma mesma rede. Atualmente, o ATM está sendo utilizado para o transporte e a comutação eficiente de informações em redes de grande capacidade (*backbones*). É de razoável aceitação que a tecnologia ATM oferece vantagens em potencial tanto em termos de vazão agregada de comutação quanto no suporte a qualidade de serviço (QoS – *Quality of Service*). Entretanto, as redes ATM experimentam um crescimento tremendo em termos de tamanho, complexidade e heterogeneidade. Neste cenário, projetar, analisar e avaliar o desempenho destas redes constitui um desafio para os profissionais de telecomunicações. Várias soluções podem ser utilizadas para auxiliar na execução destas tarefas, entre elas a simulação computacional, que tem sido apontada como a melhor alternativa. Assim sendo, neste artigo apresentaremos um conjunto de modelos para a simulação de redes ATM no nível de células desenvolvido para o ambiente de simulação de redes Hydragyrum [3][4].

O restante deste artigo está dividido como segue: a seção 2 apresenta o ambiente de simulação Hydragyrum; a seção 3 apresenta o conjunto de modelos desenvolvido; a seção 4 apresenta um exemplo de aplicação do conjunto de modelos para a simulação de uma rede ATM. Finalmente, a seção 5 termina o artigo com algumas considerações finais.

2. AMBIENTE DE SIMULAÇÃO

O Hydragyrum versão 1.0 [3][4] é um simulador expansível baseado na técnica de simulação *event-driven*, que foi desenvolvido em C++ para o sistema operacional *Windows* 95/98/NT™. A seguir descreveremos a estrutura e o funcionamento do Hydragyrum.

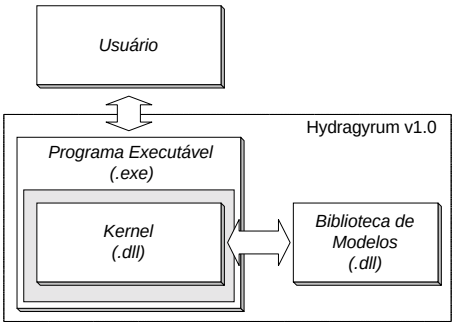

Figura 1. Estrutura do Hydragyrum.

2.1 Estrutura e Funcionamento

A estrutura do Hydragyrum pode ser dividida em duas partes principais como ilustra a Figura 1: programa executável e biblioteca de modelos. O programa executável é a parte principal do Hydragyrum. Ele possui uma instância do *kernel* (ou núcleo) do simulador. A biblioteca de modelos é o conjunto de todos os modelos que podem ser utilizados para construir uma rede a ser simulada. O Hydragyrum disponibiliza dois programas executáveis: um com interface em modo caractere e outro com interface gráfica. A Figura 2 mostra o programa executável com interface gráfica. No Hydragyrum os modelos e o *kernel* são implementados como bibliotecas de ligação dinâmica (DLLs – *Dynamic Link Libraries*) [5] do sistema operacional *Windows* 95/98/NT™.

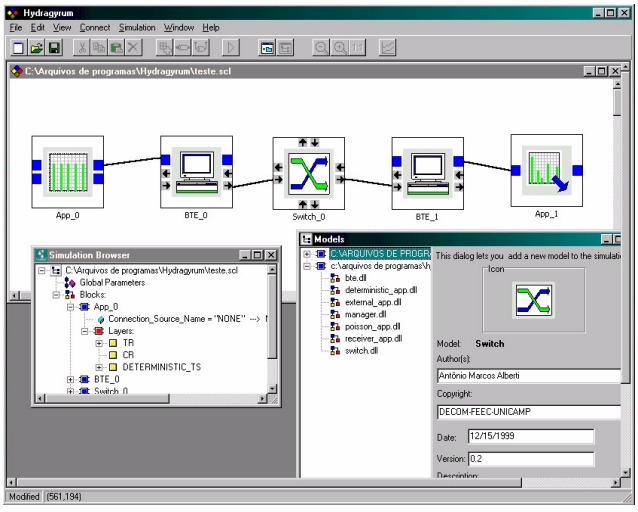

Figura 2. Programa executável com interface gráfica.

Toda a estrutura do simulador foi desenvolvida utilizando-se a linguagem de programação C++ [6], que permite usufruir das vantagens da programação orientada a objetos (OOP – *Object*

Oriented Programming), tais como: encapsulamento, herança, polimorfismo, etc.

2.1.1 Kernel

O *kernel* do Hydragyrum é responsável pelo gerenciamento do processo de simulação, pela carga e descarga de modelos e pelo suporte à interação com o usuário. O *kernel* do simulador foi desenvolvido utilizando-se a técnica de simulação baseada em eventos (*event-driven*) [7]. A Figura 3 mostra a estrutura do *kernel*, que possui vários modelos, utilizados para compor a rede a ser simulada, várias **conexões de blocos**, **conexões de rede** e **conexões de dados**, utilizadas para conectar os modelos, e vários módulos funcionais.

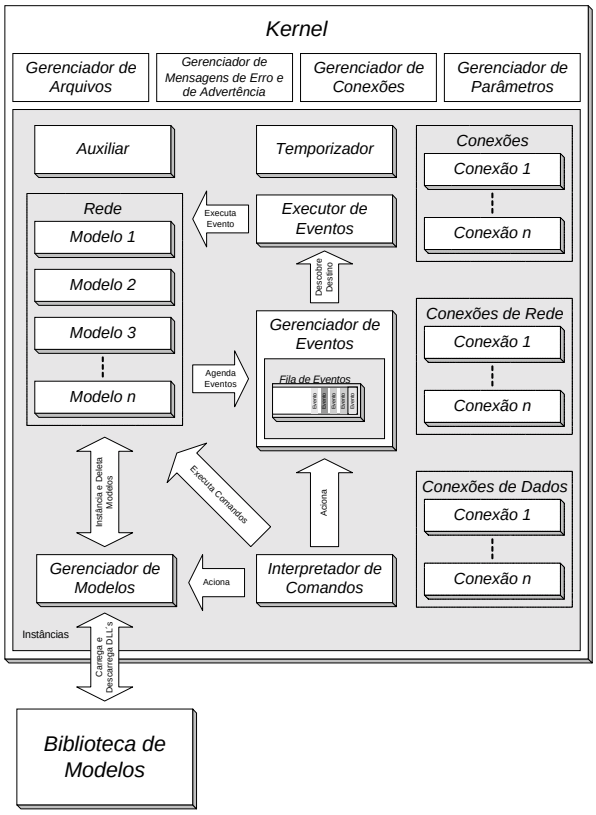

Figura 3. Estrutura do *kernel* do Hydragyrum.

Os principais módulos funcionais do *kernel* do Hydragyrum são:

Interpretador de Comandos – É responsável pela execução de comandos junto a rede a ser simulada, pelo acionamento do gerenciador de eventos e do construtor de modelos, e pela troca de comandos com o programa executável.

Gerenciador de Eventos – É responsável pelo armazenamento de eventos em uma fila com prioridades. Esta fila ordena os eventos para a execução conforme o tempo de chegada.

Executor de Eventos – Encaminha os eventos para os modelos de destino.

Gerenciador de Modelos – Realiza a carga e a descarga dos modelos (*.dll*) da biblioteca do simulador para a memória.

2.1.1.1 Conexões

Conexões são objetos utilizados para representar o relacionamento topológico entre os modelos presentes no ambiente de simulação. O Hydragyrum possui uma estrutura hierárquica de conexões. O primeiro nível desta estrutura é formado pelas **conexões de blocos**, que são utilizadas para conectar somente dois modelos (*.dll*) entre si. O segundo nível é formado pelas **conexões de rede**, que são utilizadas para estabelecer um caminho entre dois modelos (*.dll*). Este caminho é definido como um grupo de **conexões de blocos**. Finalmente, o terceiro nível hierárquico é formado pelas **conexões de dados**, que também são utilizadas para estabelecer um caminho entre dois modelos (*.dll*). Porém, neste caso, este caminho é definido como um grupo de **conexões de rede**.

2.1.1.2 Parâmetros

Parâmetros são variáveis utilizadas para configurar ou modificar o comportamento dos modelos. Os parâmetros do Hydragyrum permitem o armazenamento de valores escalares, vetoriais e matriciais.

2.1.2 Biblioteca de Modelos

Um dos principais recursos do Hydragyrum é permitir que os usuários implementem e incorporem novos modelos à biblioteca do ambiente de simulação. O Hydragyrum v1.0 possui uma estrutura hierárquica de modelos, como ilustra a Figura 4.

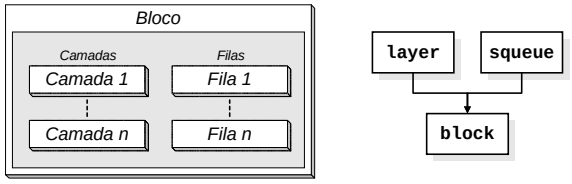

Figura 4. Estrutura hierárquica de modelos no Hydragyrum.

Esta estrutura se baseia em três classes base [6]: `Block`, `Layer` e `Squeue`. As classes `Layer` e `Squeue` constituem um primeiro nível desta estrutura hierárquica, enquanto a classe `Block` constitui um segundo nível. Portanto, no Hydragyrum os modelos são construídos a partir da derivação das classes base `Block`, `Layer` e `Squeue`. Chamaremos de **blocos**, **camadas** e **filas**, os modelos derivados destas três classes base, respectivamente. Assim sendo, os **blocos** podem ou não possuir várias **camadas** e **filas**, que chamaremos de modelos internos de um **bloco**.

3. CONJUNTO DE MODELOS NO NÍVEL DE CÉLULAS

O modelamento no nível de células foi a primeira solução encontrada para modelar redes ATM. Neste tipo de estratégia os modelos preocupam-se com o transporte, armazenamento, serviço e processamento individual de células ATM [7]. Dentro deste contexto, nós desenvolvemos um conjunto de modelos no nível de células para o ambiente Hydragyrum. Este conjunto de modelos é implementado no simulador através de **blocos**, **camadas** e **filas**. A Figura 5 mostra a estrutura do conjunto de modelos desenvolvido.

Esta estrutura é baseada em quatro **blocos**: aplicativo (*App – Application*), terminal faixa larga (BTE – *Broadband Terminal Equipment*), chaveador (SW – *Switch*) e gerente (*Manager*).

O aplicativo é o criador de conexões e a fonte de tráfego da rede ATM. Ele possui quatro **camadas**: camada criadora de conexões, camada removedora de conexões, camada fonte de tráfego e camada receptora de tráfego.

O terminal faixa larga é um dispositivo de borda da rede ATM, como por exemplo um cartão de interface com a rede (NIC – *Network Interface Card*) [2]. O BTE possui quatro **camadas**: camada ATM, camada de adaptação ATM, camada física de entrada e camada física de saída. Além das **camadas**, O BTE possui cinco **filas** que implementam modelos de estruturas de filas, escalonadores e de algoritmos de gerenciamento de tráfego ATM [8]. São modelados algoritmos de controle de admissão de conexões, algoritmos de gerenciamento de estruturas de filas e algoritmos de descarte seletivo de células ATM. Nas próximas seções descreveremos em maiores detalhes estes modelos. O chaveador é um dispositivo de comutação de células da rede ATM [1][2]. Ele possui três **camadas**: camada ATM, camada física de entrada e camada física de saída, e as mesmas **filas** do BTE**.** O gerente é responsável pelo roteamento, estabelecimento e remoção de **conexões** ATM e de **dados**. Para isto, a versão atual do modelo gerente possui um modelo de protocolo de roteamento que chamamos de Protocolo de Roteamento do Primeiro Caminho com Qualidade de Serviço. Este protocolo encontra um caminho na rede que seja capaz de atender ao requisitos de QoS desejados para um nova conexão ATM. O primeiro caminho encontrado é aceito.

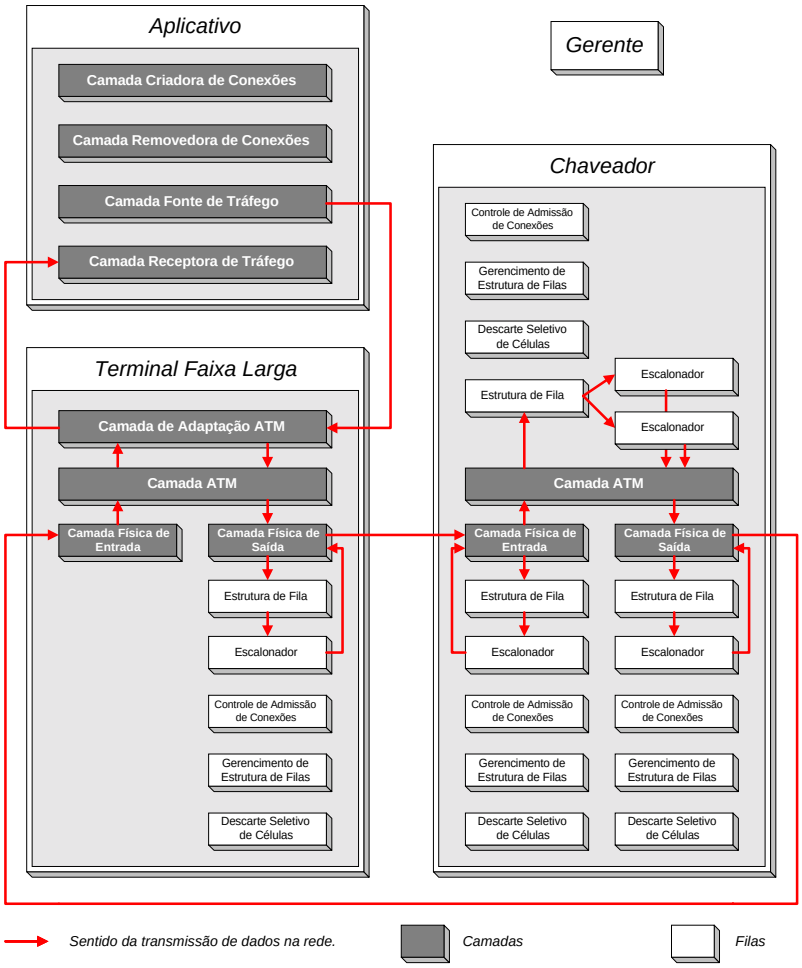

Figura 5. Estrutura do conjunto de modelos no nível de células.

3.1 Modelos de Camadas

3.1.1 Camadas Criadoras de Conexões

O conjunto de modelos no nível de células possui atualmente 5 **camadas** criadoras de conexões, cada uma delas desenvolvida com o objetivo de estabelecer conexões virtuais chaveadas ATM (SVCs – *Switched Virtual Connections*) [1] de acordo com uma determinada categoria de serviço definida pelo ATM *Forum* [10]. São elas: **Camada** Criadora de Conexões com Taxa de *Bits* Constante (CBR – *Constant Bit Rate*), **Camada** Criadora de Conexões com Taxa de *Bits* Variável em Tempo Real (RT-VBR – *Real Time Variable Bit Rate*), **Camada** Criadora de Conexões com Taxa de *Bits* Variável Não em Tempo Real (NRT-VBR – *Non-Real Time Variable Bit Rate*), **Camada** Criadora de Conexões com Taxa de *Bits* Disponível (ABR – *Available Bit Rate*) e **Camada** Criadora de Conexões com Taxa de *Bits* Não Especificada (UBR – *Unespecified Bit Rate*). Estas **camadas** possuem dois parâmetros que permitem configurar a duração do período de tempo em que as **conexões de dados** e de **rede** permanecerão ativas e o intervalo de tempo em que não serão estabelecidas conexões. São eles: ON_Period e OFF_Period. A Figura 6 ilustra os períodos ativos (ON) e inativos (OFF) das **camadas** criadoras de **conexões**, que podem ser determinísticos ou dados por uma distribuição exponencial negativa.

3.1.2 Camadas Fonte de Tráfego

As **camadas** fonte de tráfego são responsáveis pela criação dos pacotes que trafegam na rede. Elas permitem definir o instante de tempo em que cada pacote será transmitido, o tamanho de cada pacote, a **conexão de dados** e a **conexão de rede** a que o pacote pertence. O conjunto de modelos no nível de células possui atualmente 3 **camadas** fontes de tráfego: **Camada** Fonte de Tráfego Determinístico, **Camada** Fonte de Tráfego Poissoniano e **Camada** Fonte de Tráfego Externo. A fonte determinística gera pacotes de tamanho fixo com intervalos também fixos entre pacotes. A fonte poissoniana gera pacotes de tamanho dado pela distribuição de *poisson* com intervalos entre pacotes dados pela distribuição exponencial negativa. A fonte externa gera um padrão de tráfego carregado a partir de um arquivo externo ao Hydragyrum.

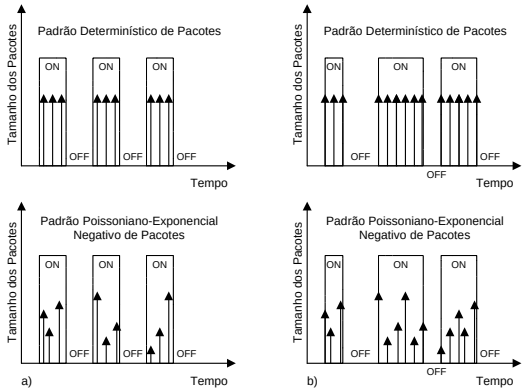

Figura 6. Padrão de tráfego das fontes determinística e poissoniana para duas situações: a) Conexões com durações determinísticas. b) Conexões com durações exponenciais negativas.

3.1.3 Camadas de Adaptação ATM, ATM e Físicas

O conjunto de modelos no nível de células possui somente um modelo de camada de adaptação ATM: a **camada** de adaptação ATM tipo 5. Escolhemos modelar esta camada devido a sua popularidade. A AAL5 é responsável pela segmentação de pacotes das **camadas** superiores em células ATM no ponto de ingresso à rede e pela remontagem destes pacotes a partir das células ATM no ponto de egresso da rede.

Foram desenvolvidos ainda dois modelos para a camada ATM: **camada** ATM do **bloco** BTE (BTE_ATM) e **camada** ATM do **bloco** chaveador (SW_ATM). A razão para isto está na considerável diferença funcional entre estas **camadas**. A BTE_ATM envia as células recebidas da AAL_5 para a **camada** física, ou vice-versa. A SW_ATM é responsável pela comutação das células ATM de um enlace de entrada para um enlace de saída. Ela implementa um modelo de uma *switch fabric* (SF) com memória compartilhada [8].

O conjunto de modelos no nível de células possui ainda dois modelos para a camada física: a **camada** física de entrada (PHY_IN) e a **camada** física de saída (PHY_OUT). A PHY_IN é responsável pelo envio das células para a **camada** ATM na entrada de um modelo de equipamento de rede. Se houver necessidade de armazenamento, a PHY_IN envia estas células para os modelos de estrutura de filas e de escalonador adequados. A PHY_OUT é responsável pelo envio das células ATM para a PHY_IN do próximo equipamento da rede.

3.2 Modelos de Estruturas de Filas

Uma rede ATM prove suporte para uma grande variedade de serviços com diferentes requisitos de QoS [1][2]. Estes serviços compartilham os recursos da rede (largura de banda, armazenamento, etc.) e tentam utilizá-los simultaneamente. Devido ao compartilhamento de recursos, pontos de congestionamento poderão aparecer na rede, causando a necessidade do uso de estruturas de filas [8] (QS – *Queueing Structures*) para armazenar temporariamente células ATM. Estas estruturas organizam filas lógicas que são atendidas por escalonadores [8]. O conjunto de modelos no nível de células possui atualmente 2 modelos de QS: estrutura de filas *default*, que mantém uma fila FIFO – *First In First Out* de células ATM, e estrutura de filas com filas por conexão (*Per*-VC *Queueing*), que mantém várias filas FIFO, uma para cada **conexão de rede** ATM. Nestas filas as células aguardam pela transmissão em um enlace entre equipamentos ou interno a uma *switch fabric*.

3.3 Modelos de Escalonadores

Escalonadores [8] (*schedulers*) são implementados em cada estrutura de filas para selecionar a ordem apropriada do serviço das células ATM, a fim de garantir os objetivos de QoS negociados. Dois modelos de escalonadores foram desenvolvidos até o momento. Ambos mantém uma fila FIFO para determinar a ordem de transmissão das células ATM. São eles: escalonador *default* e escalonador de pacotes com compartilhamento de processador generalizado (PGPS – *Packet Generalized Processor Sharing*) [9]. No escalonador *default* as células são armazenadas de acordo com a ordem de chegada, enquanto no escalonador PGPS as células são armazenadas de acordo com um tempo virtual de final de serviço. O modelo de escalonador PGPS foi desenvolvido a partir do trabalho de *Parekh* [9].

3.4 Modelos de Algoritmos de Gerenciamento de Tráfego

3.4.1 Algoritmos de Controle de Admissão de Conexões

Para determinar se uma conexão ATM pode ou não ser estabelecida em uma rede, é necessário que um algoritmo de controle de admissão de conexões (CAC – *Connection Admission Control*) [8][10] seja consultado, de forma a verificar se os requisitos de QoS para esta conexão podem ser aceitos sem que o QoS das conexões existes seja deteriorado. O conjunto de modelos no nível de células possui atualmente 2 algoritmos de controle de admissão de conexões ATM. São eles: Algoritmo de CAC *Default* e Algoritmo de CAC *Elwalid Mitra Wentworth* [11]. O *default* CAC aceita uma nova conexão se a largura de faixa disponível no escalonador for suficiente, independente do recursos disponíveis na estrutura de filas. O *Elwalid Mitra Wentworth* CAC aceita uma nova conexão baseado em cálculos de banda efetiva e *buffer* efetivo necessários para cada conexão.

3.4.2 Algoritmos de Gerenciamento de Estrutura de Filas

Algoritmos de gerenciamento de estruturas de filas (GEF) [8][10] devem ser implementados em cada estrutura de filas a fim de julgar se uma célula recebida pode ou não ser armazenada em uma estrutura que enfrenta congestionamento. O conjunto de modelos no nível de células possui atualmente 2 algoritmos de GEF. São eles: Algoritmo de GEF *Default* e Algoritmo de GEF com Particionamento Dinâmico. O modelo *default* armazena uma célula somente se o número total de células na QS somado de uma célula for menor ou igual a capacidade da QS. O modelo com particionamento dinâmico foi desenvolvido a partir do trabalho de *Krishnan*, *Choundhury* e *Chiussi* [12]. Este modelo decide se uma célula pode ser armazenada ou não na estrutura de filas de acordo com um limiar calculado para cada fila da QS.

3.4.3 Algoritmos de Descarte Seletivo de Células

Numa situação de congestionamento células ATM eventualmente terão que ser descartadas. Nesta situação, algoritmos de descarte seletivo de células [8][10] são necessários, pois células de menor importância devem ser

descartadas em beneficio de células mais prioritárias. Somente um algoritmo de descarte seletivo de células foi desenvolvido até o momento, o Algoritmo de Descarte Seletivo de Células *Default*. Este modelo descarta a célula recém recebida se o algoritmo de GEF indicar que não há espaço para esta célula na QS.

4. SIMULAÇÃO DE UMA REDE ATM

Nesta seção, apresentaremos um exemplo da aplicação do conjunto de modelos desenvolvido e do Hydragyrum na simulação de uma rede ATM. A Figura 7 mostra a topologia da rede simulada. Esta rede possui quatro aplicativos externos, que carregam respectivamente quatro arquivos com padrões de tráfego auto-similares (*self-similars*) [13]. Estes padrões possuem parâmetros de *hurst* 0.5, 0.6, 0.7 e 0.8. Os aplicativos App_0, App_2 e App_3 tem como aplicativo de destino o App_4, enquanto o aplicativo App_1 tem como aplicativo de destino o App_5. Todos os modelos de estruturas de filas, escalonadores e algoritmos de gerenciamento de tráfego utilizados são do tipo *default*. Todos os aplicativos tem períodos ON e OFF de 1 ms e as conexões criadas são da categoria de serviço CBR. Todas as taxas de enlaces e internas a *switch fabric* são de 149.76 Mbits/seg. As estruturas de filas do BTE_0 e BTE_1 possuem capacidade para 50 células, enquanto as estruturas de fila do chaveador possuem capacidade para 100 células. A distância entre os equipamentos é de 10 metros. A simulação realizada teve uma duração de 11 ms.

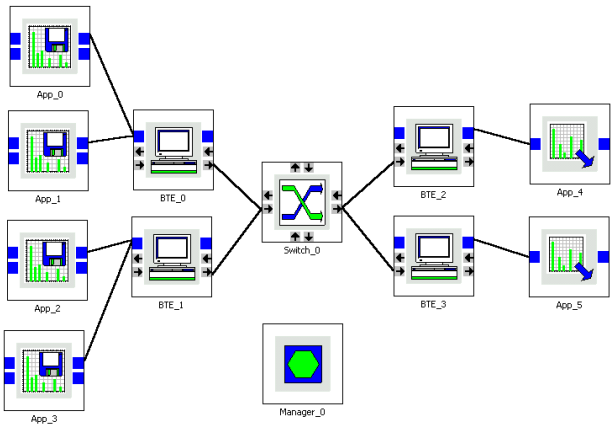

Figura 7. Configuração da rede ATM simulada.

4.1 Resultados

Agrupamos os resultados da simulação conforme os aplicativos fonte, as estruturas de filas e os escalonadores dos BTE_0, BTE_1 e Switch_0, os equipamentos de egresso (BTE_2 e BTE_3) e os aplicativos de destino. Para melhor entendermos as legendas dos gráficos que seguem, mostramos na Tabela 1 um resumo do significado das variáveis estatísticas apresentadas.

4.1.1 Aplicativos Fonte

A Figura 8 mostra o estado das **camadas** criadoras de conexões dos aplicativos App_0, App_1, App_2 e App_3. Quando o estado for igual a 1 significa que existe uma conexão estabelecida entre os aplicativos fonte e destino. A Figura 9 mostra o número de requisições de conexões aceitas e bloqueadas na rede. Pode-se observar que não houveram conexões bloqueadas, ou seja a probabilidade de bloqueio de conexões (CBP – *Connection Blocking Probability*) é igual a zero.

Variável	Significado
S	Estado da camada criadora de conexões.
M_S	Média do estado da camada criadora de conexões.
NoAC	Número de conexões aceitas.
NoBC	Número de conexões bloqueadas.
NoAC	Número de células aceitas.
NoDC	Número de células perdidas.
NoC	Número total de células recebidas.
CLR	Taxa de perda de células.
M_CLR	Taxa média de perda de células.
Qt	Número de células na estrutura de filas.
M_Qt	Número médio de células na estrutura de filas.
Dt	Atraso das células na estrutura de fila.
M_Dt	Atraso médio das células na estrutura de filas.
et	Banda efetiva total alocada pelo modelo de CAC.
M_et	Média da banda efetiva total alocada pelo modelo de CAC.
SR	Banda total usada.
M_SR	Média da banda total usada.
NoSP	Número de pacotes recebidos com sucesso.
NoDP	Número de pacotes perdidos.
NoP	Número total de pacotes.
PD	Atraso dos pacotes na rede.
M_PD	Média do atraso dos pacotes na rede.

Tabela 1 – Significado das abreviaturas de variáveis estatísticas.

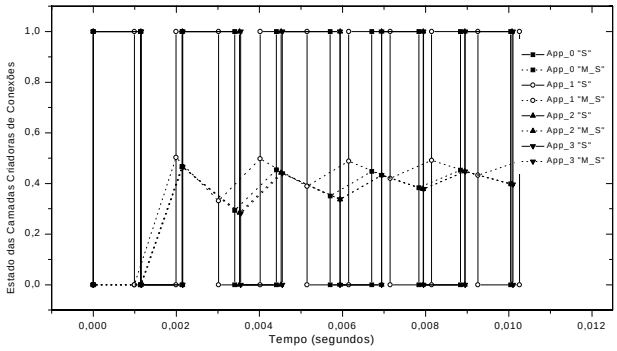

Figura 8. Estado das camadas criadoras de conexões.

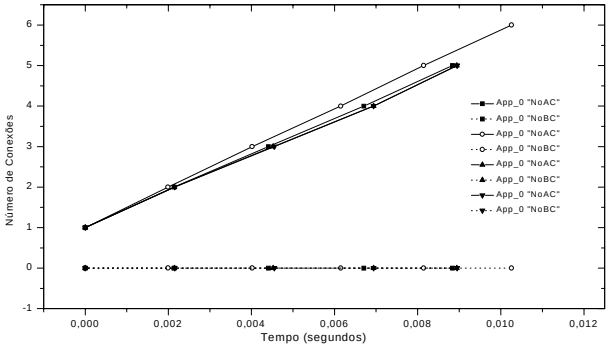

Figura 9. Número de conexões aceitas e bloqueadas na rede.

4.1.2 Estruturas de Filas

A Figura 10 mostra que algumas células ATM foram perdidas na entrada das QSs dos equipamentos da rede[1]. Isto ocorreu porque a capacidade destas estruturas (50 e 100 células) foi ultrapassada, como mostra a Figura 11. A Figura 12 mostra a taxa de perdas de células (CLR) na entrada destas estrutura de filas. A Figura 13 mostra o atraso das células nas estruturas de filas enquanto elas aguardam para serem transmitidas. Pode-se observar que os maiores valores ocorrem na estrutura de filas da *switch fabric* do Switch_0.

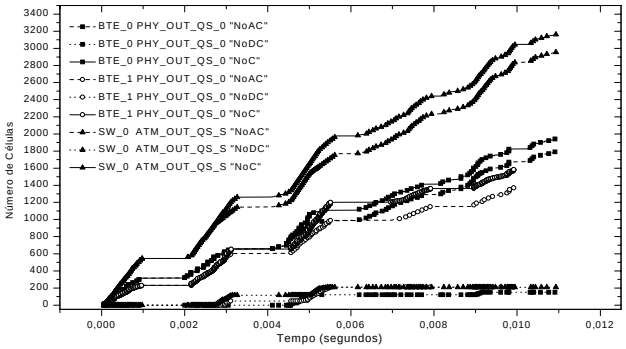

Figura 10. Número de células ATM aceitas, perdidas e total na entrada das estruturas de filas dos equipamentos da rede.

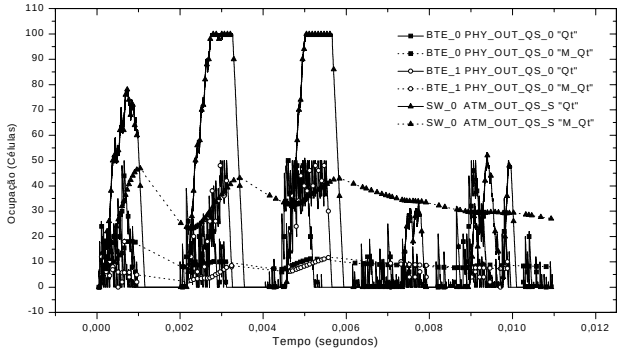

Figura 11. Ocupação total das estruturas de filas dos equipamentos da rede.

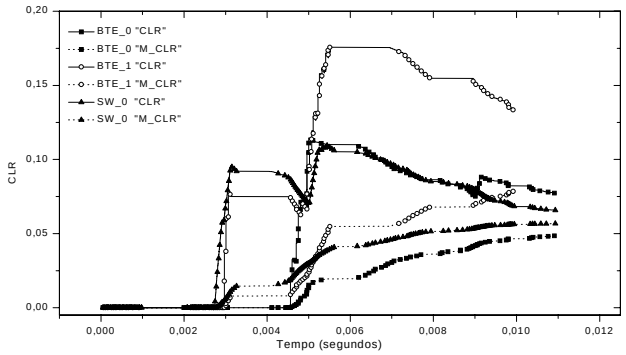

Figura 12. Taxa de perda de células (CLR) nas estruturas de filas dos equipamentos da rede.

4.1.3 Escalonadores

A Figura 14 mostra a banda efetiva média alocada para as conexões pelo CAC nos escalonadores da rede. A Figura 15 mostra a utilização média destes escalonadores. Note que existem diferenças entre a banda efetiva alocada para as conexões e a largura de faixa em células/seg. submetida à rede. É possível portanto, que os parâmetros descritores de tráfego passados para o CAC não condizam com a realidade.

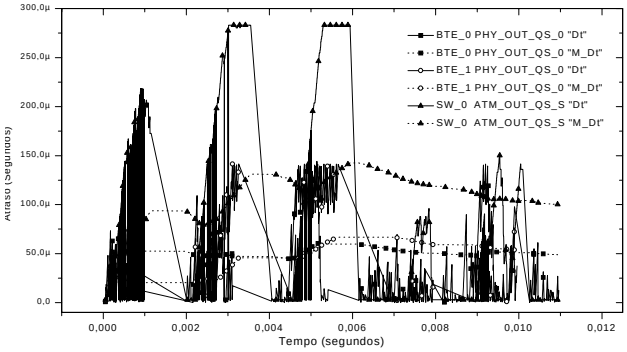

Figura 13. Atraso das células ATM nas estruturas de filas dos equipamentos da rede.

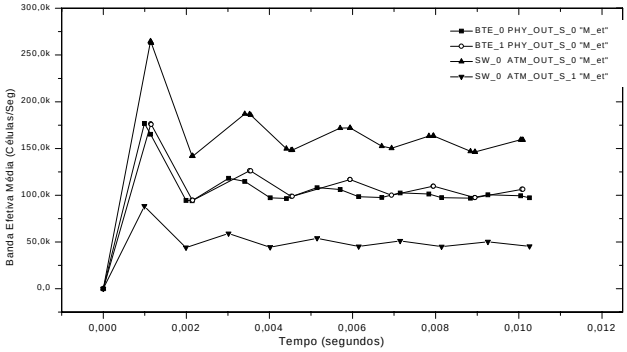

Figura 14. Banda efetiva média alocada para as conexões nos escalonadores dos equipamentos da rede.

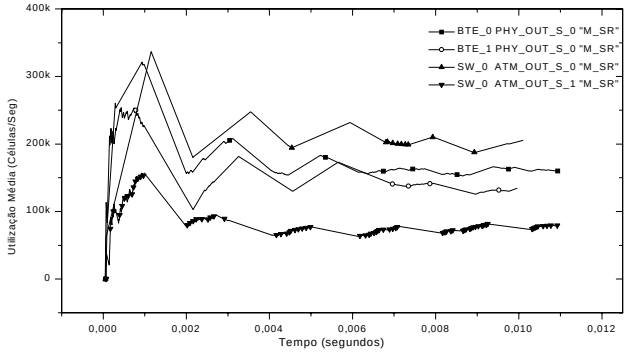

Figura 15. Utilização média dos escalonadores dos equipamentos da rede.

[1] A PHY_OUT_QS_0 é uma estrutura de filas associada a camada PHY_OUT, enquanto a ATM_OUT_QS_S é um QS compartilhada por todos as FOLs da *Switch Fabric*.

4.1.4 Equipamentos de Egresso

A Figura 16 mostra que houveram pacotes inutilizados na AAL dos BTEs 2 e 3 devido a perda de células. A Figura 17 mostra a taxa de perda de pacotes (PLR).

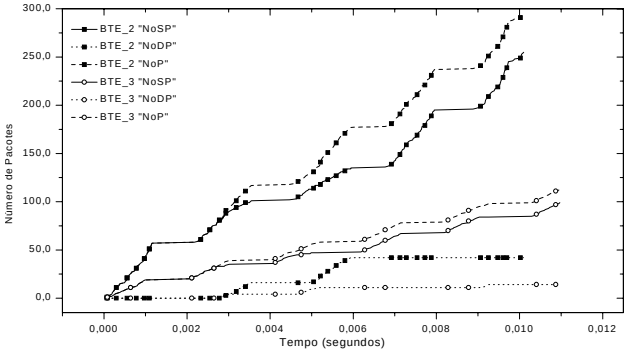

Figura 16. Número de pacotes recebidos com sucesso, perdidos e total na AAL dos BTEs 2 e 3.

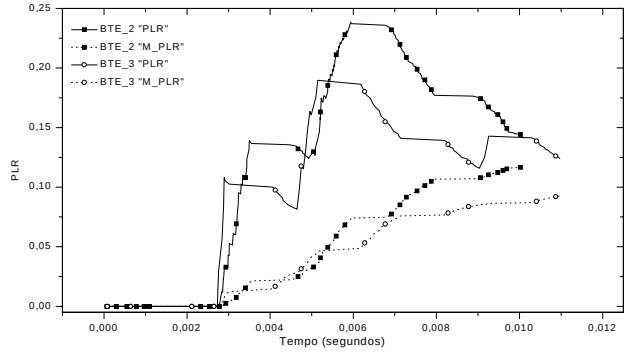

Figura 17. Taxa de perda de pacotes (PLR) na AAL dos BTEs 2 e 3.

4.1.5 Aplicativos de Destino

A Figura 18 mostra que o maior tempo de transmissão de pacotes na rede foi de aproximadamente 0.85 ms.

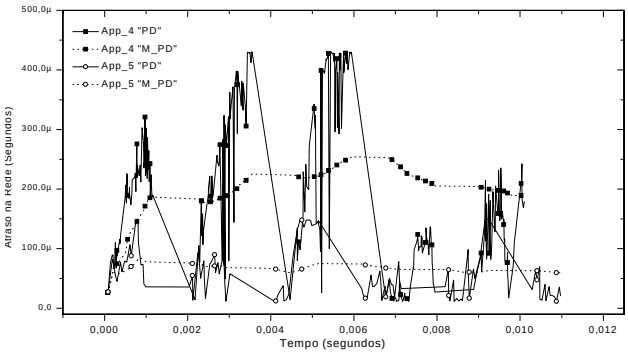

Figura 18. Tempo de transmissão dos pacotes na rede.

5. Considerações Finais

Neste artigo apresentamos um conjunto de modelos para a simulação de redes ATM no nível de células desenvolvido para o Hydragyrum, que é um ambiente de simulação de redes de comunicações expansível desenvolvido em C++ para o sistema operacional *Windows* 95/98/NT™. Vimos que o conjunto de modelos desenvolvido abrange o transporte de células ATM, o roteamento, estabelecimento e remoção de conexões ATM, e as funções de gerenciamento de tráfego ATM. A flexibilidade, potencialidade e os recursos do conjunto de modelos desenvolvido foram demostrados a partir de um exemplo de simulação de uma rede ATM sujeita a padrões de tráfego auto-similares e com recursos de largura de faixa e espaço em *buffer* limitados.

6. Agradecimentos

Gostaríamos de agradecer à FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo).

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] U. Black, "ATM: *Foundation for Broadband Networks*", *Prentice-Hall*, 1995.

[2] G. Sackett e C. Metz, "ATM *and Multiprotocol Networking*", *McGraw Hill*, 1997.

[3] Ernesto Andrade Neto, "*Hydragyrum Network Simulation Environment Programming Manual* 1.0", 1999.

[4] http://www.mc21.fee.unicamp.br/hydragyrum

[5] C. Petzol, "Programando para Windows 95", *Markron Books* e *Microsoft Press*, 1996.

[6] B. Stroustrup, "*The C++ Programming Language*", *Third Edition*, *Addison-Wesley*, 1997.

[7] Antônio M. Alberti, "Simulação de Redes ATM", Anais do XVII – Simpósio Brasileiro de Telecomunicações, 1999.

[8] N. Giroux e S. Ganti, "*Quality of Service in* ATM *Networks*: *State-of-Art Traffic Management*", *Prentice Hall*, 1998.

[9] A. Parekh, "*A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*: *The Single-Node Case*", IEEE/ACM *Transactions on Networking*, Vol. 1, No 3, Junho de 1993.

[10] ATM *Forum*, "TM 4.0 – *Traffic Management Specification*", 1996.

[11] A. Elwalid, D. Mitra, R. Wentworth, "*A New Approach for Allocating Buffers and Bandwidth to Heterogeneous, Regulated Traffic in an* ATM *Node*", IEEE *Journal on Selected Areas in Communications*, 13(6), 1995.

[12] S. Krishnan, A. Choudhury, F. Chiussi, "*Dynamic Partitioning*: *A Mecanism for Shared Memory Management*", IEEE INFOCOM'99, 1999.

[13] Z. Sahinoglu e S. Tekinay, "*On Multimedia Networks*: *Self-Similar Traffic and Network Performance*", IEEE Communications Magazine, Janeiro de 1999.