UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES
DECOM – FEEC – UNICAMP

SimATM: Um Ambiente para a Simulação de Redes ATM

Por:
Eng. Antônio Marcos Alberti

Orientador:
Prof. Dr. Leonardo de Souza Mendes

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas como parte dos pré-requisitos necessários a obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração: Eletrônica e Comunicações

Campinas – SP – Brasil
1998

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

AL14s

Alberti, Antônio Marcos
SimATM: um ambiente para a simulação de redes ATM. / Antônio Marcos Alberti.--Campinas, SP: [s.n.], 1998.

Orientador: Leonardo de Souza Mendes
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Rede digital de serviços integrados. 2. Telecomunicações. 3. Simulação (Computadores digitais). I. Mendes, Leonardo de Souza. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

A minha família, que sempre me apoiou em todas as minhas iniciativas.

"Quem se contenta com o possível nunca constrói o que quer, pois fica preso nos limites da mediocridade"

Herbert de Souza, o Betinho

Agradecimentos

Em primeiro lugar, agradeço a minha família. Aos meus queridos pais, Luiz e Leda, pelo apoio, otimismo, carinho, dedicação e lição de vida. Aos meus queridos irmãos, Fernando e Ricardo, pelo apoio, carinho, amizade e compreensão. Que vocês alcancem conquistas muito maiores que esta. Aos meus tios e tias, primos e primas, pelo carinho e amizade. Ao meu primo, Evandro, pelo incentivo.

A Deus, criador do universo, pelo dom da vida.

Em memória do meu tio Alberto Alberti, por sua lição de vida, simplicidade e sabedoria. Em memória dos meus avós Ângelo e Amabile Alberti, e Armindo e Leonida Goltz.

Agradeço aos meus amigos Gean, Fabrícia, Marcelo e Juliana, que tem sido a minha família nestes dois últimos anos. Obrigado pelos bons momentos de convivência e amizade.

Aos amigos, Jackson, Tina, Ernesto Andrade, Marcos, Marta, Natanael, Sandro, Maurício, Carla, Ernesto Barrientos, Luis Ono, Miguel Barrientos, Raul.

Aos colegas Jackson, Ernesto Andrade, Marcelo, Maurício, Ernesto Barrientos, Fernando, Gustavo, Antônio, pelas contribuições valiosas que tornaram possível este trabalho.

Ao pessoal do DECOM, DT, DMO e MULTICOM21, onde encontrei bons amigos.

A todo pessoal do LACESM, ao Everton, e especialmente ao Nelson, pelas oportunidades e lições valiosas que me ajudaram a crescer como ser humano.

Agradeço ao meu orientador Leonardo Mendes, pela oportunidade, confiança e orientação deste trabalho.

Agradeço ao professor Maurício Ferreira Magalhães e a professora Tereza Cristina Melo de Brito Carvalho, por aceitarem participar da banca julgadora.

Ao CNPq, pela bolsa de mestrado.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Esta tese descreve o **SimATM**, um ambiente para a simulação de redes ATM. O **SimATM** é um aplicativo para a pesquisa, análise e projeto de redes ATM desenvolvido em C++ para o sistema operacional *Windows* 95/NT™. O **SimATM** utiliza uma técnica de simulação baseada em eventos para realizar a simulação ATM no nível de células. A arquitetura das redes ATM é modelada a partir do Modelo de Referência de Protocolos da B-ISDN e da Arquitetura Funcional Geral de um Elemento de Rede ATM. Quatro tipos de modelos foram desenvolvidos para o **SimATM**: modelos de camadas, modelos de sistemas de fila, modelos de equipamentos ATM e modelos de aplicativos ATM. Nesta tese é feita uma descrição dos elementos da estrutura do simulador e dos modelos desenvolvidos. Detalhes da implementação em C++ destes elementos e modelos também são apresentados. Um exemplo de simulação de rede ATM é realizado, e os resultados obtidos são comprovados através da lei de *Little*. Finalmente, são apresentadas algumas conclusões e sugestões para trabalhos futuros.

Abstract

This thesis describes **SimATM**, *an environment to* ATM *network simulation. The* **SimATM** *is an application tailored to research, analysis and design of ATM networks developed in* C++ for the *Windows* 95/NT™ *operating system. The* **SimATM** *utilizes an event-driven simulation technique to achieve ATM simulation at cell level. The* ATM *networks architecture is modeled from the* B-ISDN *Protocol Reference Model and from the General Functional Architecture of an* ATM *Network Element. Four kinds of models were developed for* **SimATM**: *layer models, queueing system models,* ATM *equipment models and* ATM *application models. A description of simulator structure elements and developed models is provided in this thesis. Implementation details of these elements and models are also presented. An example of ATM Network simulation is realized, and the obtained results are confirmed by Little's Law. Finally, some conclusions and suggestions for future works are presented.*

Capítulo 1

Introdução

Asynchronous Transfer Mode, abreviado ATM, é uma tecnologia orientada a conexão, usada para transportar pequenos pacotes de tamanho fixo, chamados de células, através de uma rede de alta velocidade. Isto possibilita a integração e o transporte de voz, vídeo, imagens e dados sobre uma mesma rede, bem como o suporte a diferentes garantias de qualidade de serviço [1][2][3] para cada tipo de tráfego. A tecnologia ATM permite ainda o uso sob demanda dos recursos da rede e pode ser utilizada tanto em redes locais (LANs – *Local Area Networks*) como em redes de longa distância (WANs – *Wide Area Networks*).

Baseado nas suas inúmeras qualidades, o ATM foi escolhido pelo ITU-T – *International Telecommunications Union – Telecommunications Standards Sector* [4][5] como a tecnologia básica de transporte para a futura Rede Digital de Serviços Integrados Faixa Larga (B-ISDN – *Broadband Integrated Service Digital Network*) e, embora tenha sido originalmente projetado para redes públicas, também está assumindo uma posição central na evolução das redes privadas. Em parte, isto se deve aos contínuos esforços feitos pelo grupo ATM *Forum* [6][7] em desenvolver um conjunto de especificações e acordos de interoperabilidade que, para redes privadas, habilita o desenvolvimento de redes interconectadas baseadas na tecnologia ATM.

Entretanto, o ATM tem se demonstrado uma tecnologia complexa e ainda em desenvolvimento. Enquanto a comutação de células possibilita a construção em *hardware* de comutadores com alta velocidade e performance, permitindo assim altas taxas de transmissão, a implantação de redes que utilizam a tecnologia ATM requer uma infra-estrutura sobreposta e altamente complexa de protocolos [8]. E mais, devido a tecnologia ATM ser orientada à conexão e oferecer garantias de qualidade de serviço, protocolos específicos de sinalização e roteamento são necessários, bem como um amplo e sofisticado conjunto de novos mecanismos e funções de gerenciamento de tráfego, a fim de estabelecer, controlar e manter tais garantias de qualidade de serviço (QoS – *Quality of Service*).

O desenvolvimento e a avaliação de desempenho destes novos protocolos e mecanismos necessários à evolução da tecnologia ATM, bem como o projeto, a análise e a

implementação dessas redes, são tarefas complexas e trabalhosas. Várias soluções podem ser utilizadas para auxiliar na execução destas tarefas: experimentação com redes ATM reais, experimentação em *testbeds* ATM, utilização de técnicas preditivas, tais como análise matemática e simulação, etc. Em geral, algumas dessas soluções possuem custos muito elevados. Outras se aplicam a situações restritas. Entretanto, o uso de ferramentas de simulação constitui uma solução bastante flexível e relativamente barata. Dentro deste contexto, este trabalho discute o projeto e a implementação de uma ferramenta para a simulação de redes ATM, chamada **SimATM**.

O **SimATM** pretende atuar como uma ferramenta de aprendizado, pesquisa, análise e projeto de redes ATM, bem como uma ferramenta de validação de modelos e de análise de estratégias de interconexão de redes através da tecnologia ATM [8].

Como uma ferramenta de aprendizado da tecnologia ATM, o **SimATM** visa proporcionar um ambiente para a familiarização com as técnicas de multiplexação e comutação de células ATM, bem como o entendimento de protocolos que operam nas camadas física, ATM e de adaptação ATM do modelo de referência da B-ISDN [9], abrangendo funções relacionadas com a transmissão de dados e com o gerenciamento destas camadas.

Como uma ferramenta de pesquisa, o **SimATM** visa proporcionar um ambiente para o desenvolvimento de novos protocolos, algoritmos de gerenciamento de recursos e mecanismos de controle fluxo e de congestionamento, entre outros.

Como uma ferramenta de análise, o **SimATM** visa proporcionar um ambiente para a avaliação de desempenho de protocolos, algoritmos e mecanismos utilizados em redes ATM.

Como uma ferramenta de projeto de redes ATM, o **SimATM** visa proporcionar um ambiente para teste de redes sobre diferentes condições de carga e disposições geográficas. Assim, a especificação dos equipamentos a serem utilizados em uma rede pode ser feita através da otimização de variáveis chave, como por exemplo a capacidade de *buffers*, número de portas, vazão e atraso de processamento em chaveadores. Distâncias podem ser reduzidas ou aumentadas a fim de melhor caracterizar redes locais ou de grande distância.

Como uma ferramenta de validação de modelos, o **SimATM** pode ser utilizado para a validação de novos modelos matemáticos, como por exemplo modelos de geração de tráfego [10] e de geração de falhas de transmissão.

O **SimATM** pode ainda ser utilizado como uma ferramenta de análise de estratégias de interconexão de redes, permitindo avaliar o impacto do tráfego de redes legadas sobre redes ATM, através do uso de modelos de tráfego apropriados.

A discussão deste trabalho está dividida em nove capítulos e dois apêndices. O Capítulo 2 descreve a operação básica das redes ATM. O Capítulo 3 descreve a padronização de equipamentos de tecnologia ATM. Estes capítulos fornecem os elementos necessários para a compreensão da estrutura desenvolvida para o simulador. O Capítulo 4 discute a simulação de redes ATM. O Capítulo 5 descreve o desenvolvimento do **SimATM**. O Capítulo 6 descreve a estrutura atual do simulador. O Capítulo 7 mostra como foram implementados os elementos da estrutura do **SimATM**. O Capítulo 8 mostra um exemplo de simulação de rede ATM, comparando os resultados obtidos com a teoria de filas [11] e mostrando o desempenho do simulador durante a simulação. Finalmente, no Capítulo 9 são apresentadas as conclusões do trabalho e sugestões para trabalhos futuros.

O Apêndice A mostra os fluxogramas dos processos dos modelos de camadas desenvolvidos para os equipamentos ATM da rede.

O Apêndice B descreve a linguagem de comandos desenvolvida para o **SimATM**.

1.1 - Referências Bibliográficas

[1] ITU-T *Recommendation* I.356, "B-ISDN ATM *Layer Cell Transfer Performance*", *Geneva, January* 1996.

[2] ITU-T *Recommendation* I.371, "*Traffic Control and Congestion Control in* B-ISDN", *Frozen Issue, Geneva, July* 1995.

[3] Jae-Il Jung, "*Quality of Service in Telecommunications Part I: Proposition of a* QoS *Framework and Its Application to* B-ISDN", IEEE *Communications*, Vol. 34 No. 8 pg. 108, *August* 1996.

[4] ITU-T *Recommendation* I.113, "*Vocabulary of Terms for Broadband Aspects of* ISDN", *Geneva*, 1991.

[5] http://www.itu.ch/

[6] http://www.atmforum.com/

[7] Daniel Minolli, Anthony Alles, "LAN, ATM, *and LAN emulation technologies*", *Artech House*, 1996.

[8] Anthony Alles, "ATM *Internetworking*", *White Paper*, *Cisco Systems, Inc.*, *May* 1995.

[9] Martin De Prycker, "*Asynchronous Transfer Mode*: *Solutions for Broadband* ISDN", *Prentice Hall*, *Third Edition*, 1995.

[10] Abdelnaser Adas, "*Traffic Models in Broadband Networks*", IEEE *Communications*, Vol. 35 No. 7 pg. 82, *July* 1997.

[11] Leonard Kleinrock, "*Queueing Systems – Volume* I: *Theory*", *John Willey & Sons*, 1975.

Capítulo 2

ATM

2.1 - Breve Histórico

Em meados dos anos 70, o contínuo avanço no desenvolvimento das tecnologias de transmissão e chaveamento digital tornaram real a possibilidade de construção de uma rede digital capaz de integrar diferentes tipos de serviços, conhecida como RDSI – Rede Digital de Serviços Integrados (ISDN – *Integrated Services Digital Network*), que estava sendo desenvolvida pelo CCITT – *Consultive Committee on International Telegraphy and Telephony*, atualmente ITU-T – *International Telecommunications Union – Telecommunication Standardization Sector.*

Em 1984, um conjunto de recomendações chamado de Série I [1], foi publicado pelo CCITT e incentivou que indústrias e provedoras de serviço iniciassem o desenvolvimento de equipamentos e serviços baseados na ISDN. Contudo, a série I ainda não estava suficientemente detalhada e somente com a versão de 1988 tornou-se possível a implementação preliminar de redes ISDN. Ao final dos anos 80 muitos dos esforços de projeto e desenvolvimento do ITU-T tornaram-se voltados para um novo conceito de rede que seria muito mais revolucionário que a própria rede ISDN. Este novo conceito tem sido referenciado como RDSI-FL – Rede Digital de Serviços Integrados Faixa Larga (B-ISDN – *Broadband Integrated Service Digital Network*), e se caracteriza como uma extensão da ISDN.

Como parte da Série I de recomendações sobre a ISDN, o ITU-T anexou as duas primeiras recomendações relacionadas com a B-ISDN: Vocabulário de Termos para os Aspectos Faixa Larga da ISDN (I.113) [2] e Aspectos Faixa Larga da ISDN (I.121) [3]. Estas recomendações forneceram uma descrição preliminar e apontaram as direções básicas no processo de padronização da futura B-ISDN.

De acordo com a Recomendação I.121 [3] os fatores que estão guiando os trabalhos do ITU-T no desenvolvimento da B-ISDN são:

- A emergente demanda por serviços faixa larga.
- A disponibilidade de tecnologias de alta velocidade de transmissão, chaveamento e processamento de sinais.
- A capacidade de melhor processamento de imagens e dados.
- A necessidade de integrar serviços interativos e distributivos [4].
- A necessidade de integrar o modo de transferência[1] de redes de circuitos e de pacotes [1] em uma rede faixa larga universal.
- A necessidade de prover flexibilidade no atendimento de requisitos tanto de usuários quanto de provedores.
- A necessidade de cobrir os aspectos relacionados com a B-ISDN em recomendações do ITU-T.

De acordo com a Recomendação I.113 [2] um novo modo de transferência, chamado Modo de Transferência Assíncrono (ATM – *Asyncronous Transfer Mode*), foi definido para ser utilizado na B-ISDN pelas seguintes razões [2]:

- Possibilita o acesso flexível à rede devido ao conceito de transporte de células.
- Possibilita a alocação dinâmica de largura de faixa sob demanda.
- Permite a alocação da capacidade de transporte de forma flexível.
- É independente do meio físico de transporte de dados.

Uma descrição mais aprofundada sobre outros modos de transferência, incluindo ATM, pode ser encontrada em [6].

Em 1991, devido a demora do ITU-T no desenvolvimento das normas relacionadas com a B-ISDN e ao grande nível de interesse na tecnologia ATM, um grupo de indústrias se organizou em um consórcio chamado ATM *Forum*, com o objetivo de acelerar e facilitar o desenvolvimento da tecnologia ATM.

---

[1] Modo de Transferência é o termo usado pelo ITU-T para descrever a técnica empregada em uma rede de telecomunicações para cobrir os aspectos relacionados a transmissão, multiplexação e chaveamento.

Atualmente, o consórcio ATM *Forum* é composto por aproximadamente 800 indústrias, concessionárias, vendedores, clientes e outros grupos. O ATM *Forum* não é um organismo de padronização e, portanto, atua apenas na elaboração de acordos de implementação baseados em normas internacionais.

Entretanto, o ATM *Forum* tem produzido especificações para funções que não tem sido preocupação de organismos internacionais de padronização. Estas especificações adicionais são específicas para redes de dados e para ambiente de redes locais, e preocupam-se com a interconexão de redes através da tecnologia ATM [7]. Como exemplo de tais especificações podemos citar:

- LANE - ATM LAN *Emulation* [8]
- MPOA - *Multiprotocol Over* ATM [9]
- Implantação do serviço de taxa de *bits* disponível (ABR - *Available Bit Rate*)

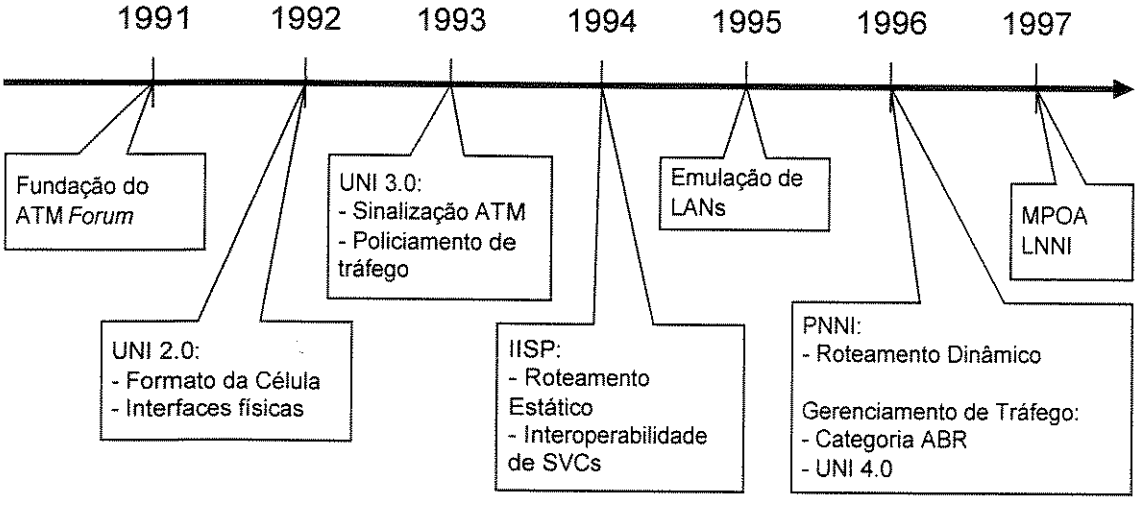


Figura 2.1 – Histórico do desenvolvimento de especificações no ATM *Forum* [7]

A Figura 2.1 apresenta um breve histórico do desenvolvimento de especificações no ATM *Forum*. Maiores detalhes sobre essas especificações podem ser encontradas em [10].

2.2 - Aspectos Básicos

ATM é uma tecnologia de transmissão, multiplexação e chaveamento usada para transportar pequenos pacotes[2] de tamanho fixo, chamados de células, sobre uma rede de alta velocidade.

[2] O ATM é um modo de transferência baseado no transporte de pacotes [1].

A seguir apresentaremos algumas características básicas da tecnologia ATM [5] [6][11][12][13][14][15].

2.2.1 - Características Básicas da Tecnologia ATM

2.2.1.1 - Pacotes de Tamanho Fixo

ATM utiliza pequenos pacotes de tamanho fixo chamados de células. Uma célula tem 53 *bytes*, sendo 5 *bytes* de cabeçalho e 48 *bytes* para o campo de informações. Toda a informação (voz, vídeo, dados, etc.) é transportada pela rede através de células ATM.

2.2.1.2 - A Funcionalidade do Cabeçalho das Células é Reduzida

Para garantir o processamento rápido dentro da rede, o cabeçalho das células ATM é limitado em termos de funcionalidade. Sua principal função é identificar uma conexão virtual (lógica) por meio de identificadores que são selecionados em uma fase de estabelecimento de conexão e garantem um encaminhamento adequado de cada célula pela rede.

Além dos identificadores de conexão virtual, um número bem limitado de outras funções é suportado pelo cabeçalho. Para evitar o encaminhamento errado das células dentro da rede ATM, devido a erros nos identificadores de conexão virtual, foi inserido um campo de proteção contra erros no cabeçalho (HEC - *Header Error Control*) da célula ATM.

Dada a limitada funcionalidade do cabeçalho das células ATM, o seu processamento é bastante simples e pode ser feito a taxas muito altas (155.52 Mbps até Gbps).

2.2.1.3 - O Campo de Informações das Células é Pequeno

O campo de informações da célula ATM foi padronizado pelo ITU-T [16] em 48 *bytes* a partir de um compromisso firmado entre vários grupos de interesse e levando-se em conta uma série de fatores conflitantes, dos quais podemos destacar [6]:

- Atrasos na rede
- Eficiência de transmissão
- Complexidade de implementação

2.2.1.4 - Roteamento de Células

O fluxo de informações é estabelecido através de percursos predefinidos, chamados canais virtuais (VCs – *Virtual Channels*). O cabeçalho das células ATM contém identificadores que amarram a célula ao seu percurso.

As células de um canal virtual seguem o mesmo percurso através da rede e são entregues ao destino na mesma ordem que foram inseridas na rede.

2.2.1.5 - Funções de Adaptação

O fluxo de informações de um usuário final ATM precisa ser adaptado para trafegar através da rede ATM. Esta adaptação é feita através de protocolos específicos que possibilitam o tratamento diferenciado para cada tipo de serviço.

Assim, o fluxo de informações de um usuário final ATM é fragmentado em células no ponto de ingresso na rede e recuperado no ponto de egresso da rede.

2.2.1.6 - Nenhuma Proteção ou Controle de Fluxo no Nível de Enlace

Se um enlace introduz um erro durante a transmissão de células ATM, portanto causando a perda de informações de usuário, nenhuma ação será tomada no nível deste enlace para corrigir tal erro. Esta proteção de erro pode ser omitida, uma vez que os enlaces utilizados nas redes ATM apresentam alta qualidade, ou seja, possuem uma baixa taxa de erro de *bits* (BER – *Bit Error Rate*).

As redes ATM também não tem controle de fluxo no nível de enlace, uma vez que a lógica de processamento necessária para tal controle é muito complexa para ser acomodada às altas taxas deste nível. Em vez disto, as redes ATM utilizam um conjunto de controles de taxa de entrada que limita o tráfego entregue à rede. Tal controle de fluxo possui características muito diferentes daquele utilizado em redes tradicionais e por isso tem sido alvo de muitos estudos.

2.2.1.7 - Operação Orientada a Conexão

ATM provê um serviço de transmissão de dados orientado a conexão. Isto significa que antes que qualquer informação seja transmitida entre duas estações (*hosts*) ATM, uma fase de estabelecimento de conexão virtual/lógica deve ser realizada com o objetivo de permitir à rede reservar os recursos necessários. Se os recursos disponíveis não forem suficientes, tal conexão será recusada. Ao final da transmissão os recursos da rede são desalocados e a conexão é encerrada.

O serviço não orientado a conexão [1] é suportado em redes ATM, mas neste caso o fluxo de dados será transmitido sobre um ou mais caminhos preestabelecidos.

2.2.1.8 - Controle de Congestionamento

Existe apenas uma coisa que uma rede ATM pode fazer quando um nó da rede torna-se congestionado: células ATM serão descartadas até que o problema seja resolvido. Algumas células (baixa prioridade) podem ser marcadas de forma que se ocorrer um congestionamento elas serão as primeiras a serem descartadas.

Os pontos finais de uma conexão ATM não são notificados quando células são perdidas. Portanto, de forma geral, cabe as funções de adaptação detectar e recuperar as informações perdidas devido a congestionamentos.

Maiores detalhes sobre controle de fluxo e de congestionamento serão apresentados no item 2.3.7 - Gerenciamento de Tráfego.

2.2.1.9 - Uso de Conexões Virtuais

O ATM usa conexões virtuais (Atualmente chamadas conexões de canal virtual – VCCs – *Virtual Channel Connections*) como um mecanismo para transportar dados entre uma fonte e um destino. Uma conexão virtual é dedicada para um par fonte/destino. Assim, uma ou mais conexões virtuais podem utilizar o mesmo enlace físico.

2.2.1.10 - Controle de Erro

A rede ATM verifica apenas o cabeçalho das células a fim de encontrar erros. Se um erro for encontrado e não puder ser corrigido, a célula errada será simplesmente descartada.

2.2.1.11 - Suporte para Qualidade de Serviço

A rede ATM suporta qualidade de serviço. Isto significa que a rede irá prover ou reservar recursos que garantam um valor especificado mínimo de vazão e de perda de informações de usuário e um valor máximo de atraso, durante a duração de uma dada conexão.

Este suporte de QoS por conexão habilita as redes ATM a atender a qualquer tipo atual de tráfego sobre uma mesma rede.

2.2.2 - Formato das Células ATM

As células ATM tem 53 *bytes*, sendo que 5 *bytes* são de cabeçalho e os restantes 48 *bytes* são destinados ao campo de informações.

Dois formatos diferentes para o cabeçalho da célula ATM foram definidos pelo ITU-T [16], conforme mostra a Figura 2.2: um para a interface usuário-rede (UNI – *User-Network*

Interface) e outro para a interface rede-rede (NNI – *Network-to-Network Interface*). A diferença entre os formatos está nos 4 *bits* usados para o controle de fluxo genérico (GFC – *Generic Flow Control*) no cabeçalho da célula UNI, que são realocados para o campo de identificador de caminho virtual (VPI – *Virtual Path Identifier*) no cabeçalho da célula NNI. Isto faz sentido, pois, por definição, o controle de fluxo não é realizado através de uma NNI. Uma outra vantagem é que grandes redes de chaveadores ATM interconectados podem suportar mais caminhos virtuais (ex. redes privadas virtuais), fazendo portanto o chaveamento de um grande número de VCs mais eficientemente.

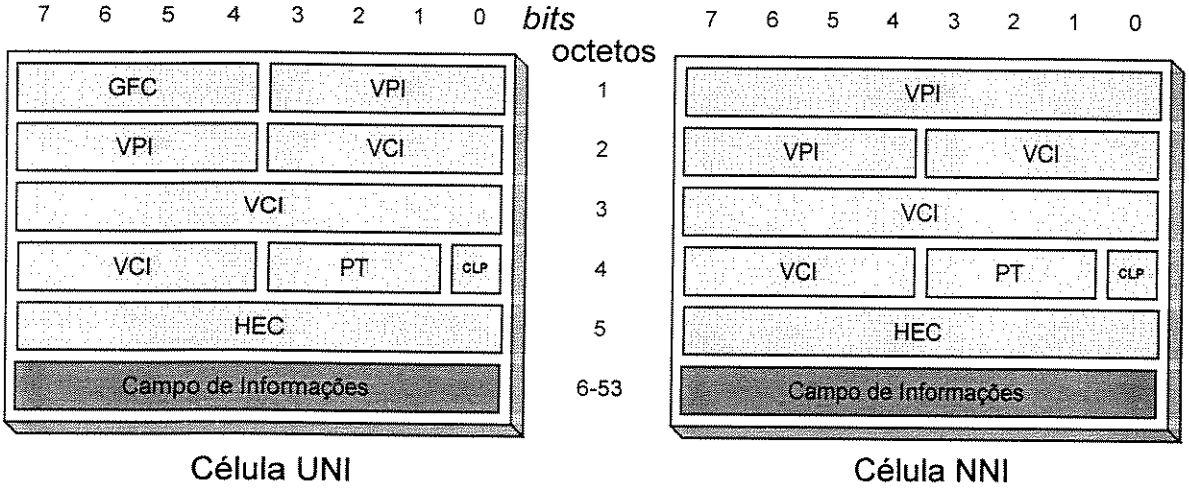


Figura 2.2 – Formato das células ATM na UNI e NNI

Os campos definidos no cabeçalho da célula ATM são [16]:

- Controle de Fluxo Genérico (GFC - *Generic Flow Control*) - Estes quatro *bits*, presentes somente na UNI, tem significado apenas entre o usuário final ATM e o chaveador adjacente (interface UNI). O GFC foi concebido como um meio para que certos mecanismos de acesso priorizem células.
- Identificador de Caminho Virtual (VPI – *Virtual Path Identifier*) – Na interface UNI possui 8 *bits* e na interface NNI 12 *bits*.
- Identificador de Conexão Virtual (VCI – *Virtual Channel Identifier*) – Possui 16 *bits* na UNI e na NNI.
- Tipo de Carga (PT – *Payload Type*) - Este campo de 3 *bits* é usado para indicar qual o conteúdo de uma célula.
- Prioridade de Perda de Célula (CLP – *Cell Loss Priority*). Este *bit* é usado para sinalizar quando uma célula está de acordo com um contrato de tráfego

preestabelecido (CLP=0) ou passível de ser descartada quando ocorre um congestionamento na rede (CLP=1).

- Controle de Erro do Cabeçalho (HEC – *Header Error Control*). Usado para detectar e corrigir erros de *bits* no cabeçalho da célula ATM.

A Tabela 2.1 mostra os valores binários possíveis e os respectivos significados para o campo PT.

Valor binário	Significado
000	Célula de dados de usuário. Congestionamento não experimentado. AUU[3] = 0.
001	Célula de dados de usuário. Congestionamento não experimentado. AUU = 1.
010	Célula de dados de usuário. Congestionamento experimentado. AUU = 0.
011	Célula de dados de usuário. Congestionamento experimentado. AUU = 1.
100	Célula associada ao fluxo OAM F5 de segmento
101	Célula associada ao fluxo OAM F5 fim a fim
110	Célula de gerenciamento de recursos
111	Reservado para funções futuras

Tabela 2.1 – Descrição dos Valores do Campo PT [6]

2.2.3 - Conexões ATM

Conexões ATM podem ser classificadas de acordo com a forma que são estabelecidas e com o número de usuários finais ATM envolvidos em uma transmissão.

Segundo a forma como são estabelecidas, existem dois tipos fundamentais de conexões ATM:

- Conexões Virtuais Permanentes (PVCs – *Permanent Virtual Connections*) – São conexões estabelecidas e encerradas por um mecanismo externo, tipicamente um software de gerenciamento de rede, e geralmente permanecem ativas por longo tempo.
- Conexões Virtuais Chaveadas (SVCs – *Switched Virtual Connections*) – São conexões estabelecidas e encerradas automaticamente através de um protocolo de sinalização e permanecem ativas até que um sinal indique que a conexão deve ser encerrada.

[3] AUU – ATM-*user-to*-ATM-*user indication*

Segundo o número de usuários finais ATM envolvidos na transmissão também existem dois tipos fundamentais de conexões ATM:

- Conexões Ponto a Ponto (*Point-to-point connections*) – Conecta apenas dois usuários finais ATM e podem ser unidirecionais ou bidirecionais.
- Conexões Ponto para Multiponto (*Point-to-multipoint connections*) – Conecta um usuário final ATM fonte (nó raiz) com múltiplos usuários finais ATM de destino (nós folhas). A replicação de células deve ser feita no nó onde a conexão ATM se divide a fim de possibilitar que todos os nós folhas recebam suas células. Tais conexões são unidirecionais, ou seja, permitem que o nó raiz transmita para os nós folhas, mas não permitem que os nós folhas transmitam para o nó raiz ou para outros nós folhas. As conexões ponto para multiponto desempenham um papel importante na habilidade de conduzir tráfego *broadcast*[4] e *multicast*[5] sobre redes ATM.

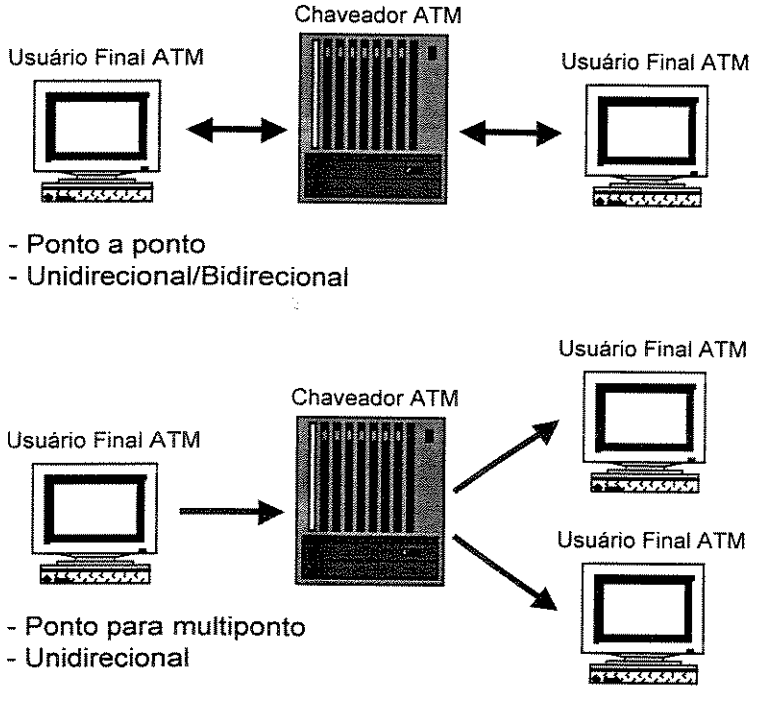


Figura 2.3 – Conexões ATM ponto a ponto e ponto para multiponto

[4] *Broadcast* – Uma estação transmite para todas as outras estações de uma rede.

[5] *Multicast* – Uma estação transmite para um grupo específico de estações de uma rede.

2.2.3.1 - Conexões Virtuais

Uma conexão virtual é um canal lógico entre dois usuários finais ATM e é usada para transportar células. As recomendações do ITU-T se referem a estas conexões lógicas de ponta a ponta entre dois usuários finais ATM como: conexão de canal virtual (VCC - *Virtual Channel Connection*). Uma VCC é uma concatenação de um ou mais canais virtuais (VC – *Virtual Channel*). Um canal virtual simplesmente descreve o transporte unidirecional de células ATM com um identificador comum, VCI, em cada célula. Um enlace de canal virtual (VCL - *Virtual Channel Link*) é um canal virtual entre dois pontos (ex. estação e chaveador) em uma VCC onde o VCI é atribuído, trocado ou removido.

O VCI no cabeçalho da célula tem significado apenas para as células fluindo sobre um enlace de canal virtual (VCL). Em outras palavras, o VCI para as células fluindo em um VCC pode mudar conforme elas passam através de diferentes chaveadores.

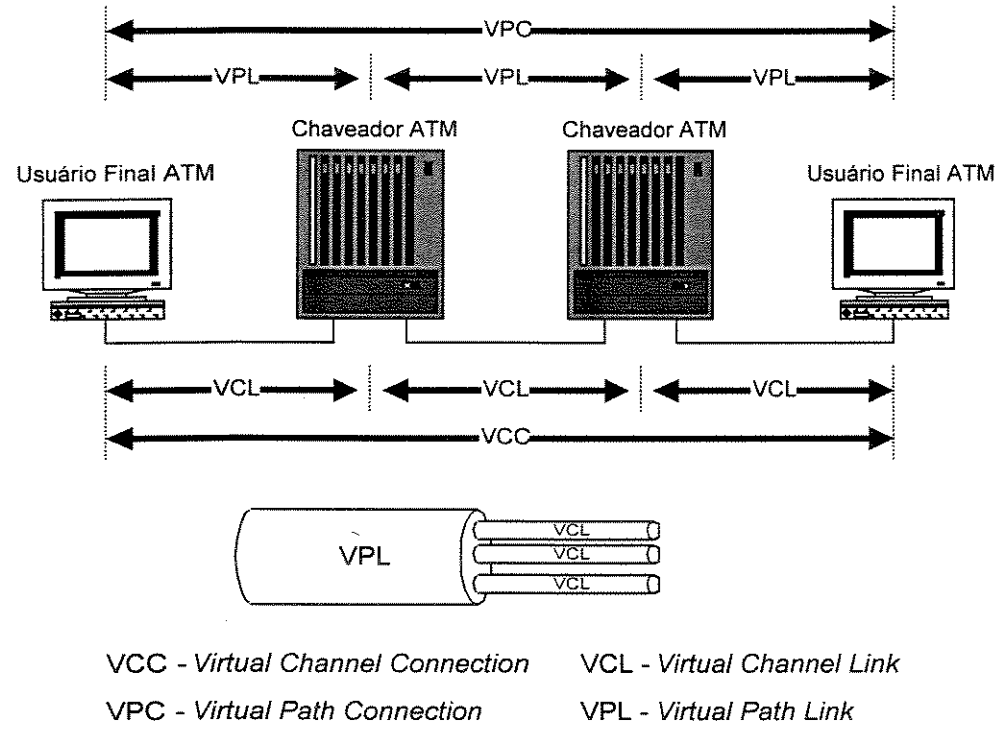


Figura 2.4 – Relacionamento entre VCC e VCL, e entre VPC e VPL

2.2.3.2 - Caminhos Virtuais

Um caminho virtual ATM (VP – *Virtual Path*) é um grupo de canais virtuais. Cada canal virtual é associado a um caminho virtual. Múltiplos canais virtuais podem ser associados com um mesmo caminho virtual. Um caminho virtual está apoiado sobre um enlace de caminho virtual (VPL – *Virtual Path Link*). Um VPL é um caminho virtual entre

dois pontos onde o VPI é atribuído, trocado, ou removido. Uma conexão de caminho virtual (VPC – *Virtual Path Connection*) é a concatenação de um ou mais VPLs.

Os conceitos de caminho virtual e conexão virtual oferecem um mecanismo flexível e robusto para o estabelecimento e o chaveamento de conexões dentro de uma rede ATM. Combinados existem 24 *bits* para o VPI/VCI na UNI e 28 *bits* na NNI.

2.2.4 - Roteamento de Células

O roteamento de células através de uma rede ATM é baseado no conceito de troca de identificadores (*label swapping*). *Label swapping* é uma técnica de roteamento usada em outras tecnologias de chaveamento de pacotes, tal como X.25 e *frame relay* [1][5]. O funcionamento desta técnica é bastante simples: cada pacote contém um identificador de conexão lógica. Em cada chaveador ao longo de uma dada conexão existe uma tabela de roteamento que contém um registro que relaciona o par identificador e porta de entrada com o par identificador e porta de saída. Assim, quando um pacote chega em uma porta de entrada de um chaveador, o seu identificador é lido e uma consulta a tabela de roteamento é feita, a fim de determinar com qual identificador e para qual porta de saída o pacote deve ser enviado.

A técnica *label swapping* é eficiente por várias razões [5]:

- O processo de extração e processamento de identificadores é muito rápido, uma vez que tipicamente tais identificadores tem alguns *bits* de comprimento, ao contrário de soluções que utilizam endereços.
- O roteamento de pacotes é feito em *hardware*, o que minimiza o tempo gasto em cada chaveador e viabiliza o tráfego sensível a atrasos.
- A tabela de roteamento é construída através de aplicativos de gerenciamento de rede ou protocolos de sinalização antes que qualquer pacote pertencente a uma conexão seja enviado. Em outras palavras, todas as decisões de roteamento de pacotes são tomadas antes que qualquer pacote seja transmitido.
- Os pacotes possuem pequenos cabeçalhos e são transportados através da rede em seqüência, sobre um caminho predeterminado e com um atraso mínimo.

Nas redes ATM, o identificador de conexões lógicas é uma combinação dos campos VPI e VCI presentes no cabeçalho das células. Como existem dois identificadores em redes ATM e por conseqüência caminhos virtuais e conexões virtuais, dois níveis de chaveamento

podem ser executados: chaveamento VP (VP *switching*) e chaveamento VC (VC *switching*). O chaveamento VP apenas executa a troca do campo VPI das células, enquanto o chaveamento VC executa a troca dos campos VPI e VCI.

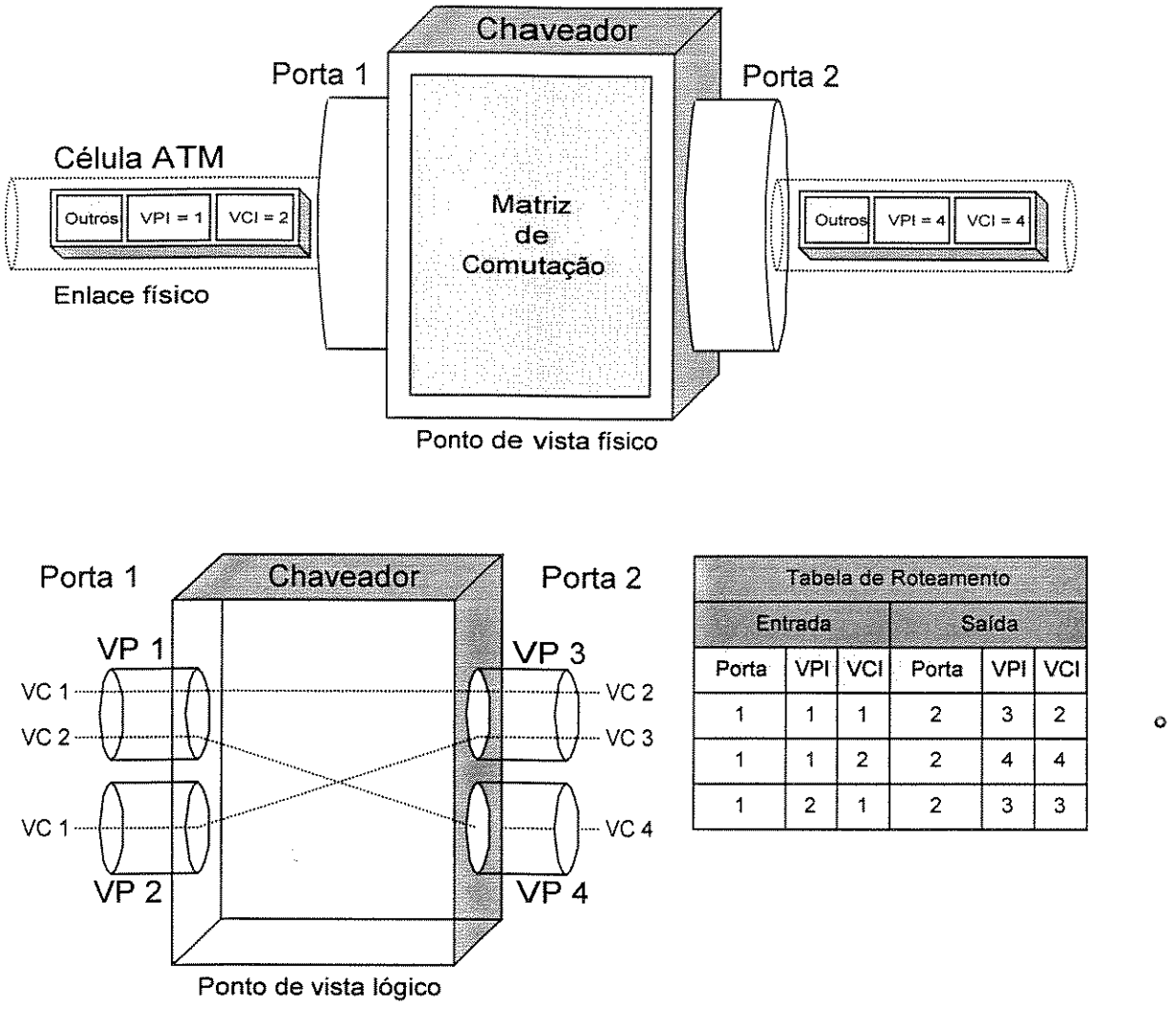


Figura 2.5 – Operação de um Chaveador ATM

A Figura 2.5 mostra a operação básica de um comutador ATM. Do ponto de vista físico, o fluxo de *bits* correspondente a uma célula ATM é chaveado de uma porta de entrada para uma porta de saída através de uma matriz de comutação (*Switch Fabric*). Do ponto de vista lógico, as células ATM são enviadas para uma porta de saída previamente definida, a partir do processamento de um registro feito na tabela de roteamento. Quando uma célula chega a uma porta de entrada, os seus identificadores virtuais são lidos e utilizados para uma consulta na tabela de roteamento, que determina os novos identificadores virtuais e a porta de saída para esta célula.

2.3 - Arquitetura das Redes ATM

Arquiteturas de rede podem ser estruturadas a partir de modelos de referência. Um modelo de referência é composto por camadas sobrepostas, onde cada camada possui protocolos e funções específicas.

Dois dos modelos de referência comumente utilizados são: Modelo OSI [5][13][17] e Modelo SNA [5]. O Modelo SNA – *Systems Network Architecture* foi desenvolvido como referência para uma arquitetura de rede proprietária da IBM. O Modelo OSI – *Open Systems Interconnection* foi desenvolvido nos anos 80 por várias organizações internacionais de padronização, entre elas a ISO – *International Standards Organization* [18] e o ITU-T.

O Modelo OSI é organizado em sete camadas: Camada Física, Camada de Enlace, Camada de Rede, Camada de Transporte, Camada de Sessão, Camada de Apresentação e Camada de Aplicação. Cada camada contém vários protocolos e é responsável por funções específicas que visam suportar aplicativos de usuários finais, que acessam a rede através da Camada de Aplicação.

No Modelo OSI uma camada é considerada um provedor de serviços para a camada imediatamente superior e um usuário dos serviços das camadas inferiores.

A arquitetura das redes ATM é baseada no Modelo de Referência de Protocolos da B-ISDN (B-ISDN PRM – B-ISDN *Protocol Reference Model*) [17] que foi desenvolvido pelo ITU-T e documentado na Recomendação I.321 [19]. O Modelo da B-ISDN por sua vez se baseia no Modelo OSI e nas recomendações ISDN. Entretanto, existem algumas diferenças entre o Modelo OSI e o modelo adotado para B-ISDN. Uma comparação entre o Modelo OSI e o Modelo da B-ISDN é feito em [17].

O Modelo da B-ISDN é um modelo tridimensional composto por três planos e três camadas: plano de usuário (*User Plane*), plano de controle (*Control Plane*) e plano de gerenciamento (*Management Plane*); camada física (*Physical Layer*), camada ATM (ATM *Layer*) e camada de adaptação ATM (AAL – ATM *Adaptation Layer*).

O plano de usuário provê a transferência de informações do usuário. Ele contém uma camada física, uma camada ATM e várias AALs que suportam diferentes serviços, tal como voz e vídeo. O plano de usuário é responsável por prover transferência, controle de fluxo e recuperação de informações de usuários.

O plano de controle fornece funções de sinalização e de controle necessárias ao estabelecimento, gerenciamento e finalização de conexões virtuais chaveadas. O plano de

controle compartilha com o plano de usuário as camadas física e ATM e possui uma AAL específica de sinalização. O plano de controle não é necessário quando se utiliza apenas conexões virtuais permanentes (PVCs).

O plano de gerenciamento habilita o trabalho conjunto dos planos de usuário e de controle e fornece dois tipos de funções: gerenciamento de planos e gerenciamento de camadas. O gerenciamento de planos não possui estrutura em camadas e é responsável pela coordenação de todos os planos. O gerenciamento de camadas é responsável pelo gerenciamento de entidades (vide próximo item) nas camadas e pela execução de serviços de operação, administração e manutenção (OAM – *Operation, Administration and Maintenance*).

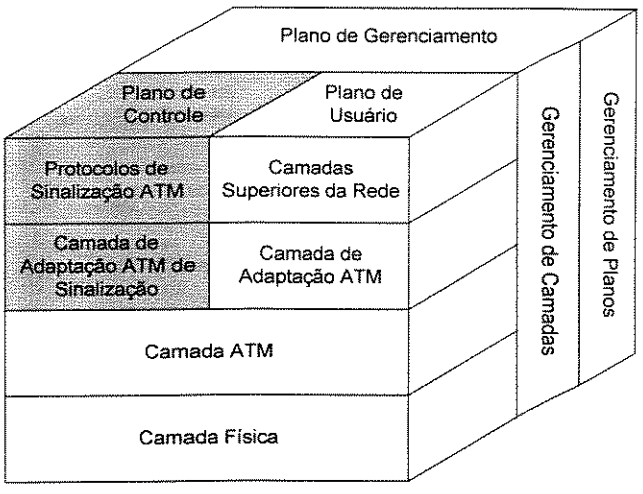


Figura 2.6 – Modelo de Referência de Protocolos da B-ISDN

A área sombreada na figura indica as funções necessárias para a implementação de conexões virtuais chaveadas (SVCs).

2.3.1 - Nomenclatura Básica

A fim de melhor compreenderemos a funcionalidade das camadas do Modelo de Referência de Protocolos da B-ISDN, alguns conceitos básicos do Modelo OSI são necessários (baseados na Figura 2.7) [13]:

- Unidade de Dados de Serviço (SDU – *Service Data Unit*) – Para a camada N, uma SDU é uma PDU da camada N+1, que foi transferida transparentemente da camada N+1 para a camada N, através de uma primitiva. Então, uma SDU da camada N corresponde às informações de uma PDU da camada N+1.

- Informação de Controle de Protocolo (PCI – *Protocol Control Information*) – São informações de controle acrescentadas à SDU da camada N por um protocolo. Tais informações acrescentadas à SDU também são chamadas de cabeçalhos ou *trailers*, dependendo da posição em que são inseridos.
- Unidade de Dados de Protocolo (PDU – *Protocol Data Unit*) – A PDU da camada N consiste da junção das informações de controle de protocolo (PCI) e da SDU dessa camada. A PDU da camada N é enviada para a camada N-1 através de uma primitiva, e será tratada então como a SDU da camada N-1.
- Entidades (*Entities*) – As funções desempenhadas por uma camada podem ser estruturadas e particionadas em pequenos módulos chamados entidades. Entidades de protocolo podem ser implementadas em *hardware* ou em *software*.
- Ponto de Acesso ao Serviço (SAP – *Service Acess Point*) – É uma interface entre camadas, implementada em *hardware* ou *software*, que possibilita a troca de serviços.
- Primitivas (*Primitives*) – Descrevem de maneira simplificada a troca de informações e controles através de um ponto de acesso ao serviço (SAP). Quando uma camada quer passar informações e controles para outra camada ela utiliza primitivas.

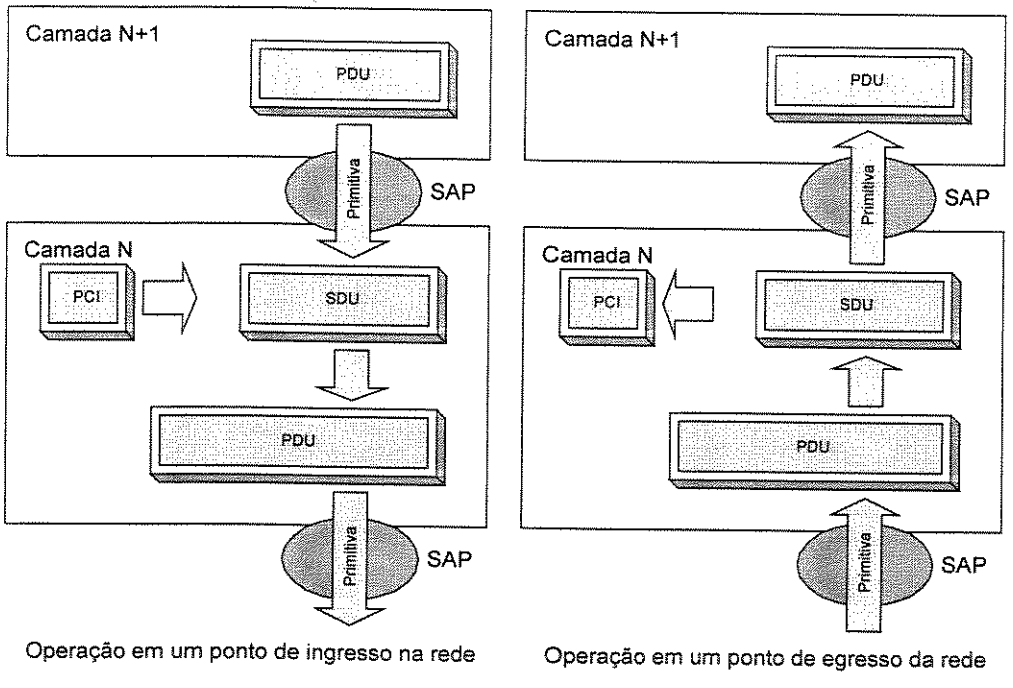


Figura 2.7 – Conceitos básicos do Modelo OSI

A Figura 2.7 mostra a troca de informações entre camadas utilizando os conceitos básicos do Modelo OSI. Em um ponto de ingresso na rede, uma PDU da camada N+1 é transferida para a camada N através de uma primitiva, em um ponto de acesso ao serviço (SAP). Desta forma as informações das camadas superiores da rede são encapsuladas, até que na camada física, a PHY-PDU resultante é transferida até o próximo equipamento da rede, e assim sucessivamente, até o ponto de egresso da rede, onde ocorre o processo contrário. Cada camada retira a PCI inserida junto à SDU da camada superior, até que a informação original seja entregue ao usuário de destino.

2.3.2 - Camada Física

A camada física é responsável pela transmissão das células entre dois equipamentos ATM através de um enlace físico específico. De acordo com a Recomendação I.321 [19], a camada física se divide em duas subcamadas:

- Subcamada Dependente do Meio Físico (*Physical Medium Dependent Sublayer*)
- Subcamada de Convergência de Transmissão (*Transmition Convergence Sublayer*)

2.3.2.1 - Subcamada Dependente do Meio Físico

A subcamada dependente do meio físico, como o próprio nome já diz, é dependente do meio físico utilizado. Esta subcamada é responsável pela transmissão e recepção sincronizada do fluxo contínuo de *bits*.

A subcamada dependente do meio físico preocupa-se com as seguintes funções e especificações:

- Código de Linha, Alinhamento de *Bits* e Conversão Eletro-Óptica – Estas funções estão relacionadas com o processo de codificação de *bits* em sinais elétricos ou ópticos.
- Características do Meio Físico – Especifica as características do meio físico de transporte, tal como o tipo de fibra óptica, o comprimento de onda do transmissor/receptor, a sensitividade do receptor, etc.

2.3.2.2 - Subcamada de Convergência de Transmissão

A subcamada de convergência se situa acima da subcamada dependente do meio físico e abaixo da camada ATM. Esta subcamada encapsula o fluxo de células proveniente da

camada ATM em *frames*[6] de transmissão e envia para a subcamada dependente do meio físico.

De acordo com a Recomendação I.321 [19], a subcamada de convergência de transmissão é responsável pelas seguintes funções:

- Geração e Recuperação dos *Frames* de Transmissão – A transmissão de informações no nível da camada física é feita através de *frames*. Esta função preocupa-se em gerar e manter a estrutura de *frames* apropriada para uma dada taxa de transmissão.
- Adaptação do *Frame* de Transmissão – Esta função encapsula o fluxo de células proveniente da camada ATM em *frames* na extremidade de inicio de um enlace e extrai o fluxo de células dos *frames* na extremidade final desse mesmo enlace.
- Delineação de Célula – Com o propósito de sincronizar a transmissão, o fluxo de *bits* pode ser embaralhado. Esta função identifica o limite das células de tal forma que estas possam ser recuperadas depois de serem embaralhadas.
- Geração da Seqüência de Controle de Erro de Cabeçalho e Verificação do Cabeçalho das Células – Cada cabeçalho de células é protegido por um campo HEC, conforme já abordamos anteriormente. Essa função gera e verifica este campo a fim de encontrar erros. Se um erro for encontrado, ele será corrigido se possível. Se não for possível corrigir tal erro, a célula será descartada.
- Ajuste da Taxa de Células - Este processo adapta a velocidade do fluxo de células da camada ATM à atual taxa de células na interface da camada física. Para isso, insere e suprime células vazias, de forma a adaptar a taxa de células ATM válidas à capacidade de carga de informações do sistema de transmissão.

2.3.2.3 - Interfaces Físicas ATM

O ATM pode usar qualquer meio físico capaz de carregar suas células. Alguns dos padrões de interfaces existentes e outros que estão sendo definidos são mostrados na Tabela 2.2.

[6] *Frames* – Segundo *Black* [13], na camada física as PHY-PDUs podem ser chamadas de *frames*.

Descrição	Taxa (Mbps)	Especificação
ATM 25.6 Mb sobre UTP-3[7]	25.6	ATM *Forum*
51.84 Mb SONET[8] STS-1[9] sobre UTP-3	51.84	ATM *Forum*
TAXI[10] 100 Mb sobre MMF[11]	100	ATM *Forum*
155 Mb *FibreChannel* sobre MMF	155.52	ATM *Forum*
155 Mb SONET STS-3c[12] sobre SMF[13]/MMF	155.52	ITU-T I.432
155 Mb SONET STS-3c sobre UTP-3	155.52	ATM *Forum*
155 Mb SONET STS-3c sobre UTP-5	155.52	ATM *Forum*
DS-1[14]	1.544	ITU-T G.804
DS-3[15]	44.736	ITU-T G.703
E1[16]	2.048	ATM *Forum*
E3[17]	34.368	ATM *Forum*
E4	139.264	ATM *Forum*
622 Mb SONET STS-12c	622.08	ATM *Forum*
2488 Mb SONET STS-48c	2488.32	ATM *Forum*

Tabela 2.2 – Interfaces ATM da Camada Física [5]

2.3.2.4 - Camada Física para a Interface Baseada em Células

A Recomendação I.432 do ITU-T [20] define uma camada física para a interface baseada puramente em células.

2.3.2.4.1 - Características do Meio Físico

[7] UTP – *Unshielded Twiested Pair* – Par de fios de cobre trançados. O ATM *Forum* especificou duas categorias: 3 e 5.

[8] SONET – *Syncronous Optical Network* – Padrão de transmissão óptica digital de alta velocidade e qualidade definido pela ANSI.

[9] STS-1 – *Synchronous Transport Signal*-1 – Padrão de sinal básico SONET para transmissão óptica.

[10] TAXI – *Transparent Asyncronous Transmiter/Receiver Interface* – Uma interface que provê conectividade sobre enlaces de fibra multimodo.

[11] MMF – *Multi-mode Fiber* – Fibra multimodo.

[12] STS-X – *Synchronous Transport Signal*-X – Formato de sinal SONET com X vezes a taxa básica.

[13] SMF – *Single-mode Fiber* – Fibra monomodo.

[14] DS-1 – Canal TDM (*Time Division Multiplexing*) digital.

[15] DS-3 – Canal TDM digital.

[16] E1 – *European Digital Signal 1* – Padrão europeu para interface física digital.

[17] E3 – *European Digital Signal 3* – Pode suportar simultaneamente 16 circuitos E1.

As características do meio físico de uma interface baseada em células são idênticas àquelas definidas para a interface baseada em SDH (*Synchronous Digital Hierarchy*) [20] e SONET, e são descritas em detalhe na Recomendação I.432.

2.3.2.4.2 - Características de Convergência de Transmissão

Nesta interface, as células ATM são transportadas continuamente, sem qualquer estrutura de *frames*. A taxa de *bits* disponível para as células de usuário, sinalização e OAM é de 149.76 Mbps quando se utiliza um sistema de transmissão a 155.52 Mbps, e 599.04 Mbps quando se utiliza um sistema de transmissão a 622.08 Mbps. Estas taxas são idênticas às taxas disponíveis para a transmissão de células de usuário, sinalização e OAM na interface SDH. Assim, na interface baseada em células são utilizadas células de adaptação de taxa, chamadas células da camada física (PL *Cells – Physical Layer Cells*).

As células PL são geradas e interpretadas na camada física. O espaçamento máximo entre duas células PL é de 26 células ATM, ou seja, após 26 células da camada ATM serem transmitidas, necessariamente uma célula PL deve ser inserida no fluxo. As células PL podem ser células vazias ou células de OAM da camada física (PLOAM – *Physical Layer* OAM *Cells*). As células vazias apenas executam a função de adaptação de taxa, enquanto as células PLOAM carregam informações de OAM relativas à camada física. As células PL são identificadas por um cabeçalho específico, como se pode ver na Tabela 2.3.

Tipo de Célula	Primeiro Octeto	Segundo Octeto	Terceiro Octeto	Quarto Octeto
Células Vazias	00000000	00000000	00000000	00000001
Células PLOAM	00000000	00000000	00000000	00001001

Tabela 2.3 – Valores predefinidos para o cabeçalho das células na camada física

2.3.2.4.3 - Implementação de OAM

Maiores detalhes sobre as funções de OAM serão apresentadas na seção 2.3.6.3 - Implementação de OAM na Camada Física para a Interface Baseada em Células.

2.3.3 - Camada ATM

A camada acima da camada física na arquitetura das redes ATM é chamada de camada ATM. A camada ATM é independente da camada física e da camada de adaptação, e é responsável por um grande número de funções envolvendo o cabeçalho das células, com exceção do campo HEC que é manejado pela camada física.

Os cabeçalhos das células são gerados e extraídos pela camada ATM. Na estação final de ingresso na rede, a camada ATM insere os campos de cabeçalho, incluindo os identificadores virtuais VPI e VCI, em cada AAL-PDU proveniente da AAL. Na estação de egresso, o cabeçalho das células ATM é removido e as ATM-SDUs são passadas para a AAL.

Outras funções executadas pela camada ATM são a multiplexação e a demultiplexação de células. Em uma estação de ingresso na rede ATM, as células provenientes de vários caminhos virtuais e conexões virtuais são combinadas em um fluxo descontínuo que é passado para a camada física para a transmissão. A função de multiplexação permite que a integração do fluxo de células de várias conexões seja multiplexada sobre um único enlace físico. Na estação de egresso, o fluxo de células é demultiplexado em caminhos virtuais ou conexões virtuais específicas, baseado no conteúdo dos campos VPI e VCI do cabeçalho das células.

A camada ATM também executa a translação de VPI e VCI. Esta função é tipicamente executada em chaveadores ou em *cross-connects*. Os campos de VPI e VCI são transladados conforme explicado no item 2.2.4 - Roteamento de Células.

A camada ATM também deve ter a habilidade para discriminar células tendo como base as informações contidas no cabeçalho. Tal habilidade é necessária porque algumas funções e estados da rede devem ser acionados através da interpretação de um ou mais campos do cabeçalho das células ATM. Por exemplo, o campo PT habilita a camada ATM a discriminar entre células que contém informações de usuário e células que contém outras informações. Outro exemplo é a utilização dos campos VPI e VCI para carregar informações de controle e OAM.

A Tabela 2.4 mostra alguns valores que são atribuídos aos campos VPI e VCI e o seu significado junto à camada ATM.

Função	VPI	VCI
Identificação de Células Vazias	0	0
Metasinalização	0	1
Fluxo de OAM F4 de segmento	0	3
Fluxo de OAM F4 fim a fim	0	4
Sinalização UNI	0	5
Identificação de Células SMDS[18]	0	15
Identificação de Células ILMI[19]	0	16

Tabela 2.4 – Valores Predefinidos de VPI e VCI

[18] SMDS – *Switched Multimegabit Data Service*

[19] ILMI – *Interim Local Management Interface*

Outra função muito importante desempenhada pela camada ATM é o gerenciamento de tráfego. O objetivo desta função é suportar a QoS de cada conexão da rede durante o seu tempo de duração e proteger os usuários finais e a rede de congestionamento.

Uma discussão mais detalhada das funções de controle de fluxo e de congestionamento, especificadas para a camada ATM, será realizada no item 2.3.7 - Gerenciamento de Tráfego.

2.3.4 - Camada de Adaptação ATM

A camada de adaptação ATM auxilia a camada ATM no suporte às camadas superiores da rede. A AAL executa funções solicitadas pelos planos de usuário, controle e gerenciamento, e funciona como uma camada de ligação entre os serviços oferecidos pela camada ATM e os serviços solicitados pelas camadas superiores da rede. Portanto, a AAL atua na adaptação do fluxo de informações das camadas superiores para a camada ATM e vice-versa.

A fim de atender diferentes tipos de serviço, a AAL suporta múltiplos protocolos. A Recomendação I.362 do ITU-T [21] classifica os serviços a serem atendidos pela AAL e define protocolos específicos para atender cada uma destas classes de serviço.

A Tabela 2.5 [5] mostra a classificação de serviços para a AAL e os protocolos designados para atender cada classe de serviço.

Aspecto	Classe A	Classe B	Classe C	Classe D	Classe X[20]
Relacão temporal entre fonte e destino	Requerida	Requerida	Não requerida	Não requerida	Definida pelo usuário
Taxa de bits	Constante	Variável	Variável	Variável	Definida pelo usuário
Modo de conexão	Orientado a conexão	Orientado a conexão	Orientado a conexão	Não orientado a conexão	Orientado a conexão
Protocolo	AAL 1	AAL 2	AAL ¾ e AAL 5	AAL ¾ e AAL 5	AAL 0

Tabela 2.5 – Classificação de Serviços para a AAL

A classificação dos serviços da AAL foi feita com o objetivo de minimizar o número de protocolos necessários. Para isto, os seguintes parâmetros foram utilizados:

- Relacão temporal entre fonte e destino (requerida ou não requerida)

[20] A classe X e a AAL 0 foram especificadas pelo ATM *Forum*.

- Taxa de *bits* (constante ou variável)
- Modo de Orientação (Orientado a conexão ou não orientado a conexão)

2.3.4.1 - As Subcamadas da AAL

A camada de adaptação ATM é dividida em duas subcamadas: Subcamada de Convergência (CS – *Convergence Sublayer*) e Subcamada de Segmentação e Remontagem (SAR – *Segmentation and Reassembly Sublayer*). Segundo a Recomendação I.362, estas subcamadas podem ser divididas novamente. Este é o caso nos protocolos AAL ¾ e AAL 5. Nestes protocolos a Subcamada de Convergência é dividida em Subcamada de Convergência de Serviços Específicos (SSCS – *Service Specific Convergence Sublayer*) e Subcamada de Convergência de Serviços Comuns (CPCS – *Common Part Convergence Sublayer*). A SSCS foi projetada para suportar aspectos específicos de um aplicativo e a CPCS para suportar funções genéricas comuns a mais de um tipo de aplicativo.

A Subcamada de Convergência é dependente do tipo de serviço e executa funções tais como: manipulação da variação de atraso de células (CDV – *Cell Delay Variation*), recuperação de freqüência e correção de erros. Embora cada protocolo AAL tenha suas próprias funções, no geral, a Subcamada de Convergência descreve os serviços e funções necessárias para a conversão entre protocolos ATM e não ATM.

A Subcamada de Segmentação e Remontagem é responsável pela fragmentação das CPCS-SDUs de informação em SAR-PDUs na fonte, e pela remontagem dessas SAR-PDUs em CPCS-PDUs no destino. A SAR acrescenta cabeçalhos e *trailers* nos fragmentos da CPCS-SDU e encaminha as SAR-PDUs de 48 *bytes* para a camada ATM. Cada protocolo AAL possui seu próprio formato de SAR. No destino, cada campo de informação de célula é extraído na camada ATM e convertido para o PDU apropriado.

Algumas das características e funções comuns a todas AALs são as seguintes:

- As AALs são localizadas em equipamentos de usuários finais ATM.
- As AALs são dependentes dos aplicativos de camadas superiores em uso.
- As informações de aplicativos são passadas para a AAL através de um ponto de acesso ao serviço (SAP – *Service Access Point*), no formato de AAL-SDUs, que podem ter até 64 *Kbytes*.

A seguir detalharemos as principais características dos protocolos AAL.

2.3.4.2 - AAL Tipo 1

A AAL 1 suporta o tráfego da classe A. Como já vimos, o tráfego da classe A possui taxa de *bits* constante. Voz e vídeo em tempo real pertencem a esta classe. O termo constante implica que a taxa de *bits* deve ser invariável e sincronizada entre fonte e destino.

Os serviços providos pela AAL 1 são [22]:

- Transferência de informações com a taxa de *bits* da fonte constante e entrega destas informações no destino com a mesma taxa.
- Transferência da informação temporal entre fonte e destino.
- Indicação de informações perdidas ou erradas, que não foram recuperadas pela AAL 1.

2.3.4.3 - AAL Tipo 2

A AAL 2 suporta o tráfego da classe B. Áudio e vídeo de taxa variável pertencem a esta classe. A AAL 2 ainda está em estudo pelos órgãos padronizadores. Pouca coisa foi definida pelo ITU-T para este tipo de AAL.

2.3.4.4 - AAL Tipo ¾

A AAL ¾ suporta o tráfego das classes C ou D. Originalmente, o ITU-T definiu um protocolo AAL 3 para o suporte de tráfego orientado a conexão e um protocolo AAL 4 para o suporte de tráfego não orientado a conexão. Mas, devido a semelhança entre os serviços prestados por estas AALs, o ITU-T acabou juntando-as na AAL ¾.

O tráfego suportado pela AAL ¾ é caraterizado por:

- A existência ou não de uma conexão entre a AAL ¾ fonte e a de destino
- Taxa de *bits* variável
- Nenhuma informação de temporização é passada entre fonte e destino

A AAL ¾ suporta dois modos de serviço: modo de serviço de mensagens (*Message Mode Service*) e modo de serviço de fluxo (*Streaming Service Mode*). O modo de serviço de mensagem é usado para transferir apenas um PDU de informação de um aplicativo, enquanto o modo de serviço de fluxo é usado para transferir um ou mais PDUs de informação em instantes diferentes.

A AAL ¾ também suporta transmissão assegurada e não assegurada. No caso da transmissão assegurada é feita a retransmissão de dados quando erros forem detectados. Já para a transmissão não assegurada, todos os dados, inclusive aqueles onde foram detectados erros, são entregues à AAL de destino, que notifica ao usuário final a presença de erros.

2.3.4.5 - AAL Tipo 5

A AAL 5 suporta os mesmas classes de serviço da AAL ¾, ou seja, as classes C e D. Assim como a AAL ¾, a AAL 5 também suporta dois modos de serviço, bem como transmissão assegurada e não assegurada.

Originalmente chamada de Camada de Adaptação Eficiente Simples (SEAL – *Simple Eficient Adaptation Layer*), a AAL 5 foi projetada para serviços que não requerem um processamento extensivo na AAL, como por exemplo o tráfego de dados IP (*Internet Protocol*) [23]. Assim, a AAL 5 executa um processamento mínimo no nível da camada de adaptação.

Atualmente, a AAL 5 vem sendo a camada de adaptação mais implementada. Algumas das razões que levaram a tal popularidade são [5]:

- Proteção de erros e integridade de PDUs – A AAL ¾ executa a detecção e correcção de erros tanto na subcamada SAR como na CPCS. A AAL5 verifica e corrige erros apenas na subcamada CPCS. Ou seja, a AAL ¾ verifica a integridade de informações no nível de SAR-PDUs e de CPCS-PDUs, enquanto a AAL 5 verifica apenas no nível de CPCS-PDUs. Isto torna a AAL 5 mais "leve" e barata de ser implementada.
- Campo de informações disponível - O campo de informações disponível na AAL 3/4 é de 44 *bytes*, enquanto na AAL 5 é de 48 *bytes*. Levando-se em conta que o tamanho da célula ATM é de 53 *bytes*, esta diferença é significativa em termos de eficiência de transmissão.
- Processamento de cabeçalho – A ausência de cabeçalhos ou *trailers* na subcamada SAR reduz os recursos da AAL 5. Entretanto, mecanismos adequados de proteção de erro tornam a AAL 5 mais atrativa do que a AAL ¾.
- Padronização – Grande parte das especificações produzidas pelo ATM *Forum* estão fundamentadas no uso da AAL 5.

2.3.4.6 - AAL de Sinalização

A AAL de sinalização fornece um meio estruturado e confiável para o transporte de tráfego de sinalização entre dois usuários finais ATM. Como integrante do plano de controle, a *Signaling ATM Adaptation Layer* – SAAL atua como interface entre as funções de controle das camadas superiores e as funções de sinalização ATM.

A SAAL usa os serviços providos pelas subcamadas SAR e CPCS da AAL 5. A SAAL possui ainda a subcamada SSCS, que contém duas funções:

- Função de Coordenação Específica de Serviço (SSCF – *Service Specific Coordination Function*) – Esta função é responsável pelo mapeamento dos aplicativos de camadas superiores para o protocolo SSCOP.
- Protocolo Orientado a Conexão Específico de Serviço (SSCOP – *Service Specific Connection Oriented Protocol*) – O SSCOP é um protocolo orientado a conexão que atua no nível de enlace (segunda camada do Modelo OSI) e fornece um transporte confiável para as mensagens de sinalização. Suporta detecção e correção de erros, seqüenciamento e recuperação seletiva de PDUs.

2.3.5 - Primitivas de Serviço

Como já abordamos anteriormente, primitivas são requisições de serviço passadas entre camadas que contém informações de controle de protocolos e dados de usuários.

Conforme veremos no Capítulo 6, na versão atual do **SimATM** implementamos somente um modelo para a AAL Tipo 5. Por este motivo, a seguir apresentaremos somente as primitivas trocadas entre a AAL 5 e as camadas superiores, internamente à AAL 5, entre a AAL 5 e a camada ATM e entre a camada ATM e a camada física.

2.3.5.1 - Primitivas Trocadas entre a AAL 5 (subcamada CPCS) e as Camadas Superiores

As informações trocadas entre a AAL 5 e as camadas superiores são baseadas nas seguintes primitivas [16]:

- CPCS-UNITDATA *Invoke* – Esta primitiva é enviada pelas camadas superiores a fim de transferir informações contidas na CPCS-SDU e invocar os serviços da subcamada CPCS.

- CPCS-UNITDATA *Signal* – Esta primitiva é enviada pela subcamada CPCS às camadas superiores a fim de sinalizar a chegada de uma CPCS-SDU.

O formato destas primitivas é ilustrado na Figura 2.8.

2.3.5.1.1 - Descrição dos Parâmetros das Primitivas

- *Interface Data* (ID) – Este parâmetro contém a unidade de dados de interface (*Interface Data Unit*) a ser trocada entre as subcamadas CPCS e SSCS. Se a entidade da CPCS está operando no modo de serviço de mensagem, os dados da interface representam uma CPCS-SDU completa. Quando a CPCS opera no modo de serviço de fluxo, os dados da interface não precisam necessariamente representar uma CPCS-SDU completa.
- *More* (M) – No modo de serviço de mensagem, este parâmetro não é usado. No modo de serviço de fluxo este parâmetro especifica se os dados da interface contém o inicio, continuação ou final de uma CPCS-SDU.
- CPCS-*Loss Priority* (CPCS-LP) – Este parâmetro indica a prioridade de perda da CPCS-SDU associada à primitiva, e pode assumir apenas dois valores: um para prioridade alta e outro para prioridade baixa. Este parâmetro é mapeado para o parâmetro SAR-*Loss Priority* da primitiva SAR-UNITDATA *Invoke*, e é preenchido a partir do parâmetro SAR-*Loss Priority* da primitiva SAR-UNITDATA *Indication*.
- CPCS-*Congestion Indication* (CPCS-CI) – Este parâmetro indica se a CPCS-SDU associada à primitiva experimentou algum congestionamento. Este parâmetro é usado para configurar o parâmetro SAR-*Congestion Indication* da primitiva SAR-UNITDATA *Invoke*, e é configurado a partir do parâmetro SAR-*Congestion Indication* da primitiva SAR-UNITDATA *Indication*.
- CPCS-*User-to-User Indication* (CPCS-UU) – Este parâmetro é transportado transparentemente pela CPCS.
- *Reception Status* (RS) – Este parâmetro indica se a CPCS-SDU associada à primitiva pode estar corrompida, e só é usado no modo de operação não assegurado.

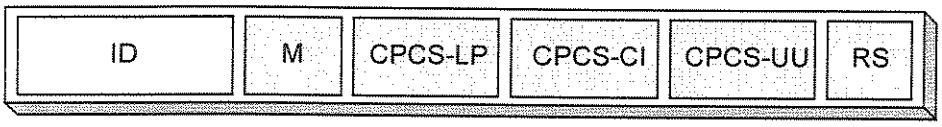


Figura 2.8 – Formato das Primitivas CPCS-UNITDATA

É importante salientar que as primitivas trocadas entre as camadas superiores e a CPCS na verdade são trocadas com a subcamada SSCS. Porém, segundo a Recomendação I.363 [22] a subcamada SSCS pode apenas executar o mapeamento de primitivas entre a CPCS e as camadas superiores. Este é o caso considerado nesta descrição.

2.3.5.2 - Primitivas Trocadas Internamente na AAL 5 entre as Subcamadas CPCS e SAR

As informações trocadas internamente entre as subcamadas da AAL 5 são baseadas nas seguintes primitivas [16]:

- SAR-UNITDATA *Invoke* – Esta primitiva é enviada pela subcamada CPCS a fim de transferir as informações contidas na CPCS-PDU e invocar os serviços da subcamada SAR.
- SAR-UNITDATA *Signal* – Esta primitiva é enviada pela subcamada SAR à subcamada CPCS a fim de sinalizar a chegada de uma CPCS-PDU.

O formato destas primitivas é ilustrado na Figura 2.9.

2.3.5.2.1 - Descrição dos Parâmetros das Primitivas

- *Interface Data* (ID) – Este parâmetro contém a unidade de dados da interface (*Interface Data Unit*) a ser trocada entre as subcamadas SAR e CPCS.
- *More* (M) – Este parâmetro especifica se o dado da interface (ID) comunicado contém o final da SAR-SDU.
- SAR-*Loss Priority* (SAR-LP) – Este parâmetro indica a prioridade de perda para o SAR-ID associado, e pode assumir dois valores: um para prioridade alta e outro para prioridade baixa. Este parâmetro é usado para configurar o parâmetro *Loss Priority* da primitiva ATM-DATA *Request*, e é configurado a partir deste mesmo parâmetro na primitiva ATM-DATA *Indication*.

- SAR-*Congestion Indication* (SAR-CI) – Este parâmetro indica se o associado SAR-ID experimentou algum congestionamento. Este parâmetro é mapeado para o parâmetro *Congestion Indication* da primitiva ATM-DATA *Request*.

Figura 2.9 – Formato das Primitivas SAR-UNITDATA

2.3.5.3 - Primitivas Trocadas entre a Camada ATM e a AAL 5 (subcamada SAR)

As informações trocadas entre a camada ATM e a AAL 5 são baseadas nas seguintes primitivas [16]:

- ATM-DATA *Request* – Esta primitiva é enviada pela camada de adaptação à camada ATM para requisitar a transferência de uma SAR-PDU. Os seguintes parâmetros são usados para preencher os campos CLP e PTI das células ATM: *Loss Priority* e ATM-*user-to*-ATM-*user Indication*, respectivamente.
- ATM-DATA *Indication* – Esta primitiva é enviada pela camada ATM à camada de adaptação para indicar o recebimento de uma ATM-SDU. Na ausência de erros, a mesma ATM-SDU da primitiva ATM-DATA *Request* é devolvida à AAL.

O formato destas primitivas é ilustrado na Figura 2.10.

2.3.5.3.1 - Descrição dos Parâmetros das Primitivas

- ATM-SDU – Este parâmetro contém 48 *bytes* de dados provenientes da camada de adaptação ATM.
- *Loss Priority* (LP) – Este parâmetro indica a importância relativa da requisição de transporte de informações na ATM-SDU, e pode assumir dois valores: um para prioridade alta e outro para prioridade baixa.
- *Congestion Indication* (CI) – Este parâmetro indica que a ATM-SDU recebida passou por um nó congestionado.
- ATM-*user-to*-ATM-*user Indication* (AUU) – Este parâmetro é transportado transparentemente pela camada ATM.

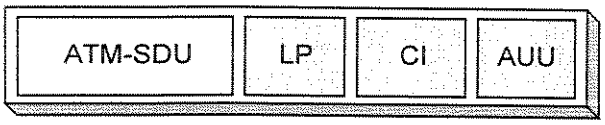


Figura 2.10 – Formato das Primitivas ATM-DATA

2.3.5.4 - Primitivas Trocadas entre a Camada ATM e a Camada Física

As informações trocadas entre a camada ATM e a camada física são baseadas nas seguintes primitivas [16]:

- PHY-DATA *Request* – Esta primitiva é enviada pela camada ATM à camada física a fim de requisitar o transporte de uma célula ATM. Todas as células são trocadas entre a camada ATM e a camada física através do ponto de acesso ao serviço da camada física (PHY-SAP).
- PHY-DATA *Indication* – Esta primitiva é enviada pela camada física à camada ATM a fim de indicar a chegada de uma célula. Na ausência de erros, a PHY-SDU é a mesma que foi enviada pela camada ATM na primitiva PHY-DATA *Request*.

O formato destas primitivas é ilustrado na Figura 2.11.

2.3.5.4.1 - Descrição dos Parâmetros das Primitivas

- PHY-SDU – Este parâmetro contém uma célula ATM.

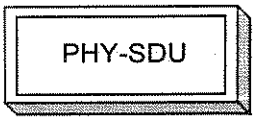
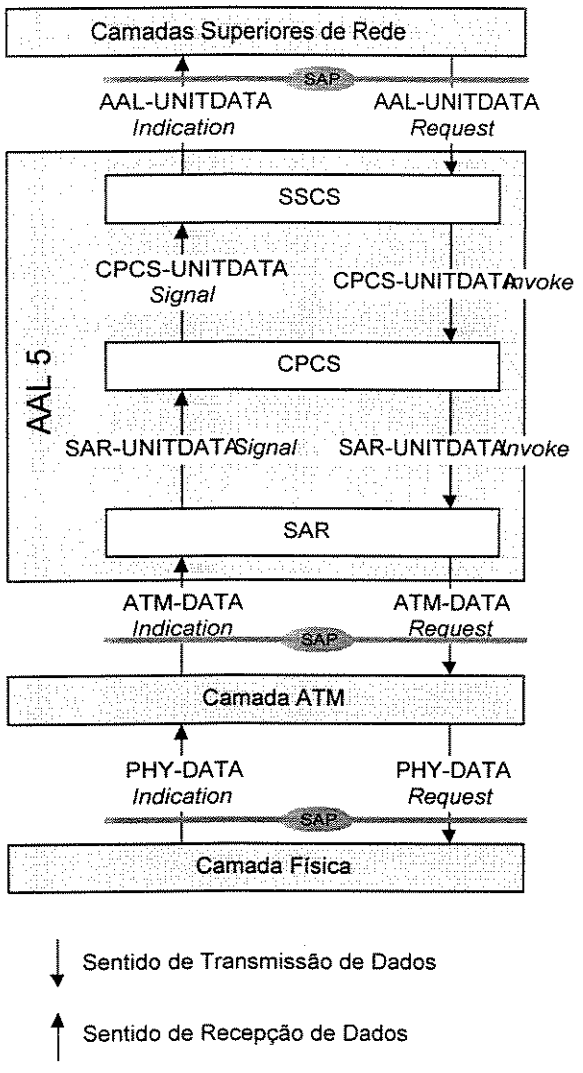


Figura 2.11 – Formato das Primitivas PHY-DATA

2.3.5.5 - Visão Geral da Troca de Primitivas no Plano de Usuário

A Figura 2.12 mostra um resumo das primitivas trocadas entre as camadas no plano de usuário. No sentido de transmissão de dados, as camadas superiores solicitam os serviços de transporte de informações da rede ATM através da primitiva AAL-UNITDATA *Request*, que contém o PDU destas camadas. Se a subcamada SSCS for nula, a primitiva AAL-UNITDATA *Request* será mapeada para a primitiva CPCS-UNITDATA *Invoke* e enviada para a CPCS. Note que, como já vimos anteriormente, somente os protocolos AAL ¾ e 5 especificam uma subcamada SSCS. Então, a AAL-SDU será fragmentada em AAL-PDUs de 48 *bytes* e enviada para a camada ATM através da primitiva ATM-DATA *Request*. A camada

ATM coloca cada ATM-SDU em uma célula e solicita os serviços da camada física através da primitiva PHY-DATA *Request*.



Finalmente, a camada física acomoda as células ATM em um *frame* e transmite até o próximo equipamento da rede.

No sentido de recepção de dados, cada PHY-PDU é retirada do *frame* de transmissão, e enviada para a camada ATM junto à primitiva PHY-DATA *Indication*. A camada ATM então processa o cabeçalho da célula e envia a ATM-PDU para a AAL através da primitiva ATM-DATA *Indication*. A AAL remonta a AAL-SDU original e envia para as camadas superiores através da primitiva AAL-UNITDATA *Indication*.

Figura 2.12 – Troca de primitivas entre camadas no plano de usuário

É importante salientar que o cenário de troca de primitivas apresentado na Figura 2.12 refere-se ao plano de usuário. Existem outras primitivas que são trocadas entre as camadas do plano de usuário e as entidades de gerenciamento de camadas existentes no plano de gerenciamento [13][16]. No plano de controle, as primitivas da camada ATM são trocadas com a camada de adaptação ATM de sinalização, ao invés dos protocolos AAL tipo 1, 2, ¾ e 5.

2.3.6 - Operação, Administração e Manutenção

A rede ATM precisa realizar um conjunto de ações para garantir uma operação apropriada. Estas ações são chamadas de funções de Operação, Administração e Manutenção (OAM - *Operation, Administration and Maintenance*).

A Recomendação I.610 do ITU-T [24] descreve os princípios e funções de OAM executadas pelo gerenciamento de camadas da B-ISDN. Segundo a I.610, as funções de OAM são executadas em cinco níveis hierárquicos, relativos à camada física e à camada ATM. Tais funções resultam nos fluxos de informações bidirecionais F1, F2, F3, F4 e F5, referidos como fluxos de OAM, tal como se pode observar na Figura 2.13. Cada fluxo corresponde a um nível de OAM. Estes fluxos são implementados por células ATM especiais que fluem periodicamente na rede ATM entre seus vários equipamentos. Os fluxos F1, F2 e F3 circulam na camada física, enquanto os fluxos F4 e F5 na camada ATM.

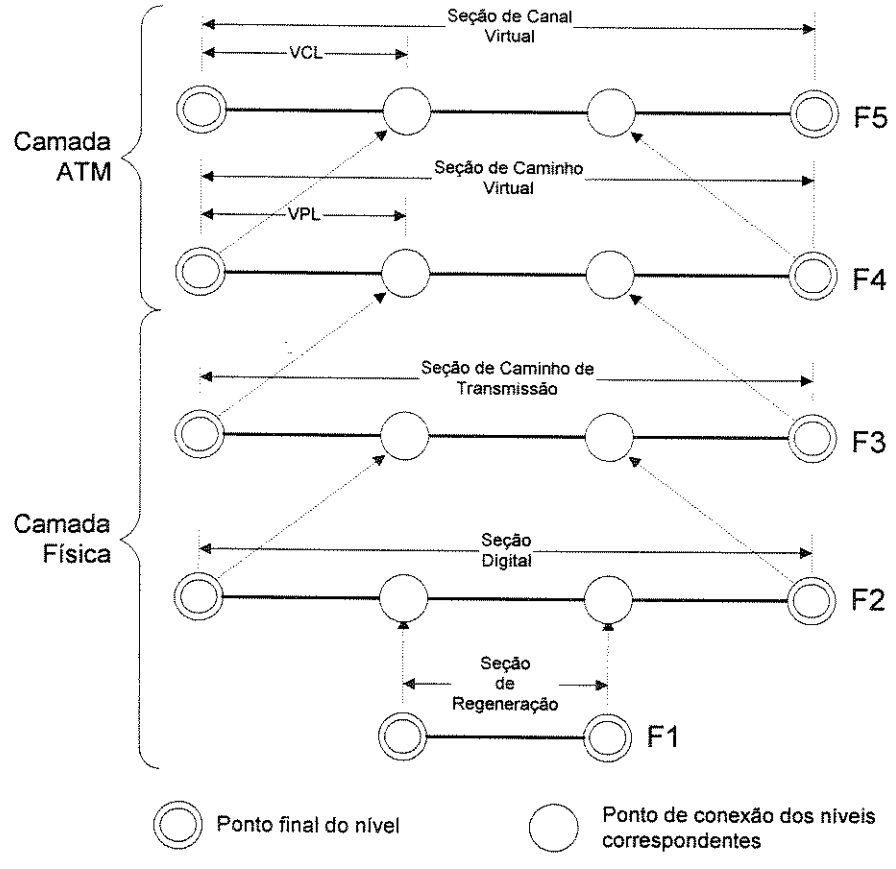


Figura 2.13 – Níveis hierárquicos OAM e seus relacionamentos com a camada ATM e física

Os níveis hierárquicos de OAM são os seguintes [24]:

- Nível de Canal Virtual (*Virtual Channel Level*) – Estende-se entre elementos da rede que executam funções de terminação de conexões de canal virtual.
- Nível de Caminho Virtual (*Virtual Path Level*) – Estende-se entre elementos da rede que executam funções de terminação de conexões de caminho virtual.
- Nível de Caminho de Transmissão (*Transmition Path Level*) – Estende-se entre elementos da rede que fazem a segmentação e remontagem das informações carregadas no sistema de transmissão e associam este com os fluxos de OAM. Funções de delineamento de célula e de controle de erro no cabeçalho (HEC) são requeridas nos pontos finais de cada caminho de transmissão. O caminho de transmissão é conectado através de uma ou mais seções digitais.
- Nível de Seção Digital (*Digital Section Level*) – Estende-se entre pontos finais de seções digitais.
- Nível de Seção de Regeneração (*Regenerator Section Level*) – Uma seção de regeneração é uma porção da seção digital.

As operações de OAM no plano de gerenciamento são divididas em três categorias de serviço de OAM:

- Gerenciamento de Faltas (*Fault Management*) – Nesta categoria de serviço de OAM, células são enviadas para indicar um problema, tal como perda de conexão, uma interface danificada ou um componente defeituoso.
- Gerenciamento de Desempenho (*Performance Management*) – Nesta categoria de serviço de OAM, células são usadas para monitorar e documentar a desempenho de conexões. Estatísticas, tais como células erradas, perdidas ou danificadas, são documentadas através de operações de gerenciamento de performance.
- Ativação/desativação (*Activation/Deactivation*) – Faz a ativação e desativação de células utilizadas para carregar informações de gerenciamento de conexões.

A Tabela 2.6 [24] mostra as funções e serviços suportados pelos fluxos de OAM.

Função	Serviços
Gerenciamento de Faltas	Fiscalização de alarme com AIS[21]
	Fiscalização de alarme com FERF[22]
	Loopback
	Checagem de continuidade
Gerenciamento de Desempenho	Monitoramento direto
	Monitoramento reverso
	Monitoramento/documentação
Ativação/desativação	Monitoramento de desempenho (ativação/desativação)
	Checagem de continuidade (ativação/desativação)

Tabela 2.6 – Funções e serviços de OAM

2.3.6.1 - Funções de OAM na Camada Física

Dois tipos de funções de OAM podem ser distinguidas para a camada física:

- Funções de OAM suportadas apenas pelos fluxos F1, F2 e F3 – Dedicadas a detecção e indicação de estados de indisponibilidade. Requerem transporte de informações de defeito em tempo real na direção dos pontos finais afetados visando a proteção do sistema de transmissão.
- Funções de OAM relacionadas com o gerenciamento do sistema – Dedicadas ao monitoramento e documentação de desempenho ou à localização de equipamentos defeituosos.

2.3.6.2 - Funções de OAM na Camada ATM

Segundo a Recomendação I.610 [24] do ITU-T, as seguintes funções de OAM são previstas para a camada ATM:

- Sinais de Indicação de Alarme (AIS – *Alarm Indication Signals*) – Documentação de indicações de defeito na direção direta.
- Indicação de Defeitos Remotos (RDI – *Remote Defect Indication*) – Documentação de indicações de defeitos remotos na direção reversa.

[21] AIS – *Alarm Indication Signal*

[22] FERF – *Far-end Receive Failure*

- Checagem de Continuidade (*Continuity Check*) – Realizam o monitoramento contínuo de continuidade na rede.
- *Loopback* – Realizam o monitoramento da conectividade sob demanda, a localização de faltas e a verificação de conectividade pré-serviço.
- Monitoramento de Desempenho Direta (*Forward Monitoring*) – Realiza a documentação de estimações de desempenho no sentido direto.
- Monitoramento de Desempenho Reversa (*Backward Monitorig*) – Realiza a documentação de estimações de desempenho no sentido reverso.
- Ativação/Desativação (*Activation/Deactivation*) – Realiza a ativação ou desativação do monitoramento de desempenho e da checagem de continuidade.
- Gerenciamento do Sistema (*System Management*) – A ser utilizada apenas por sistemas finais.

2.3.6.3 - Implementação de OAM na Camada Física para a Interface Baseada em Células

Células da camada física são utilizadas para carregar informações de OAM. A freqüência com que as células de OAM são inseridas no fluxo é determinada pelos requisitos de OAM. Entretanto, segundo a recomendação I.432, não pode haver mais que uma célula PL-OAM (*Physical Layer* OAM *Cell*) a cada 27 células inseridas no fluxo, e menos que uma célula PL-OAM a cada 513 células inseridas no fluxo. Assim, em certos momentos o número de células PL-OAM pode ser aumentado a fim de melhorar a resposta da rede a algum evento. Ainda segundo a recomendação I.432, o fluxo F2 não é usado nesta interface e as funções correspondentes são suportadas pelo fluxo F3, uma vez que não há transmissão de *frames*.

As células PL-OAM possuem um cabeçalho único que permite a identificação adequada na camada física do receptor. Estes padrões de cabeçalho são mostrados na Tabela 2.7.

Fluxo	Primeiro Octeto	Segundo Octeto	Terceiro Octeto	Quarto Octeto	Quinto Octeto
F1	00000000	00000000	00000000	00000011	01011100
F3	00000000	00000000	00000000	00001001	01101010

Tabela 2.7 – Padrão de cabeçalho para a identificação de células PL-OAM

Como já discutimos anteriormente, as células PL-OAM não são passadas para a camada ATM e, portanto, os campos apresentados na Tabela 2.7 tem significado apenas na camada física.

As informações carregadas nos 48 *bytes* de dados das células PL-OAM são descritas na Recomendação I.432.

2.3.7 - Gerenciamento de Tráfego

O controle de tráfego em uma rede ATM está fundamentalmente relacionado com a habilidade da rede prover qualidade de serviço (QoS - *Quality of Service*) diferenciada para cada aplicativo. Uma regra primária para o gerenciamento de tráfego é proteger a rede e os sistemas finais de congestionamento, permitindo o alcance de seus objetivos de desempenho. Uma regra adicional é promover o uso eficiente dos recursos da rede.

A especificação de Gerenciamento de Tráfego 4.0 (TM 4.0 – *Traffic Management* 4.0) [25] do ATM *Forum* define uma série de procedimentos e parâmetros relacionados com o gerenciamento de tráfego e QoS em redes ATM. Cinco categorias de serviço são definidas. Para cada categoria é estabelecido um conjunto de parâmetros que permite descrever o tráfego apresentado à rede e a qualidade de serviço esperada da rede. Esta especificação estende alguns tópicos apresentados nas Recomendações I.371 [26] e I.356 [27] do ITU-T.

2.3.7.1 - Controle de Tráfego e Controle de Congestionamento

Congestionamento pode ser definido como uma condição que ocorre na camada ATM dos equipamentos da rede, tal que a rede não consegue atender a objetivos de desempenho previamente negociados. Em contrapartida, controle de tráfego pode ser definido como um conjunto de ações tomadas pela rede para evitar o congestionamento.

2.3.7.2 - Qualidade de Serviço na Camada ATM

A Qualidade de Serviço da camada ATM é determinada a partir de um conjunto de parâmetros que caracterizam a desempenho de uma conexão no nível da camada ATM. Estes parâmetros, conhecidos com parâmetros de QoS, quantificam a desempenho de extremo a extremo na camada ATM. Alguns destes parâmetros podem ser negociados entre os sistemas finais ATM e a rede.

2.3.7.3 - Parâmetros de QoS

Os seguintes parâmetros de QoS podem ser negociados na UNI [25]:

- Máximo Atraso de Transferência de Célula (*maxCTD – Maximum Cell Transfer Delay*) – Determina o máximo atraso de transferência de célula para uma dada conexão.
- Variação de Atraso de Célula Pico a Pico (*peak-to-peak CDV – Peak-to-peak Cell Delay Variation*) – Diferença entre o melhor e o pior caso de variação de atraso de célula experimentado em uma conexão.
- Razão de Células Perdidas (CLR – *Cell Loss Ratio*) – Razão da soma das células perdidas sobre o total de células transmitidas.

Os seguintes parâmetros de QoS não podem ser negociados na UNI [25]:

- Razão de Células Erradas (CER – *Cell Error Ratio*) – Razão da soma das células erradas pela soma das células transmitidas com sucesso mais a soma das células erradas.
- Razão de Blocos de Células Severamente Erradas (SECBR – *Severely Errored Cell Block Ratio*) – Razão do número de blocos de células severamente errados sobre o número total de blocos de células transmitidos. Um bloco de células é uma seqüência de N células transmitidas sobre uma mesma conexão.
- Razão de Células "Mal Inseridas" (CMR – *Cell Misinsertion Rate*) – Razão do número de células "mal inseridas" sobre um dado intervalo de tempo.

2.3.7.4 - Contrato de Tráfego

Como já vimos anteriormente, um usuário final ATM requisita uma conexão através de protocolos de sinalização (SVC) ou através de assinatura (PVC). Junto com esta requisição de conexão, é inserido um descritor de tráfego e alguns parâmetros de QoS, a fim de caracterizar a quantidade e a qualidade do tráfego que será transmitido em uma determinada conexão. A rede ATM usa estas informações para estabelecer tal conexão e policiar o tráfego que será transmitido pela rede. Este acordo entre os usuários finais ATM e a rede é chamado de contrato de tráfego.

2.3.7.5 - Parâmetros de Tráfego

Os parâmetros de tráfego descrevem as características de tráfego de uma fonte e podem ser qualitativos ou quantitativos.

A especificação TM 4.0 define os seguintes parâmetros de tráfego:

- Taxa de Pico de Células (PCR – *Peak Cell Rate*) – Especifica um limite superior de taxa para o tráfego submetido a uma conexão.
- Taxa Sustentável de Células (SCR – *Sustenaible Cell Rate*) – Especifica um limite superior na taxa média de células submetida a uma conexão.
- Tamanho Máximo de Surto (MBS – *Maximum Burst Size*) – Especifica o número máximo de células que podem ser transmitidas à taxa de pico de células (PCR).
- Taxa de Células Mínima (MCR – *Minimum Cell Rate*) – Especifica uma taxa mínima para transmissão de células em uma conexão.
- Tolerância à Variação de Atraso de Célula (CDVT – *Cell Delay Variation Tolerance*) – Especifica um valor aceitável de variação de atraso de célula (CDV – *Cell Delay Variation*) (*jitter*).

2.3.7.6 - Categorias de Serviços

As categorias de serviço fornecidas pela camada ATM relacionam as características de tráfego e os requerimentos de QoS, com o comportamento da rede. Em geral, funções tal como roteamento, controle de admissão de conexões (CAC - *Connection Admission Control)* e alocação de recursos são estruturadas de forma diferenciada para cada categoria de serviço. A arquitetura de serviços especificada pelo ATM *Forum* consiste das seguintes categorias de serviços [25]:

- Taxa de *Bits* Constante (CBR – *Constant Bit Rate*) – A categoria de serviço CBR é usada para atender conexões que requerem uma quantidade estática de largura de faixa. Tal largura de faixa é caracterizada pelo parâmetro PCR. A categoria CBR suporta aplicativos que operam em tempo real, ou seja, que requerem atraso e variação de atraso rigidamente limitados, como por exemplo aplicativos de voz, vídeo e de emulação de circuitos. Células que sofrerem atrasos maiores do que aqueles especificados no parâmetro *maxCTD* serão consideradas pelos aplicativos como de baixa importância.

- Taxa de *Bits* Variável em Tempo Real (rt-VBR – *Real-Time Variable Bit Rate*) – A categoria de serviço rt-VBR suporta aplicativos que operam em tempo real e que possuem tráfego surtuoso, e é caracterizada em termos dos parâmetros PCR, SCR e MBS. Nesta categoria, as células que sofrerem atrasos maiores do que aqueles especificados no parâmetro *maxCTD* também serão consideradas pelos aplicativos como de baixa importância.
- Taxa de *Bits* Variável (nrt-VBR – *Non-Real-Time Variable Bit Rate*) – A categoria de serviço nrt-VBR suporta aplicativos que possuam características de tráfego surtuoso e que não operam em tempo real, ou seja, que não requerem atraso e variação de atraso rigidamente limitados. É caracterizada em termos dos parâmetros PCR, SCR e MBS. Nenhum limite de atraso é associado a esta categoria.
- Taxa de *Bits* Não Especificada (UBR – *Unspecified Bit Rate*) – A categoria de serviços UBR também suporta aplicativos que não operam em tempo real, tal como aplicativos de comunicação de dados entre computadores. A categoria UBR não especifica garantias de serviço para o seu tráfego.
- Taxa de *Bits* Disponível (ABR – *Available Bit Rate*) – ABR é uma categoria de serviço cujas características de transmissão negociadas com a rede podem ser modificadas após o estabelecimento de uma conexão. Para isto, um mecanismo de controle de fluxo (ABR *Flow Control*) foi especificado pelo ATM *Forum*. Este mecanismo suporta vários tipos de controle de taxa de entrada, que atuam em resposta a mudanças nas características de transmissão da rede. Esta realimentação é feita através de células específicas de controle, chamadas células de gerenciamento de recursos (RM *Cells* – *Resource Management Cells*). A categoria ABR não foi planejada para suportar serviços em tempo real. Na fase de estabelecimento de uma conexão ABR, o sistema final deve especificar à rede a máxima largura de faixa requerida e a mínima largura de faixa a ser usada. Estas especificações correspondem aos parâmetros PCR e MCR, respectivamente. A largura de faixa disponível para esta conexão pode variar, mas não pode ser menor que aquela especificada através do parâmetro MCR.

A Tabela 2.8 [25] sumariza os atributos (parâmetros de QoS e de tráfego) associados com cada categoria de serviço.

Atributos	Categorias de Serviço para a Camada ATM				
Parâmetros de QoS	**CBR**	**rt-VBR**	**nrt-VBR**	**UBR**	**ABR**
peak-to-peak CDV	especificado			não especificado	
MaxCTD	especificado			não especificado	
CLR	especificado			não especificado	especificado[23]
Parâmetros de Tráfego					
PCR e CDVT	especificado			especificado[24]	especificado[25]
SCR, MBS e CDVT	não aplicável[26]	especificado		não aplicável	
MCR	não aplicável				especificado

Tabela 2.8 – Atributos das categorias de serviço ATM

2.3.7.7 - Negociação de Parâmetros de QoS

Os mecanismos de negociação de QoS entre os sistemas finais e a rede são definidos nas especificações do ATM *Forum*: UNI 4.0 – UNI *Signaling* 4.0 [28] e PNNI 1.0 – *Private Network-to-Network Interface* [29]. Basicamente a negociação da QoS para uma conexão é feita utilizando-se os protocolos de sinalização da interface usuário-rede (UNI) e protocolos de sinalização da interface rede-rede (NNI - *Network-Network Interface*).

2.3.7.8 - Medição dos Parâmetros de QoS

Uma das maneiras possíveis de medição dos parâmetros de QoS é baseada nos fluxos OAM de monitoramento de desempenho que são inseridos junto ao fluxo de células do usuário, conforme já abordamos anteriormente. Maiores detalhes de como são medidos os parâmetros de QoS podem ser encontrados na especificação TM 4.0 [25] e na Recomendação I.356 do ITU-T [27].

2.3.7.9 - Funções e Procedimentos para Gerenciamento de Tráfego

As funções e procedimentos de gerenciamento de tráfego preocupam-se em reagir a situação de congestionamento na rede. O seguinte conjunto de funções de controle de tráfego e congestionamento é especificado na TM 4.0 [25]:

---

[23] A CLR é pequena para fontes que ajustam o fluxo de células de acordo com a situação da rede. Qualquer valor especificado para este parâmetro é específico para cada rede.

[24] Pode não estar sujeito aos procedimentos de CAC e UPC.

[25] Representa o valor máximo que uma fonte ABR pode transmitir.

[26] Este parâmetro não faz sentido para esta categoria de serviço.

- Controle de Admissão de Conexão (CAC – *Connection Admission Control*) – É definido como um conjunto de ações tomadas pela rede durante a fase de estabelecimento de conexão, a fim de determinar se uma requisição de conexão pode ou não ser aceita. Uma revisão dos esquemas CAC propostos para redes ATM pode ser encontrado em [30].
- Controle de Parâmetros Usados (UPC – *Usage Parameter Control*) – É definido como um conjunto de ações tomadas pela rede para monitorar e controlar tráfego, em termos de tráfego oferecido e validação de conexões ATM nos usuários finais da rede. O principal propósito é proteger os recursos da rede de comportamentos ilícitos e não intencionais, os quais podem afetar a QoS de conexões já estabelecidas.
- Controle de Prioridade de Perda de Célula (CLP *Control* – *Cell Loss Priority Control*) – Se um congestionamento ocorrer na rede, células marcadas com uma prioridade baixa podem ser descartadas.
- Descrição de Tráfego (*Traffic Shapping*) – Mecanismos de descrição de tráfego podem ser usados para modificar características de tráfego previamente negociadas com a rede.
- Gerenciamento de Recursos da Rede (NRM – *Network Resource Management*) – O gerenciamento apropriado e efetivo de caminhos virtuais pode ser utilizado para maximizar a alocação de recursos na rede e reduzir as chances de congestionamento. Todas as funções de gerenciamento de tráfego podem ser executados no nível de VP melhor do que no nível de VC. Isto pode simplificar muitas das funções de gerenciamento de tráfego e otimizar os recursos da rede. Exemplos da utilização de caminhos virtuais para o gerenciamento de recursos podem ser encontrados em [31] e [32].
- Descarte de *Frame* (*Frame Discard*) – O ATM controla congestionamentos descartando células. A função de descarte de *frame* permite que sejam descartadas somente as células que fazem parte de uma única AAL-PDU, minimizando assim os efeitos causados por um congestionamento.
- Controle de Fluxo ABR (ABR *Flow Control*) – Um protocolo de controle de fluxo ABR é usado para adaptativamente compartilhar a largura de faixa disponível entre

os usuários participantes de conexões ABR. Uma visão geral da evolução do controle de fluxo ABR pode ser encontrada em [33].

2.3.8 - Sinalização ATM

Sinalização é a técnica usada pelos usuários finais ATM e pela rede para estabelecer, manter e terminar conexões chaveadas (SVCs).

Os protocolos de sinalização ATM variam de acordo com o tipo de enlace empregado. A sinalização UNI é utilizada entre um usuário final ATM e um chaveador, na interface UNI. A sinalização NNI é utilizada em enlaces NNI, ou seja, entre chaveadores.

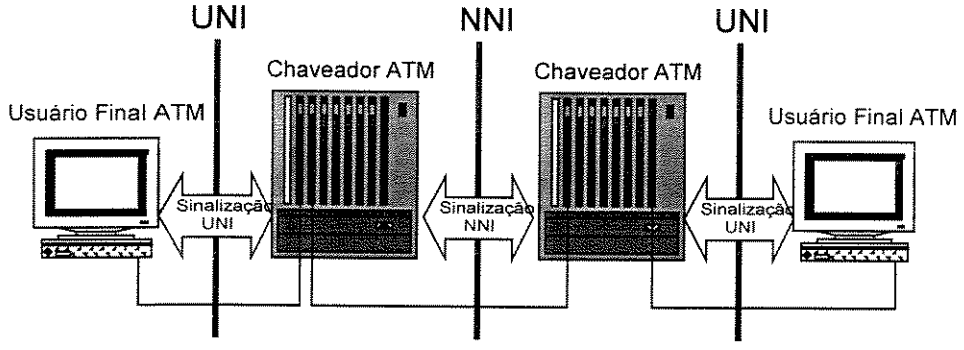


Figura 2.14 – Sinalização UNI e NNI

A sinalização UNI é descrita na especificação de sinalização UNI 4.0 [28] do ATM *Forum*. As requisições de conexão UNI são carregadas através da rede com VPI = 0 e VCI = 5. A sinalização NNI ainda está sendo estudada pelo ATM *Forum* [34].

Basicamente, a sinalização ATM funciona da seguinte forma: uma requisição de conexão de uma estação fonte é propagada através da rede, estabelecendo conexões, até alcançar a estação de destino. O roteamento das requisições de conexão e, portanto, qualquer fluxo subsequente de dados, é realizado por um protocolo de roteamento. Tal protocolo encaminha a requisição de conexão baseado no endereço de destino, nos parâmetros de tráfego e de QoS requisitados pelo estação fonte através do contrato de tráfego. A estação de destino pode escolher aceitar ou rejeitar a conexão.

2.3.9 - Roteamento

Como vimos anteriormente, a utilização de conexões SVCs implica no uso de protocolos de sinalização. Para encaminhar as requisições de conexão destes protocolos entre chaveadores ATM, um protocolo de roteamento NNI é necessário.

O ATM *Forum* especificou um protocolo de roteamento para redes ATM privadas. Trata-se do protocolo P-NNI – *Private Network-to-Network Interface* [35]. O P-NNI consiste de dois componentes: o primeiro é um protocolo de sinalização P-NNI usado para tratar requisições de conexão dentro da rede, entre interfaces UNI fonte e destino. A requisição de sinalização UNI é mapeada em sinalização NNI no chaveador de ingresso e remapeada em sinalização UNI no chaveador de egresso. O segundo componente do protocolo P-NNI é um protocolo de roteamento de circuito virtual, usado para rotear as requisições de sinalização dentro da rede ATM.

2.3.10 - Gerenciamento de Redes ATM

O gerenciamento de redes ATM [36][37] implica no monitoramento e controle de todos os seus componentes. O ATM *Forum* definiu uma interface de gerenciamento local provisória (ILMI – *Interim Local Management Interface*) ILMI 4.0 [29] que é baseada no protocolo SNMP – *Single Network Management Protocol* [38]. A interface ILMI fornece um conjunto provisório de funções de gerenciamento. Estas funções permitem o gerenciamento de parâmetros de estado, configuração e controle, presentes na UNI.

2.4 - Referências Bibliográficas

[1] William Stallings, "ISDN *and broadband* ISDN *with frame relay and* ATM", *Third Edition, Prentice-Hall*, 1995.

[2] ITU-T *Recommendation* I.113, "*Vocabulary of Terms for Broadband Aspects of* ISDN", *Geneva*, 1991.

[3] ITU-T *Recommendation* I.121, "*Broadband Aspects of* ISDN", *Geneva*, 1991.

[4] ITU-T *Recommendation* I.211, "B-ISDN *Service Aspects*", *Geneva*, 1991.

[5] George C. Sackett, Christopher Metz, "ATM *and Multiprotocol Networking*", *McGraw Hill, January* 1997.

[6] Martin De Prycker, "*Asynchronous Transfer Mode*: *Solutions for Broadband* ISDN", *Prentice Hall, Third Edition*, 1995.

[7] Anthony Alles, "*The Next-Generation* ATM *Switch*: *From Testbeds to Production Networks*", *White Paper, Cisco Systems, Inc.*, 1996.

[8] *The* ATM *Forum*, "LAN *Emulation over* ATM", *Version* 1.0, *January* 1995.

[9] *The* ATM *Forum*, "*Multi-Protocol Over* ATM *Specification*", *Version* 1.0, *July* 1997.

[10] http://www.atmforum.com/

[11] Brian Dorling , Jaap Burger, Daniel Freedman, Chris Metz, "*Internetworking over* ATM: *An Introduction*", *Prentice Hall, December* 1996.

[12] Daniel Minolli, Anthony Alles, "LAN, ATM, *and LAN emulation technologies*", *Artech House*, 1996.

[13] Uyless D. Black, "ATM: *Foundation for Broadband Networks*", *Prentice-Hall*, 1995.

[14] Luiz Fernando G. Soares, Guido Lemos, Sérgio Colcher, "Redes de Computadores: das LANs, MANs, WANs às redes ATM", Editora Campus, 1995.

[15] Jean-Yves Le Boudec, "*The Asynchronous Transfer Mode: a tutorial*", *Computer Networks and* ISDN *Systems* 24, pp. 279-309, 1992.

[16] ITU-T *Recommendation* I.361, "B-ISDN ATM *Layer Specification*", *Helsinki*, 1993.

[17] Lars Staalhagen, "A *Comparison Between the* OSI *Reference Model and the* B-ISDN *Protocol Reference Model*", IEEE *Network Magazine*, vol. 10, no. 1, pp. 24-33, *January/February* 1996.

[18] http://www.iso.ch/

[19] ITU-T *Recommendation* I.321, "B-ISDN *Protocol Reference Model and Its Application*", *Geneva*, 1991.

[20] ITU-T *Recommendation* I.432, "B-ISDN *User-Network Interface – Physical Layer Specification*", *Helsinky*, 1993.

[21] ITU-T *Recommendation* I.362, "B-ISDN ATM *Adaptation Layer* (AAL) *Function Description*", *Helsinky*, 1993.

[22] ITU-T *Recommendation* I.363, "B-ISDN ATM *Adaptation Layer* (AAL) *Specification*", *Helsinky*, 1993.

[23] Douglas Comer, "*Interneworking with* TCP/IP", *Third Edition, Prentice-Hall*, 1995.

[24] ITU-T *Recommendation* I.610, "B-ISDN *Operation and Maintenance Principles and Functions*", *Helsinky*, 1995.

[25] *The* ATM *Forum*, "*Traffic Management Specification*", *Version* 4.0, *April* 1996.

[26] ITU-T *Recommendation* I.371, "*Traffic Control and Congestion Control in* B-ISDN", *Geneva*, 1995.

[27] ITU-T *Recomendation* I.356, "B-ISDN ATM *Layer Cell Transfer Performance*", *Geneva*, 1993.

[28] *The* ATM *Forum*, "ATM *User-Network Interface* (UNI) *Signalling Specification*", *Version* 4.0, *July* 1996.

[29] *The* ATM *Forum*, "*Integrated Local Management Interface* (ILMI) *Specification*", *Version* 4.0, *September* 1996.

[30] Harry G. Perros, Khaled M. Elsayed, "*Call Admission Control Schemes: A Review*", IEEE *Communications Magazine*, vol. 34, no. 11, pp. 82-91, *November* 1996.

[31] V. J. Friesen, J.J. Harms, and J. W. Wong, "*Resource Management with Virtual Paths in* ATM *Networks*", IEEE *Network Magazine*, vol. 10, no. 5, pp. 10-20, *September/October* 1996.

[32] Thomas M. Chen, Steve S. Liu, David Wang, Vijay K. Samalam, Michael J. Procanik, and Dinyar Kavouspour, "*Monitoring and Control of* ATM *Networks Using Special Cells*", IEEE *Network Magazine*, vol. 10, no. 5, pp. 28-38, *September/October* 1996.

[33] Kerry W. Fendick, "*Evolution of Controls for the Available Bit Rate Service*", IEEE *Communications Magazine*, vol. 34, no. 11, pp. 35-39, *November* 1996.

[34] Anthony Alles, "ATM *Internetworking*", *White Paper*, *Cisco Systems*, 1995.

[35] *The* ATM *Forum*, "*Private Network-Network Interface Specification*", *Version* 1.0, *March* 1996.

[36] Mehmet Toy, "*Network Management of* ATM *Networks: A Practical View*", ICT'96, pp. 19-24, *April* 1996.

[37] Stephen C. Farkouh, "*Managing* ATM-*based Broadband Networks*", IEEE *Communications Magazine*, pp. 82-86, *May* 1993.

[38] J. Case, M. Fedor, M. Schoffstall, J. Davin, "*A Simple Network Management Protocol* (SNMP)", IETF *Request for Comments* 1157, *May* 1990.

Capítulo 3

Padronização de Equipamentos ATM

O ITU-T recentemente aprovou uma série completa de padronizações sobre operação funcional de equipamentos ATM e gerenciamento de elementos de rede ATM. O objetivo destas recomendações, segundo o ITU-T, é habilitar a interoperabilidade entre equipamentos ATM, independentemente da forma como são implementados.

A Recomendação I.731 [1] descreve a arquitetura funcional geral e as características de um elemento de rede ATM, em termos de blocos funcionais específicos, derivados do modelo de referência da B-ISDN. Também especifica os objetivos de desempenho a serem alcançados por equipamentos ATM.

A Recomendação I.732 [2] descreve detalhadamente as características funcionais de um elemento de rede ATM, especialmente para os aspectos de transferência de dados e gerenciamento de camadas, tomando como base a arquitetura funcional geral descrita na Recomendação I.731. Os elementos funcionais de um equipamento ATM são descritos em termos de uma representação equivalente ao modelo de referência da B-ISDN.

3.1 - Características Gerais de Equipamentos ATM

As características gerais de um equipamento ATM podem ser descritas fazendo-se uma separação funcional dos elementos da rede, através de blocos lógicos unidos por meio de interfaces de comunicação interna. O agrupamento funcional de blocos de acordo com o modelo de referência da B-ISDN, habilita a descrição de qualquer equipamento ATM, no nível de detalhe requerido.

3.1.1 - Arquitetura Funcional Geral de um Elemento de Rede ATM

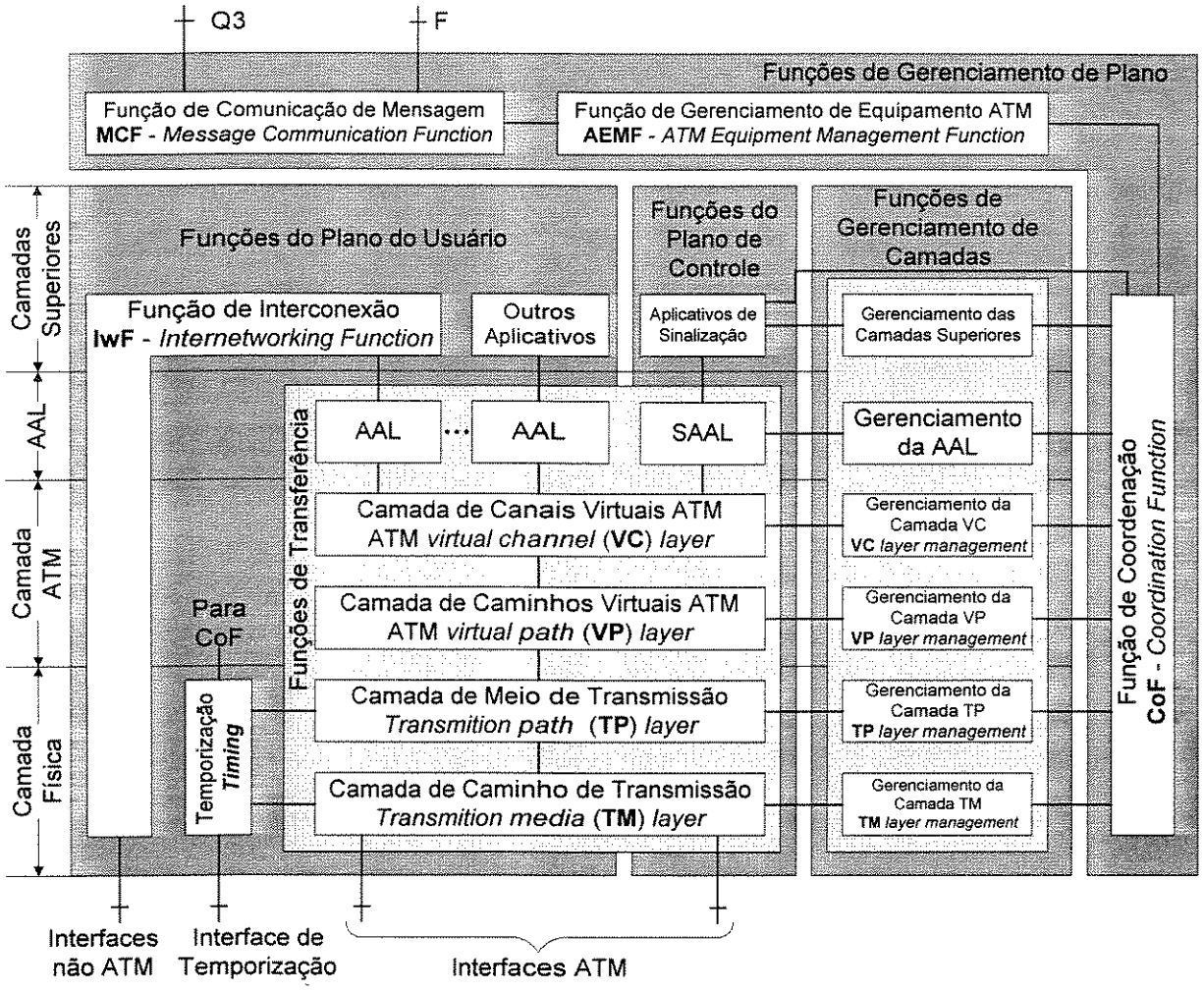


Figura 3.1 – Arquitetura funcional de um elemento de rede ATM

A arquitetura funcional geral de um elemento de rede B-ISDN é mostrada na Figura 3.1. Os seguintes grupos funcionais foram utilizados para descrever este elemento de rede [1][3]:

- Funções de Transferência (*Transfer Functions*) – Relacionam-se com o processamento básico das células ATM de usuário, sinalização, OAM e gerenciamento de recursos.
- Funções de Gerenciamento de Camadas (*Layer Management Functions*) – Preocupam-se com o gerenciamento em tempo real das funções de transferência, e com o processamento das informações trocadas com estas funções.

- Função de Gerenciamento de Equipamento ATM (AEMF – ATM *Equipment Management Function*) – Executa funções relacionadas com as cinco áreas clássicas de gerenciamento: configuração, falhas, desempenho, gerenciamento de usuários e segurança.
- Função de Comunicação de Mensagem (MCF – *Message Communication Function*) – Executa a troca de informações entre entidades AEMF e NMS – *Network Management System* [3].
- Função de Coordenação (CoF – *Coordination Function*) – Executa funções essenciais dentro de em um equipamento ATM, como por exemplo a coordenação entre funções de gerenciamento de camadas e o processamento de requisições por recursos feitos por aplicativos de sinalização, protocolos de gerenciamento de recursos (RM – *Resource Management*) [4] e pela AEMF.
- Aplicativo de Sinalização (*Signaling Application*) – Executam funções do plano de controle necessárias para o gerenciamento sob demanda de conexões chaveadas ATM (SVCs).
- Função de Temporização (*Timing Function*) – Preocupa-se em sincronizar as interfaces dos equipamentos ATM com uma fonte de *clock* externo.
- Função de Interconexão (IwF – *Internetworking Function*) – Fornece as condições necessárias para que haja interoperabilidade entre serviços ATM e outros serviços de rede, tal como *frame-relay* [5] e ISDN.

3.1.2 - Visão Geral das Funções dos Equipamentos

A seguir faremos uma breve descrição das funções que fazem parte da arquitetura funcional de um elemento de rede ATM.

3.1.2.1 - Funções de Transferência

Funções de transferência incluem todas as funções necessárias para o transporte de células de usuário, sinalização, OAM e gerenciamento de recursos (RM). Todos os serviços necessários às camadas superiores da B-ISDN compartilham das mesmas funções de transferência de informações, ou seja, compartilham das mesmas funções necessárias ao transporte de células. Os serviços específicos para a transferência de informações de usuário são atendidos pela camada de adaptação ATM e pelas camadas superiores. As funções de

transferência também são utilizadas para transportar informações relacionadas com a rede, tais como informações de sinalização e gerenciamento.

3.1.2.1.1 - Camada Física

As funções de transferência associadas com a camada física são subdivididas em: Camada de Meio de Transmissão (TML – *Transmition Media Layer*) e Camada de Caminho de Transmissão (TPL – *Transmition Path Layer*).

3.1.2.1.2 - Camada ATM

Segundo a Recomendação I.731, as funções de transferência associadas com a camada ATM são relativas a multiplexação/demultiplexação e *cross-connecting* ou chaveamento de células ATM. Estas funções são divididas em dois blocos funcionais: Camada de Caminhos Virtuais (*Virtual Path Layer*) e Camada de Canais Virtuais (*Virtual Channel Layer*). Cada um destes blocos funcionais consiste de quatro entidades:

- Entidade de Multiplexação (*Multiplexing Entity*) – Inclui funções comuns a todos os enlaces de caminho virtual (VP *Links*) ou a todos os enlaces de canal virtual (VC *Links*).
- Entidade de VP/VC (VP/VC *Entity*) – Inclui todas as funções que são realizadas para cada enlace VP ou VC individualmente, como por exemplo o processamento de fluxo F4/F5 de segmento.
- Entidade de Conexão VP/VC (VP/VC *Connection Entity*) – Executa conexões de enlaces VP/VC entre pontos finais destes enlaces, dentro do mesmo elemento de rede.
- Terminação de Conexão VP/VC (VP/VC *Connection Termination*) – Executa funções relacionadas com os pontos finais de conexões ATM, como por exemplo o processamento dos fluxo F4/F5 fim a fim.

3.1.2.1.3 - Camada de Adaptação ATM

Segundo a Recomendação I.731, os requisitos funcionais da AAL devem estar de acordo com os protocolos AAL anteriormente descritos nos itens 2.3.4.2 – AAL Tipo 1, 2.3.4.3 – AAL Tipo 2, 2.3.4.4 – AAL Tipo ¾ e 2.3.4.5 – AAL Tipo 5.

3.1.2.1.4 - Camada de Adaptação de Sinalização

Segundo a Recomendação I.731, os requisitos funcionais da AAL de sinalização devem estar de acordo com o protocolo SAAL anteriormente descrito no item 2.3.4.6 – AAL de Sinalização.

3.1.2.2 - Funções de Gerenciamento de Camadas

As funções de gerenciamento de camadas preocupam-se em processar informações trocadas com as funções de transferência, e com a gerência em tempo real dessas funções. A Figura 3.2 mostra a troca de informações com as funções de gerenciamento de camadas.

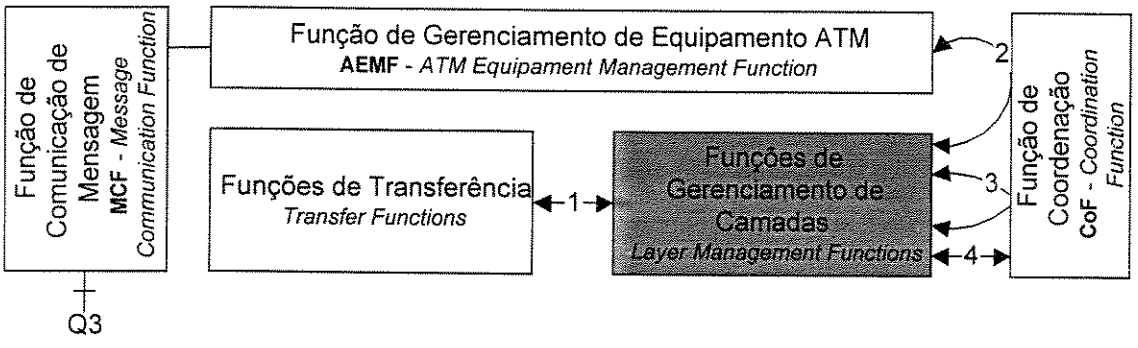


Figura 3.2 – Troca de informações com as funções de gerenciamento de camadas

As funções de gerenciamento executam quatro tipos de troca com o resto do equipamento ATM [3] (observe numeração na figura):

1. Troca de Informações com o Plano de Transferência – As informações de gerenciamento em tempo real associadas com uma dada função de transferência são passadas para a função de gerenciamento de camadas correspondente.
2. Troca de Informações com o AEMF através da CoF – Alguns resultados de gerenciamento podem ser passados para a função de gerenciamento de equipamento ATM (AEMF) para processamento posterior ou para comunicação com uma entidade de gerenciamento de rede.
3. Troca de Informações entre Funções de Gerenciamento de Camadas – Esta troca de informações permite uma descrição ótima dos mecanismos de OAM.
4. Troca de Informações com a Função de Coordenação – A função CAC está localizada na CoF e preocupa-se em estabelecer, modificar ou terminar conexões ATM. Estas operações são feitas usando os recursos fornecidos pelas funções de gerenciamento de camadas.

3.1.2.3 - Função de Gerenciamento de Equipamento ATM (AEMF)

A função de gerenciamento de equipamento ATM atua nas áreas [1]:

- Gerenciamento de Configuração
- Gerenciamento de Faltas
- Gerenciamento de Desempenho
- Gerenciamento de Usuários
- Gerenciamento de Segurança

Uma descrição detalhada da AEMF é feita na Recomendação I.751 [6].

3.1.2.4 - Função de Comunicação de Mensagem (MCF)

A função de comunicação de mensagem executa a troca de mensagens AEMF com a rede de gerenciamento de telecomunicações (TMN – *Telecommunications Management Network*) [7][8].

3.1.2.5 - Função de Coordenação

A função de coordenação é a principal entidade dentro de um equipamento ATM e executa as seguintes funções [3]:

- Coordenação entre diferentes funções de gerenciamento de camadas. A função de coordenação atua como uma entidade geral de comunicação em um equipamento ATM. A CoF pode ser "vista" como um barramento funcional. Desta forma torna possível a comunicação entre funções de diferentes camadas.
- Processamento de requisições de recursos feitos por aplicativos de sinalização, por protocolos de gerenciamento de recursos e pela AEMF (através da interface Q3). A função de coordenação executa o controle de admissão de conexões (CAC – *Connection Admission Control*).

3.1.2.6 - Aplicativos de Sinalização

Na UNI, os procedimentos e mensagens de sinalização devem estar de acordo com a Recomendação Q.2931 [9], enquanto na NNI devem estar de acordo com as Recomendações Q.2761 [10], Q.2762 [11], Q.2763 [12] e Q.2764 [13], que são um conjunto de normas básicas usadas para definir o protocolo B-ISUP – *Broadband* ISDN *User Part*.

3.1.2.7 - Função de Temporização

Preocupa-se com a sincronização das interfaces dos equipamentos, sejam ATM ou não, com uma fonte de *clock* externa.

3.1.2.8 - Funções de Interconexão

A interconexão entre serviços baseados em ATM e outros serviços de rede pode ser suportada por equipamentos ATM em algumas situações. O ITU-T definiu algumas recomendações relacionadas com a interconexão entre redes ATM e outras redes, tais como: ISDN [14], FMBS (*Frame Mode Bearer Service*) [15], etc.

3.2 - Referências Bibliográficas

[1] ITU-T *Recommendation* I.731, "*Types and general characteristics of* ATM *equipment*", *Geneva, March* 1996.

[2] ITU-T *Recommendation* I.732, "*Functional characteristics of* ATM *Equipment*", *Geneva, March* 1996.

[3] Jon Anderson, Patrice Lamy, Laurent Hué, Luc Le Beller, "*Operations Standards for Global* ATM *Networks*: *Network Element View*", IEEE *Communications Magazine*, vol. 34, no. 12, pp. 72-84, *December* 1996.

[4] Brian Dorling , Jaap Burger, Daniel Freedman, Chris Metz, "*Internetworking over* ATM: *An Introduction*", *Prentice Hall, December* 1996.

[5] William Stallings, "ISDN *and broadband* ISDN *with frame relay and* ATM", *Third Edition, Prentice-Hall*, 1995.

[6] ITU-T *Recommendations* I.751, "*Asynchronous Transfer Mode Management of the Network Element View*", *Geneva*, 1996.

[7] R. Glitho, S. Hayes, "*Telecommunications Management Network*: *Vision vs. Reality*", IEEE *Communications Magazine*, vol. 33, no. 3, pp. 47-52, *March* 1995.

[8] D. Sidor, "*Managing Telecommunications Networks Using* TMN *Interface Standards*", IEEE *Communications Magazine*, vol. 33, no. 3, pp. 54-60, *March* 1995.

[9] ITU-T *Recommendation* Q.2931, "*Broadband Integrated Services Digital Network* (B-ISDN) – *Digital Subscriber Line No. 2* (DSS 2) – *User-Network Interface* (UNI) *layer 3 specification for basic call/connection control*", 1995.

[10] ITU-T *Recommendation* Q.2761, "*Broadband Integrated Services Digital Network* (B-ISDN) – *Functional Description of the* B-ISDN *User Part* (B-ISUP) *of Signalling System No. 7*", 1995.

[11] ITU-T *Recommendation* Q.2762, "*Broadband Integrated Services Digital Network* (B-ISDN) – *General functions of messages and signals of the* B-ISDN *User Part* (B-ISUP) *of Signalling System* No. 7", 1995.

[12] ITU-T *Recommendation* Q.2763, "*Broadband Integrated Services Digital Network* (B-ISDN) – *Signalling System No. 7* B-ISDN *User Part* (B-ISUP) – *Formats and codes* ", 1995.

[13] ITU-T *Recommendation* Q.2764, "*Broadband Integrated Services Digital Network* (B-ISDN) – *Signalling System No. 7* B-ISDN *User Part* (B-ISUP) – *Basic call procedures*", 1995.

[14] ITU-T *Recommendation* I.580, "*General arrangements for internetworking betweeen* B-ISDN *and* 64 kbit/s *based* ISDN", 1993.

[15] ITU-T *Recommendation* I.555, "*Frame relaying bearer service internetworking*", 1993.

Capítulo 4

Simulação de Redes ATM

Neste capítulo faremos uma breve apresentação das características gerais das ferramentas de simulação de sistemas de comunicação, destacando as técnicas de simulação utilizadas, uma vez que os simuladores de redes ATM se enquadram neste tipo de ferramenta. Apresentaremos também dois métodos para a simulação de redes ATM: simulação no nível de células e simulação no nível de taxa de células ATM. Uma comparação entre alguns simuladores ATM também será realizada.

4.1 - Introdução

Atualmente, a simulação tem um papel decisivo no projeto, análise e implementação de sistemas de comunicação, principalmente quando estes sistemas são caros e complexos. A simulação de um sistema real pode ser definida como o processo de avaliação numérica de um modelo de simulação, que deve representar o mais fielmente possível o sistema real a ser simulado. As informações resultantes deste processo são utilizadas para estimar variáveis de interesse deste sistema.

4.2 - Características Gerais das Ferramentas de Simulação de Sistemas de Comunicação

4.2.1 - Modelo de Simulação

A geração atual de ferramentas de simulação [1] usa um diagrama de blocos hierárquicos para criar graficamente o modelo de simulação de um sistema de comunicação. Neste diagrama, cada bloco funcional representa um subsistema do modelo de simulação e é conectado a outros blocos a fim de representar o sistema completo. Estes blocos são construídos a partir de bibliotecas de modelos disponibilizadas junto com o *software* de simulação, e possuem parâmetros de simulação e de *design*. Parâmetros de simulação são usados para configurar o estado de um bloco durante a simulação, como por exemplo a amostragem ou não de uma determinada variável estatística, enquanto parâmetros de *design*

são usados para configurar o valor de variáveis específicas de um bloco, desta forma permitindo a modificação do seu comportamento.

Os blocos funcionais são representados graficamente através de ícones, que uma vez conectados, descrevem o modelo de simulação a ser simulado. A Figura 4.1 mostra o exemplo de um diagrama de blocos que constitui o modelo de simulação de um sistema de comunicação óptica para o ambiente de simulação de sistemas SimNT [2][3]. Este ambiente de simulação será apresentado no próximo capítulo (5.5.1 – SimNT).

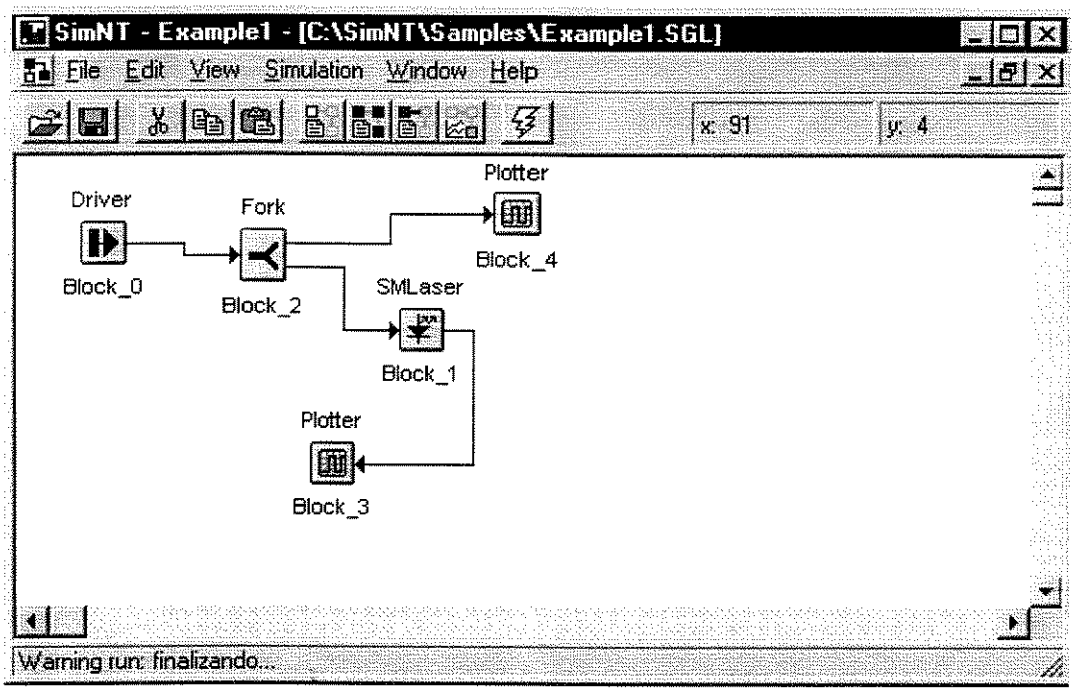


Figura 4.1 – Modelo de simulação de um sistema de comunicação óptica visualizado a partir da interface gráfica do ambiente de simulação de sistemas SimNT.

4.2.2 - Gerência de Simulação

As ferramentas atuais de simulação possuem um gerente de simulação, chamado de *kernel* ou núcleo do simulador, que é responsável por acionar os blocos que fazem parte de um modelo de simulação. O *kernel* permite que um bloco utilize os recursos computacionais disponíveis por um intervalo de tempo dependente da técnica de simulação utilizada. Quando este bloco termina suas tarefas, ele avisa o *kernel*, que por sua vez aciona outros blocos do modelo de simulação.

4.2.3 - Técnicas de Simulação

Atualmente, várias técnicas são utilizadas no desenvolvimento de *softwares* de simulação de sistemas de comunicação. Estas técnicas basicamente diferem na forma como os blocos do sistema simulado são chamados (acionados), e podem ser classificadas em quatro categorias [1]: simulação orientada pelo tempo (*Time-Driven Simulation*), simulação orientada por eventos (*Event-Driven Simulation*), simulação orientada por dados (*Data-Driven Simulation*) e simulação mista (*Mixed-Mode Simulation*), que é uma mistura das técnicas anteriores.

4.2.3.1 - Simulação Orientada pelo Tempo (*Time-Driven Simulation*)

No modo mais simples de implementação desta técnica, cada bloco do modelo de simulação é chamado uma vez a cada intervalo de avanço do tempo de simulação (*simulation clock*). Ou seja, a cada incremento no tempo de simulação, todos os blocos do modelo são chamados, atualizam seus estados internos para corresponder ao tempo atual, independente da existência ou não de alguma tarefa a ser executada. Isto torna esta técnica bastante ineficiente do ponto de vista computacional [1]. O tempo global de simulação é atualizado através de um incremento constante.

4.2.3.2 - Simulação Orientada por Eventos (*Event-Driven Simulation*)

Nesta técnica de simulação os blocos agendam chamadas, que são armazenadas em uma fila com prioridades na forma de eventos (Figura 4.2). Cada evento possui um tempo de execução. O gerente de simulação (*kernel*) ou gerenciador de eventos retira o evento de maior prioridade (menor tempo de execução) que se encontra na fila e chama o bloco para o qual o evento se destina. Vários eventos podem ser agendados para o mesmo tempo de execução. Neste caso, o *kernel* deve executar os eventos na ordem em que eles foram agendados, ou seja, segundo a disciplina FIFO (*First-in first-out*).

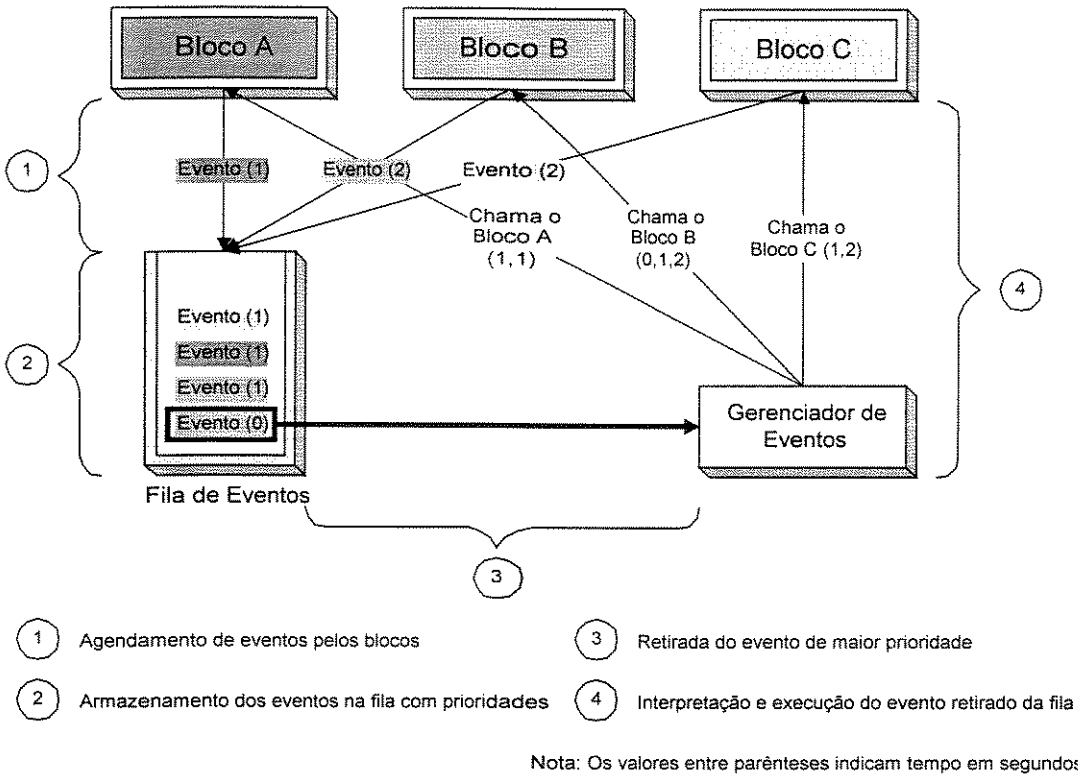


Figura 4.2 – Esquema de armazenamento e execução de eventos

Toda vez que um evento é retirado da fila, o tempo global de simulação é avançado para o tempo deste evento. Tipicamente, apenas alguns blocos são chamados a cada avanço no tempo de simulação. Quando um bloco é chamado, ele executa as suas funções específicas. Ao término do processamento dessas funções, o fluxo da simulação é retornado para o *kernel*, que retira e interpreta outro evento da fila. A simulação prossegue até que não haja mais eventos para serem executados, ou até que o tempo final de simulação seja alcançado.

Na Figura 4.2, os blocos A,B e C agendam eventos na fila com prioridades. O *kernel* retira o evento com maior prioridade da fila, interpreta este evento e realiza a chamada do bloco a que o evento se destina. Supondo que todos os eventos que são mostrados na figura sejam executados, o bloco A será chamado no instante 1 segundo duas vezes, o bloco B nos instantes 0, 1 e 2 segundos, e o bloco C no instante 1 e 2 segundos. Quando um bloco é chamado, ele pode agendar novos eventos na fila para instantes de tempo superiores ou iguais ao tempo atual.

A troca de informações entre blocos pode ser facilmente implementada utilizando-se a simulação orientada por eventos. Por exemplo, um bloco pode agendar um evento para outro bloco contendo as informações a serem trocadas. Assim, quando o *kernel* interpretar este evento, ele poderá chamar o bloco de destino e entregar as informações que lhe pertencem.

Neste caso o *kernel* funciona também como um intermediador para a troca de mensagens entre blocos.

Segundo Shanmugan [1], para a simulação de redes de comunicação e sistemas lógicos, a técnica *event-driven* é mais eficiente do ponto de vista computacional do que a técnica *data-driven*, devido a sua natureza assíncrona. Entretanto, para a simulação de operações de processamento de sinais síncronos, no nível de forma de onda, a técnica *time-driven* é mais rápida.

4.2.3.3 - Simulação Orientada por Dados (*Data-Driven Simulation*)

Nesta técnica de simulação, um bloco só é chamado se todos os dados necessários na sua entrada estiverem disponíveis. Desta forma, se em cada bloco do modelo de simulação for conhecida de antemão a disponibilidade de tais dados, é possível a construção de uma seqüência de chamada de blocos antes do inicio da simulação. Caso contrário, a disponibilidade de tais dados deve ser verificada durante a simulação e os blocos deverão ser chamados dinamicamente.

4.2.4 - Simulação Interativa

Independente da técnica de simulação utilizada, alguns pacotes oferecem a possibilidade de simulação interativa, onde o usuário pode executar passo a passo a simulação e verificar a evolução dos resultados, ou ainda parar a simulação quando certas condições forem encontradas (*check pointing*), trocar o valor dos parâmetros e finalizar a simulação. A simulação interativa fornece recursos para depurar um modelo de simulação, o que possibilita a melhor compreensão da dinâmica do sistema que está sendo simulado.

4.2.5 - Simulação Distribuída

Alguns pacotes de simulação oferecem a capacidade de particionar um modelo de simulação complexo a fim de permitir a execução em múltiplos processadores. Enquanto este particionamento é feito manualmente, a execução da simulação em múltiplos processadores e a comunicação entre estes processadores é feita automaticamente.

4.2.6 - Biblioteca de Modelos

Os modelos de simulação de sistemas de comunicação são construídos a partir da conexão de blocos, geralmente disponíveis em bibliotecas de modelos que são fornecidas

junto com as ferramentas de simulação. A utilização dessas bibliotecas permite a fácil substituição de blocos durante a fase de elaboração de modelos de simulação, bem como a possibilidade de desenvolvimento e incorporação de novos modelos às bibliotecas dessas ferramentas.

4.3 - Simulação de Redes ATM

Como já abordamos anteriormente, a tecnologia ATM ainda encontra-se em constante evolução, e portanto definir quais aspectos devem ser abrangidos por um simulador é uma tarefa bastante complexa.

As características gerais das ferramentas de simulação de sistemas de comunicação apresentadas anteriormente são aplicáveis às ferramentas de simulação de redes ATM, que podem ainda ser caracterizadas segundo o nível em que a simulação ATM é realizada: simulação no nível de células ou simulação no nível de taxa de células. A seguir apresentaremos em maiores detalhes estes dois tipos de simulação ATM.

4.3.1 - Simulação no Nível de Células (*Cell Level Simulation*)

A grande maioria dos pacotes de simulação de redes ATM atuais realizam a simulação no nível de células (Tabela 4.1), o que em geral permite acompanhar o envio de cada célula através da rede, verificando o número de células e o tempo de permanência destas células em cada *buffer* dos equipamentos da rede.

Entretanto, este tipo de simulação exige um grande esforço computacional, algumas vezes até impraticável, tendo em vista o grande número de tarefas que precisam ser executadas pelo *kernel* a fim de simular a transmissão de uma infinidade de células através da rede.

4.3.2 - Simulação no Nível de Taxa de Células (*Cell Rate Simulation*)

A simulação de algumas características das redes ATM, como por exemplo a probabilidade de perda de células, implica na transmissão de um número muito elevado de células através da rede. Nesta situação é desejável o uso de técnicas que acelerem a simulação, como por exemplo a simulação no nível de taxa de células ATM [4]. Este tipo de simulação não se preocupa com o transito isolado das células ATM, mas sim com o "surto de células" que é lançado à rede toda vez que um pacote das camadas superiores é transmitido.

Tal surto é definido como uma taxa de células que permanece constante durante um certo período de tempo.

A simulação no nível de taxa de células utiliza em geral a técnica *event-driven*. Neste tipo de simulação um evento está relacionado com a troca de uma taxa fixa de células para outra, diferindo dos eventos utilizados na simulação no nível de células, onde cada evento está relacionado com o processamento de uma única célula.

4.3.3 - Comparação entre Algumas Ferramentas de Simulação de Redes ATM

A Tabela 4.1 faz uma comparação entre alguns simuladores de redes ATM[1], a partir das características gerais das ferramentas de simulação de sistemas de comunicação e do nível de simulação ATM. Foram investigadas as seguintes ferramentas: NIST ATM *Network Simulator* [5][6], ATM-TN [7][8], *Thunder and Lightning Network Protocol Simulator* [9], CLASS – *ConnectionLess* ATM *Services Simulator* [10][11], OPNET MODELER™ [12] e COMNET III™ [13].

[1] As ferramentas de simulação ATM apresentadas na Tabela 4.1 foram descobertas a partir de procura na *Internet*.

		Características					
Nome	**Desenvolvedor**	**Técnica de Simulação**	**Simulação Interativa**	**Simulação Distribuída**	**Biblioteca de Modelos**	**Interface Gráfica**	**Nível de Simulação ATM**
NIST ATM *Network Simulator*	*National Institute of Standards and Technology* (E.U.A.)	*Event-driven*	Sim	Não	Não	Sim	Célula
ATM-TN	Projeto TeleSim – Universidades de *Calgary* (Canada), *Saskatchewan* (Canada) e *Waikate* (Nova Zelândia)	*Event-driven*	-	Sim	Sim	Sim	Célula
Thunder and Lightning ATM *Network Protocol Simulator*	Departamento de Engenharia Elétrica e Computação da Universidade da Califórnia em *Santa Barbara* (E.U.A.)	*Mixed mode (Event-driven* e *Time-driven)*	Não	Sim	Não	Sim	Célula
CLASS	Grupo de Redes de Comunicação do Departamento de Eletrônica do Instituto Politécnico de *Torino* (Itália)	*Time-driven*	Não	Não	Não	Não	Célula
OPNET MODELER™	MIL 3, *Inc. Washington* DC (E.U.A.)	*Event-driven*	Sim	Não[2]	Sim	Sim	Célula ou Taxa de Células
COMNET III™	CACI *Company California* (E.U.A.)	*Event-driven*	Sim	Não	Sim	Sim	Célula

Tabela 4.1 – Características de algumas ferramentas de simulação de redes ATM.

A partir da Tabela 4.1 podemos observar que a maioria das ferramentas investigadas utiliza a técnica de simulação orientada por eventos (*event-driven*). A exceção é o simulador *Thunder and Lightning*, que utiliza uma técnica mista, onde a transmissão de pacotes através da rede ATM é feita utilizando-se a técnica *time-driven*, enquanto a geração de tráfego em cada fonte da rede é baseada em eventos. Observa-se ainda que a maioria das ferramentas realiza a simulação ATM no nível de células. A exceção é a ferramenta OPNET *Modeler*™,

[2] Recurso atualmente em desenvolvimento.

que segundo os seus desenvolvedores, é flexível o suficiente para implementar a simulação no nível de taxa de células.

4.4 - Referências Bibliográficas

[1] San K. Shanmugan, "*Simulation and Implementation Tools for Signal Processing and Communication Systems*", IEEE *Communications Magazine*, Julho 1994.

[2] Jackson Klein, "SimNT - Uma Ferramenta para a Simulação de Sistemas de Comunicação", Tese de Mestrado, Faculdade de Engenharia Elétrica, UNICAMP, Orientador: Prof. Dr. Leonardo S. Mendes, Agosto 1995.

[3] M.R.N. Ribeiro, H. Waldman, J. Klein, L.S. Mendes, "*Error Rate Patterns for Modeling of Optically Amplified Transmission Systems*", IEEE *Journal of Selected Areas in Communications*, vol. 15, no. 4, pp. 707-716, *May* 1997.

[4] J.M. Pitts, J.A. Schormans, "*Introduction to* ATM *Design and Performance*", *John Wiley & Sons*, 1996.

[5] http://isdn.ncsl.nist.gov/

[6] N. Golmie, A. Koenig, D. Su, "*The* NIST ATM *Network Simulator*", *Operation and Programming*, *Version* 1.0, NIST *Internal Report* 5703, 1995.

[7] http://www.wnet.ca/telesim/

[8] P. Gburzynski, T. Ono-Tesfaye, S. Ramaswamy, "*Modeling* ATM *Networks in a Parallel Simulation Environment*: *A Case Study*", *Proceedings of the* 1995 *Summer Computer Simulation Conference* (SCSC'95), *Ottawa, Ontario*, pp. 847-851, *July* 1995.

[9] Michael D. Santos, P. M. Melliar-Smith, L. E. Moser, "*A Protocol Simulator for the Thunder and Lightning* ATM *Network*", *Conference etaCom'96, Portland*, 1996.

[10] http://hp0tlc.polito.it/

[11] M. A. Marsan, R. L. Cigno, M. Mufano, A. Tonietti, "*Simulation of* ATM *Computer Networks with* CLASS", *Proceedings of the* 7^{th} *International Conference on Computer Performance Evaluation*, pp. 406-424, *Viena, May* 1994.

[12] http://www.mil3.com

[13] http://www.caciasl.com

Capítulo 5

Desenvolvimento do **SimATM**

Neste capítulo apresentaremos o desenvolvimento da estrutura do **SimATM**. Faremos uma breve apresentação das ferramentas de simulação que serviram de ponto de partida para o desenvolvimento do simulador. Apresentaremos também as principais decisões de projeto tomadas durante o desenvolvimento do **SimATM**.

5.1 - Ponto de Partida

O objetivo final deste trabalho é obter uma ferramenta para simulação de redes ATM com os mesmos recursos do ambiente de simulação de sistemas de comunicação **SimNT**. Estes recursos incluem uma estrutura orientada à objetos, interface gráfica amigável, biblioteca de modelos expansível e definição hierárquica de blocos (subsistemas). A fim de melhor poder avaliar a possibilidade de incorporação dos recursos do **SimNT** no **SimATM**, realizamos um estudo deste ambiente, e procuramos obter maiores informações a respeito de ferramentas de simulação de redes ATM. Como resultado, encontramos via *Internet*, um simulador que possui alguns dos recursos de simulação ATM que incorporamos ao **SimATM**. Trata-se do simulador de redes ATM desenvolvido pelo NIST - *National Institute of Standards and Technology*, dos E.U.A. A seguir faremos uma breve descrição do **SimNT** e do simulador desenvolvido pelo NIST.

5.1.1 - **SimNT**

O **SimNT** [1][2] é um aplicativo para desenvolvimento e análise de dispositivos e sistemas de comunicação em geral, desenvolvido no Departamento de Comunicações da Faculdade de Engenharia Elétrica e Computação da UNICAMP. A implementação do **SimNT** surgiu da necessidade de se ter à disposição um aplicativo que permitisse a simulação de sistemas de comunicação (em particular sistemas de comunicação óptica) e fornecesse recursos para o desenvolvimento de novos modelos de dispositivos.

5.1.1.1 - Principais Características

As características gerais das ferramentas de simulação de sistemas de comunicação, apresentadas no capítulo anterior, se aplicam também ao **SimNT**. Dentro deste contexto, as principais características da versão atual do programa são:

- Técnica de Simulação – O **SimNT** utiliza uma técnica de simulação que funciona de forma similar à técnica DDF (*Dynamic Data Flow*), utilizada no ambiente *Ptolemy* [3].
- Biblioteca de Modelos Expansível – O **SimNT** possui uma biblioteca de modelos expansível, o que possibilita o desenvolvimento de novos modelos de dispositivos. Estes modelos são implementados como bibliotecas de ligação dinâmica (DLLs – *Dinamic Linkage Libraries*) do *Windows*™.
- Interface Gráfica – A interface gráfica do **SimNT** foi projetada e desenvolvida utilizando os recursos da API (*Applications Program Interface*) de 32 *bits* do *Win32*. Os recursos de simulação disponíveis através da interface gráfica permitem que o usuário desenhe a topologia do sistema, adicionando os diversos blocos, que representam os modelos. Além disso, é possível alterar os parâmetros locais e globais do sistema e visualizar os resultados numericamente e através de gráficos.
- Desenvolvido em C++ – O **SimNT** foi desenvolvido na linguagem de programação C++ [4][5] e, portanto, usufrui das vantagens da programação orientada a objetos. A utilização dos recursos de simulação do programa dispensa o conhecimento de C++, porém, os desenvolvedores de novos modelos devem conhecer esta linguagem, a fim de manipular adequadamente as funções de entrada e saída de dados.
- Utilização de Subsistemas – O **SimNT** suporta uma definição hierárquica de blocos, possibilitando a criação de subsistemas. Vários blocos podem ser agrupados em um subsistema, originando um novo modelo, sem a necessidade de programação.

5.1.1.2 - Estrutura do SimNT

A estrutura do **SimNT** pode ser dividida em três partes distintas. A primeira é composta pelos modelos de dispositivos, que são os blocos responsáveis pela simulação do comportamento de cada dispositivo. A segunda parte é o *kernel* do simulador, responsável

pelo gerenciamento, ordenação e execução do processo de simulação. A terceira parte é a interface gráfica, que permite a interação amigável entre o usuário e o simulador.

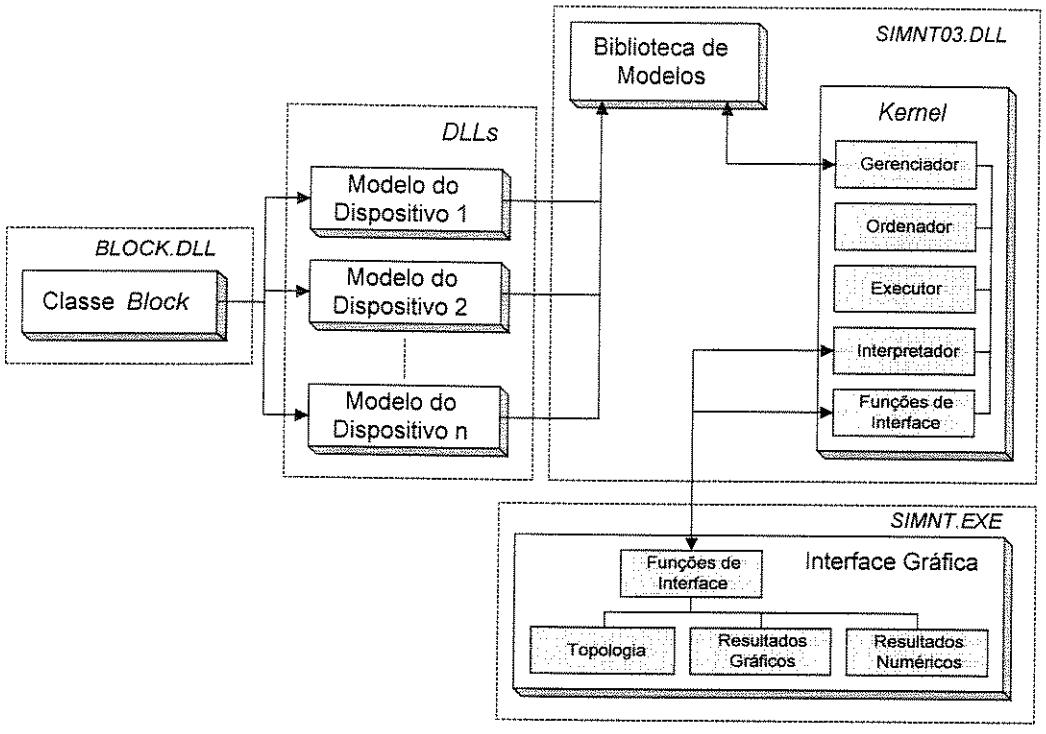


Figura 5.1 – Estrutura geral do SimNT

A Figura 5.1 mostra um diagrama da estrutura do **SimNT**. Os modelos de dispositivos são bibliotecas de ligação dinâmica (DLLs) do *Windows*™, independentes e desenvolvidas em C++. Estes modelos são construídos a partir de um objeto [4][5] chamado *Block*. O *kernel* do simulador e a biblioteca de modelos formam uma DLL que é carregada pela interface gráfica. A biblioteca de modelos atua apenas como uma ponte entre os modelos e o *kernel*, carregando e descarregando os modelos necessários para uma determinada simulação. O *kernel* possui um interpretador de comandos e funções de interface, que possibilitam a manipulação dos sistemas de comunicação a partir da interface gráfica. O *kernel* possui ainda um ordenador de chamadas de blocos, um gerenciador e um executor de chamadas. A interface gráfica contém informações sobre a topologia do sistema a ser simulado, bem como os resultados gráficos e numéricos obtidos em simulações anteriores.

5.1.2 - Simulador de Redes ATM Desenvolvido pelo NIST

O NIST – *National Institute of Standards and Technology*, orgão ligado ao Departamento de Comércio dos E.U.A., desenvolveu um simulador de redes ATM [6] com o objetivo de dotar pesquisadores e projetistas de redes com uma ferramenta de análise,

planejamento e avaliação de redes ATM. O simulador permite ao usuário criar diferentes topologias de rede, estabelecer parâmetros de operação para cada componente, visualizar a atividade da rede durante a simulação e salvar os resultados em arquivo, para posterior análise.

5.1.2.1 - Principais Características

Da mesma forma que o **SimNT**, o simulador de redes ATM desenvolvido pelo NIST também se enquadra como uma ferramenta de simulação de sistemas de comunicação. Como tal, suas principais características são:

- Técnica de Simulação – O simulador do NIST utiliza a técnica de simulação orientada por eventos (*event-driven simulation*), anteriormente descrita no Capítulo 4.
- Interface Gráfica – A interface gráfica do simulador é baseada no ambiente de janelas X *Windows*, que funciona sobre a plataforma UNIX™.
- Desenvolvido em C – O simulador foi desenvolvido em linguagem de programação C [4][5].
- Nível de Simulação ATM – O simulador executa a simulação ATM no nível de células.

A versão atual do simulador não possui biblioteca de modelos. Entretanto, é possível a criação de novos componentes a partir da "recompilação" e "lincagem" do simulador.

5.1.2.2 - Estrutura do Simulador

A estrutura do simulador é voltada para a técnica de simulação orientada por eventos e pode ser dividida em quatro partes distintas: componentes, fila de eventos, gerenciador de eventos e interface gráfica. A Figura 5.2 mostra um diagrama da estrutura do simulador de redes ATM desenvolvido pelo NIST. Os componentes constituem os blocos básicos do simulador. São eles: chaveador (*switch*), enlace físico (*physical link*), equipamento terminal faixa larga (*broadband terminal equipment*) e aplicativo ATM (ATM *application*). Cada componente consiste de uma rotina de ação e de uma estrutura de dados[7][8], ambas implementadas em linguagem C. Esta estrutura de dados é composta por uma lista (lista duplamente encadeada) de componentes vizinhos e por uma fila (lista encadeada) que contém os parâmetros do componente. A fila de eventos é uma estrutura de dados (lista encadeada)

que mantém os eventos ordenados em função do tempo de chegada. O gerenciador de eventos é responsável pelo controle desta fila de eventos, pelo gerenciamento do tempo de simulação e pela execução de eventos. O simulador possui ainda uma interface gráfica que permite visualizar a topologia da rede a ser simulada, conectar seus componentes, trocar parâmetros, estabelecer conexões ATM, controlar a simulação, salvar resultados e manipular arquivos de configuração de rede. A interface gráfica do simulador pode ser dividida nos módulos: topologia de rede, controle de simulação, manipulação de parâmetros, apresentação de mensagens, gerenciamento de gráficos e relógio de simulação.

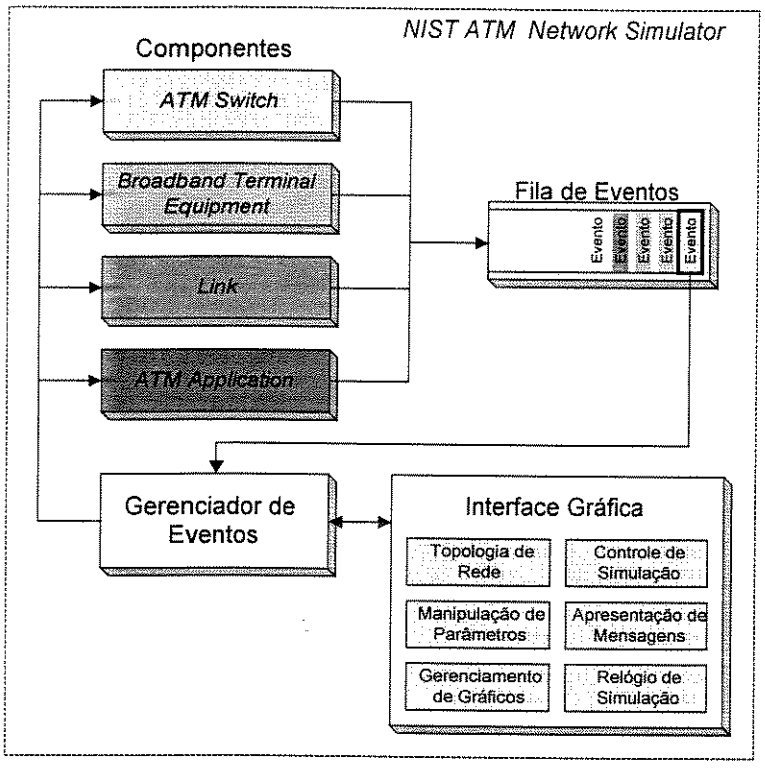


Figura 5.2 – Estrutura do simulador de redes ATM desenvolvido pelo NIST

Toda a troca de informações e envio de células entre componentes é feita através de eventos, que são armazenados na fila de eventos, como mostra a Figura 5.2. O gerenciador de eventos retira o primeiro evento da fila, interpreta-o, e executa a rotina de ação do componente ao qual este evento se destina.

O funcionamento do simulador basicamente não difere daquele apresentado anteriormente na descrição da técnica *event-driven*. Entretanto, no simulador do NIST os eventos que são agendados para um mesmo instante de tempo são executados aleatoriamente, ou seja, nenhuma ordem particular de execução é garantida.

5.2 - Estrutura Proposta para o Simulador de Redes ATM

Nesta seção apresentaremos as principais decisões de projeto tomadas durante o desenvolvimento do simulador de redes ATM. Entre elas estão a técnica de simulação empregada, a linguagem de programação utilizada, o nível da simulação ATM (nível de células ou de taxa de células) e a forma como modelamos a arquitetura das redes ATM.

5.2.1 - Técnica de Simulação

Conforme apresentamos no Capítulo 4, a técnica de simulação baseada em eventos é a técnica mais utilizada no desenvolvimento de ferramentas de simulação de redes ATM. A técnica *event-driven* fornece a eficiência e a flexibilidade necessárias à simulação de redes ATM. Dentro deste contexto, resolvemos desenvolver o **SimATM** utilizando esta técnica.

5.2.2 - Linguagem de Programação

Uma vez que o **SimNT** foi desenvolvido em C++, decidimos por utilizar esta mesma linguagem de programação para o desenvolvimento do **SimATM**.

5.2.3 - Nível da Simulação ATM

Conforme apresentamos no Capítulo 4, a simulação ATM no nível de células permite acompanhar cada célula transmitida através da rede, verificando o seu tempo de permanência nos *buffers* dos equipamentos da rede. A simulação no nível de células ATM apresenta maior fidelidade em relação a simulação no nível de taxa de células ATM. Dentro deste contexto, decidimos implementar a simulação de redes ATM do simulador no nível de células, uma vez que um dos objetivos do simulador é possibilitar o aprendizado da tecnologia ATM.

5.2.4 - Estrutura Baseada na Arquitetura Funcional Geral de um Elemento de Rede ATM

A arquitetura das redes ATM foi modelada a partir do Modelo de Referência de Protocolos da B-ISDN (B-ISDN PRM). Entretanto, o **SimATM** deveria ser uma ferramenta expansível, que possibilitasse a incorporação de novos recursos de simulação ATM de forma modular. Para tanto, a estrutura de simulação ATM do simulador foi baseada em uma série completa de Recomendações sobre a operação funcional de equipamentos ATM e gerenciamento de elementos de redes ATM, definida pelo ITU-T e anteriormente apresentada

no Capítulo 3. Esta estrutura prevê o aperfeiçoamento e a incorporação de novos recursos de simulação ATM baseado na arquitetura funcional de um elemento de rede ATM.

A Figura 5.3 situa os recursos de simulação ATM que são incorporados ao simulador nesta versão, tomando como base a arquitetura funcional geral de um elemento de rede ATM [9].

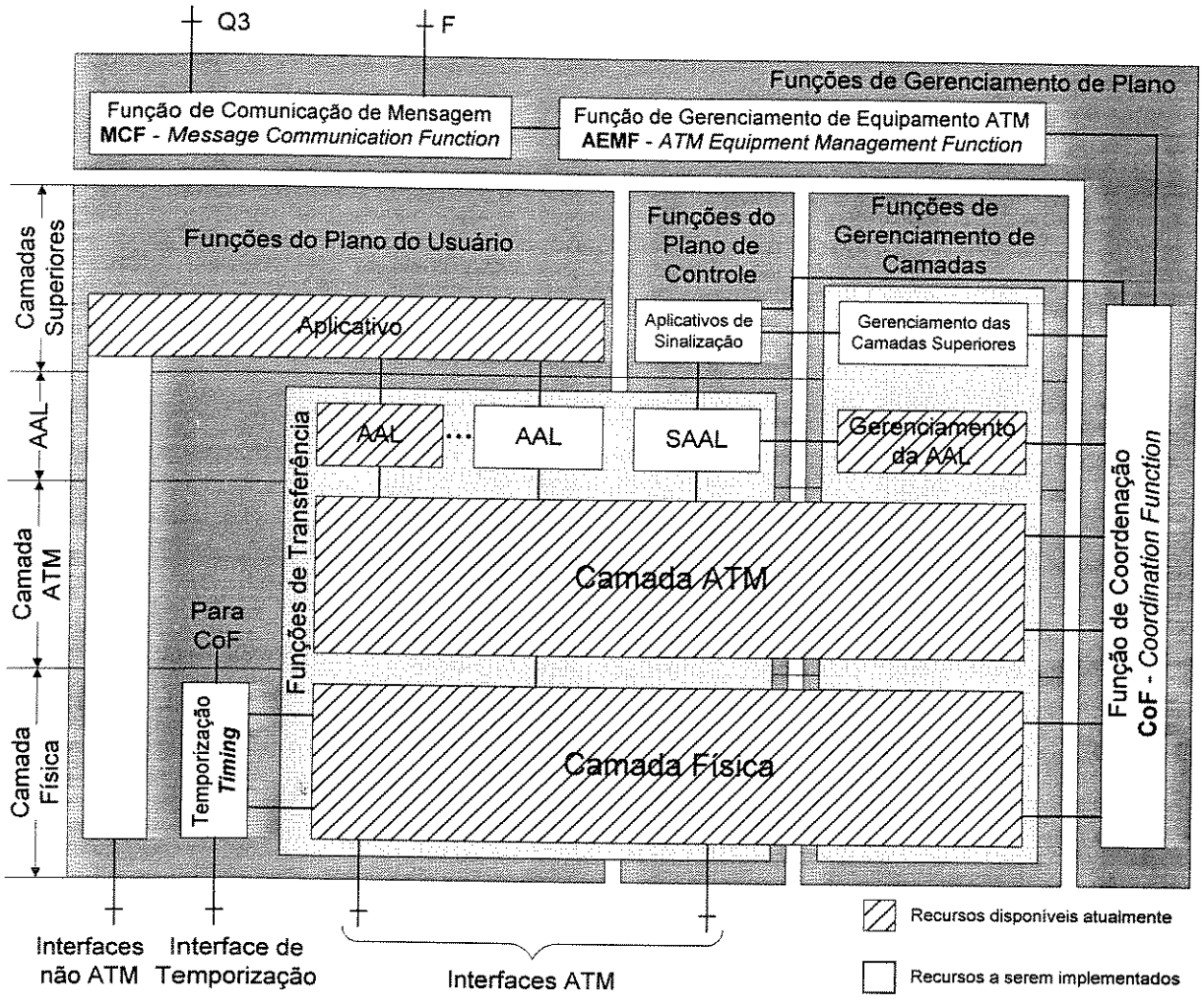


Figura 5.3 – Recursos de simulação ATM disponíveis na versão atual do simulador

Os recursos de simulação ATM do simulador estão limitados às funções de transferência, funções de gerenciamento de camadas e aos aplicativos das camadas superiores da arquitetura funcional geral de um elemento de rede ATM. A implementação de tais recursos é realizada através de modelos de camadas, conforme veremos no próximo capítulo. Os modelos das camadas física, ATM e de adaptação ATM, compreendem as funções de transferência e de gerenciamento de camadas das camadas correspondentes. Nestes modelos de camadas, a implementação das funções de gerenciamento de camadas é parcial, uma vez

que nesta versão não é nosso objetivo a implementação da função de coordenação. O modelo do aplicativo ATM agrupa os modelos de camadas para geração e recepção de tráfego, que compreendem as funções de interconexão e de suporte para outros aplicativos. As funções restantes serão incorporadas de forma modular ao simulador em futuras versões.

5.3 - Referências Bibliográficas

[1] J. Klein, L.S. Mendes, *"SimNT - Manual do Usuário - Versão 0.3"*, Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, Março 1997 (http://www.mc21.fee.unicamp.br/simnt/).

[2] M.R.N. Ribeiro, H. Waldman, J. Klein, L.S. Mendes, "*Error Rate Patterns for Modeling of Optically Amplified Transmission Systems*", IEEE *Journal of Selected Areas in Communications*, vol. 15, no. 4, pp. 707-716, *May* 1997.

[3] J. Bucketal., "*Ptolemy: A Plataform for Heterogeneus Simulation and Prototyping*", *Proc. of the 1991 European Simulation Conference, Copenhagen*, Dinamarca, Junho de 1991 (http://ptolemy.eecs.berkeley.edu/).

[4] B. Stroustrup, "*The C++ Programmig Language*", *Addison-Wesley*, 1995.

[5] Tom Swan, "*Mastering Borland* C++", *Sams Publishing*, 1992.

[6] N. Golmie, A. Koenig, D. Su, "*The* NIST ATM *Network Simulator*", *Operation and Programming*, *Version* 1.0, NIST *Internal Report* 5703, *August* 1995.

[7] W. Ford, W. Topp, "*Data Structures with* C++", *Prentice-Hall, Inc.*, 1996.

[8] Timothy A. Budd, "*Classic Data Structures in* C++", *Addison-Wesley*, 1994.

[9] ITU-T *Recommendation* I.731, "*Types and general characteristics of* ATM *equipment*", *Geneva, March* 1996.

Capítulo 6

Estrutura do **SimATM**

Neste capítulo apresentaremos a estrutura atual do **SimATM**. Começaremos com uma descrição da estrutura do *kernel* e das redes ATM do simulador. Em seguida descreveremos os demais elementos do simulador, abrangendo os modelos de equipamentos, aplicativos, camadas e de sistemas de fila desenvolvidos para o **SimATM**.

6.1 - Estrutura do *Kernel*

A estrutura atual do **SimATM** é ilustrada na Figura 6.1. O *kernel* do simulador possui um módulo interpretador de comandos, uma fila de eventos, um módulo gerenciador de eventos e várias instâncias de redes ATM.

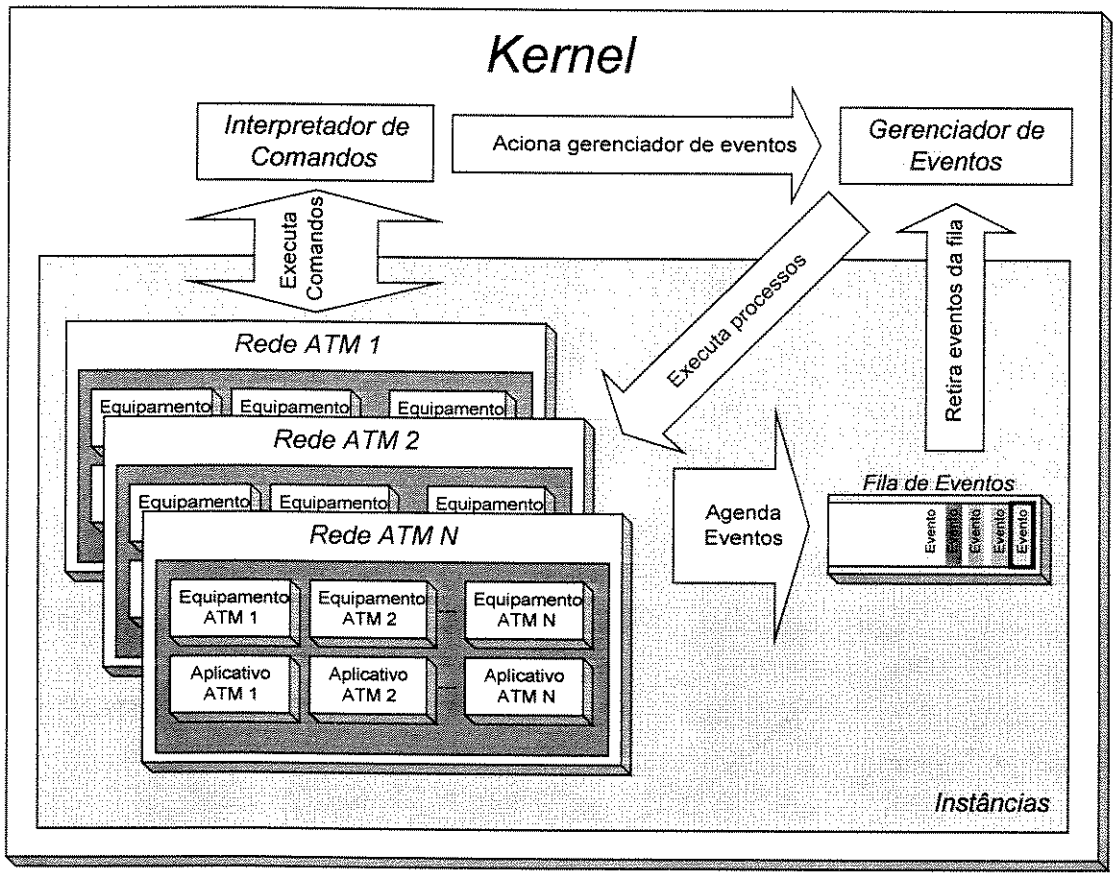


Figura 6.1 – Estrutura atual do SimATM

O módulo interpretador de comandos interpreta e executa comandos junto às várias redes ATM, e ainda aciona o módulo gerenciador de eventos para simular uma das redes ATM. Os componentes de uma rede ATM real são modelados como equipamentos ou aplicativos ATM, compostos por modelos de camadas e de sistemas de fila [4] (*Queueing Systems*) capazes de agendar eventos na fila de eventos. Os equipamentos e aplicativos ATM são construídos a partir de uma estrutura básica comum chamada **bloco**. Chamaremos de blocos do simulador todos os elementos que herdam esta estrutura.

Quando o comando *run* é executado pelo interpretador de comandos a simulação é iniciada e o controle de simulação é passado para o módulo gerenciador de eventos. A fila de eventos armazena os eventos que aguardam por execução. O módulo gerenciador de eventos retira esses eventos da fila e retorna-os para os equipamentos e aplicativos de rede ATM que está sendo simulada. Os equipamentos e aplicativos ATM são capazes de encaminhar os eventos recebidos do gerenciador de eventos para as camadas ou sistemas de fila de destino, fechando assim o ciclo de eventos. A simulação prossegue até que o tempo total de simulação seja alcançado, ou até que não hajam mais eventos para serem executados. Ao final da simulação, o controle é repassado para o interpretador de comandos.

A funcionalidade dos modelos de camadas e sistemas de fila é descrita através de processos, que são um conjunto de tarefas predefinidas. Quando um evento chega a uma determinada camada ou sistema de fila, ele dispara um desses processos, realizando assim um conjunto de tarefas agendadas para serem executadas neste instante de tempo.

6.2 - Estrutura das Redes ATM

A estrutura das redes ATM do simulador é ilustrada na Figura 6.2. A rede ATM possui: um ou mais parâmetros, uma tabela de dados, um módulo gerenciador de parâmetros, um módulo gerenciador de tabelas de dados, um módulo gerenciador de blocos, um módulo gerenciador de conexões ATM, um módulo gerenciador de conexões de dados, um módulo gerenciador de vizinhos, um módulo de inicialização de simulação, um módulo de terminação de simulação e várias instâncias de equipamentos e aplicativos ATM.

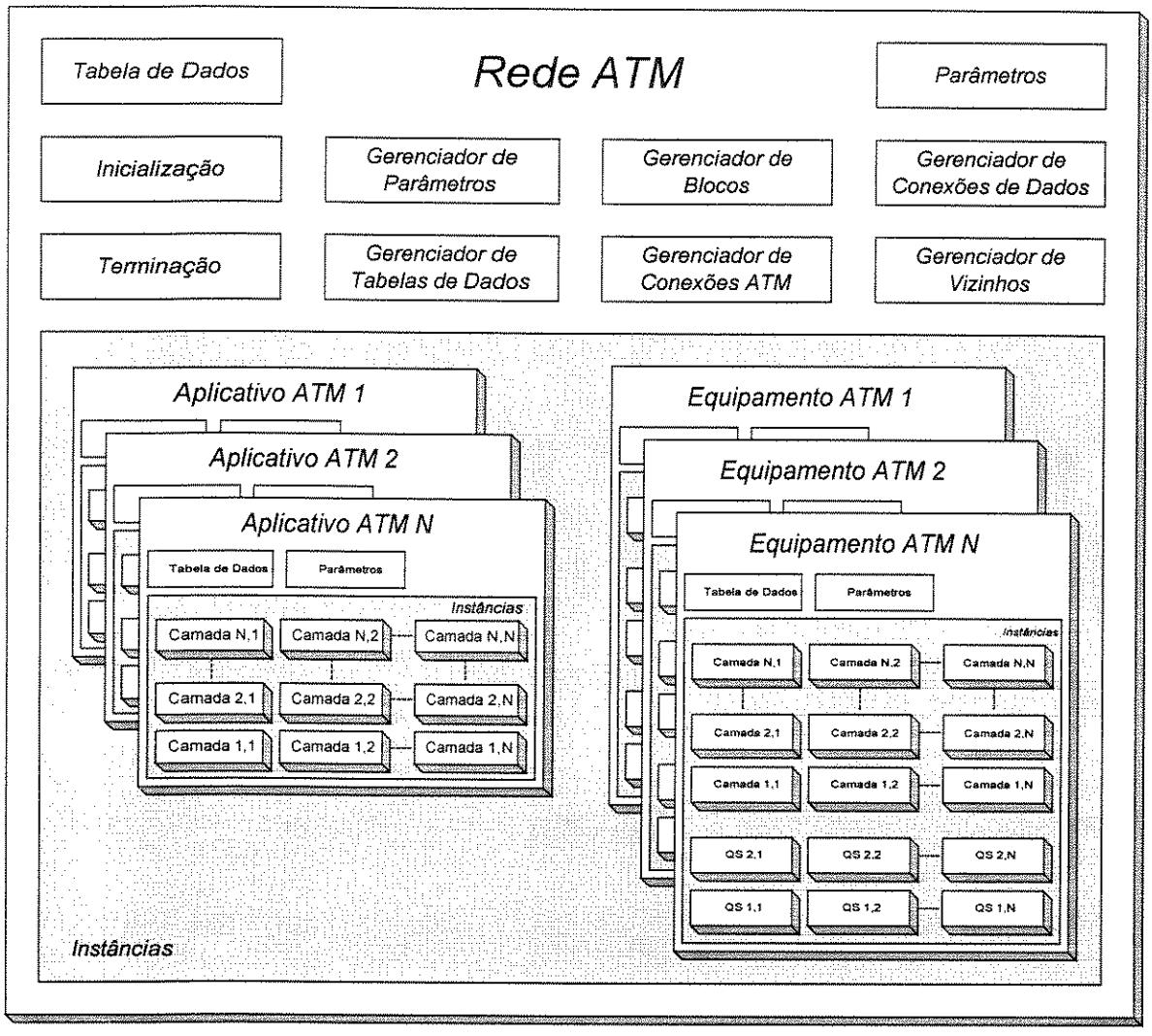


Figura 6.2 – Estrutura de uma rede ATM no SimATM

Os equipamentos ATM possuem camadas e sistemas de fila, enquanto os aplicativos possuem somente camadas, uma vez que o modelo atual dos sistemas de fila é voltado especificamente para o armazenamento e serviço de células ATM, que não trafegam no nível dos aplicativos do **SimATM**. Os equipamentos e aplicativos ATM também possuem um ou mais parâmetros e uma tabela de dados.

O módulo gerenciador de parâmetros permite a manipulação de parâmetros nos blocos da rede ATM, bem como nas suas camadas e sistemas de fila. O módulo gerenciador de tabelas de dados permite a manipulação de tabelas nos blocos da rede ATM, bem como nas suas camadas. Os sistemas de fila não possuem tabelas. O módulo gerenciador de blocos é responsável pela criação, remoção, conexão e desconexão dos blocos da rede ATM. O módulo gerenciador de conexões ATM é responsável pela manipulação das conexões ATM da rede. O módulo gerenciador de conexões de dados é responsável pela manipulação das

conexões de dados da rede ATM. Estas conexões utilizam as conexões ATM preestabelecidas para enviar dados através da rede. O módulo gerenciador de vizinhos é responsável pela manipulação de informações de vizinhança de blocos, uma vez que os blocos da rede ATM somente podem ser conectados entre si mediante a verificação de certas condições. O módulo de inicialização prepara as camadas e sistemas de fila dos blocos da rede para a simulação. O módulo de terminação não é utilizado na versão atual do **SimATM**, por que nenhuma função precisa ser executada nas camadas e sistemas de fila dos blocos após o término da simulação.

A seguir descreveremos em maiores detalhes os demais elementos da estrutura do simulador.

6.3 - Eventos

Eventos são requisições de atendimento, agendadas para um dado instante de tempo, que tem por objetivo executar um processo predefinido. A Figura 6.3 mostra a estrutura dos eventos no **SimATM**.

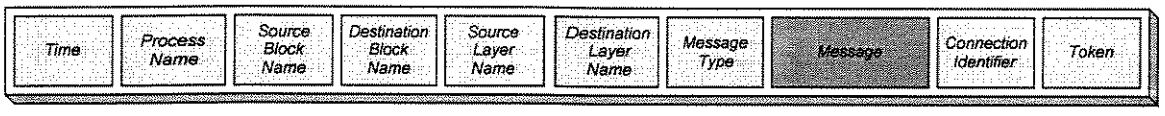


Figura 6.3 – Estrutura dos eventos do SimATM

Os eventos do **SimATM** possuem os seguintes campos de informações:

- *Time* (Tempo) – Tempo programado para a execução (em segundos).
- *Process Name* (Nome do Processo) – Nome do processo a ser executado na camada ou sistema de fila de destino.
- *Source Block Name* (Nome do Bloco Fonte) – Nome do bloco onde o evento foi gerado.
- *Destination Block Name* (Nome do Bloco de Destino) – Nome do bloco a quem o evento se destina. Este campo deve ser sempre preenchido, caso contrário, o evento será ignorado pelo *kernel*.
- *Source Layer Name* (Nome da Camada Fonte) – Nome da camada ou sistema de fila onde o evento foi gerado.
- *Destination Layer Name* (Nome da Camada de Destino) – Nome da camada ou sistema de fila ao qual o evento se destina.

- *Message Type* (Tipo de Mensagem) – Este campo serve para identificar o tipo de mensagem que o evento está carregando. Caso nenhuma mensagem esteja sendo carregada, este campo deve ser configurado com a palavra "NONE_MESSAGE", ou seja, nenhuma mensagem.
- *Message* (Mensagem) – Os eventos do **SimATM** são usados para transportar primitivas de serviço e células ATM de uma camada para outra, nos blocos da rede.
- *Connection Identifier* (Identificador de Conexão) – Em algumas situações é necessário identificar o nome da conexão a que pertence a mensagem transportada. Se isso não for necessário, este campo deve ser configurado com a palavra "UNNECESSARY", ou seja, desnecessário.
- *Token* (Ficha) – Este campo é utilizado para "desempatar" eventos cuja execução foi programada para um mesmo instante de tempo. A utilização deste campo garante que todos os eventos nesta situação serão executados conforme a disciplina FIFO (*First-in-first-out*). O valor da ficha é definido através do uso de um contador, que é incrementado toda vez que um evento é armazenado na fila de eventos.

6.3.1 - Tipos de Eventos

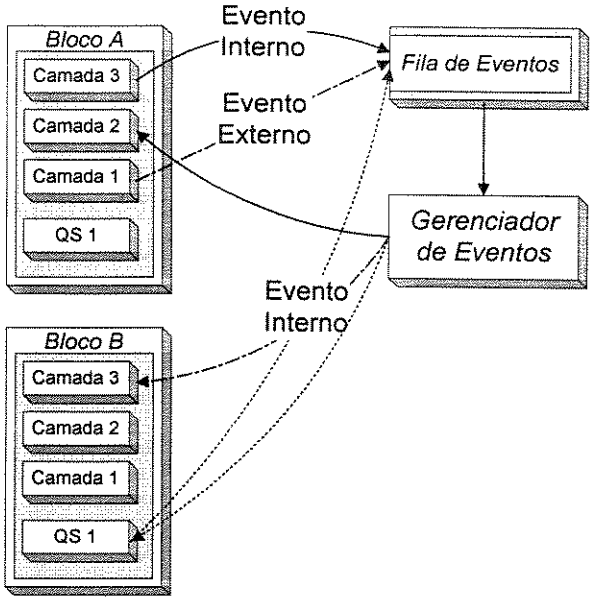


Figura 6.4 – Eventos internos e eventos externos

Os eventos do **SimATM** podem ser classificados segundo a localização da camada ou sistema de fila de destino. Se a camada ou sistema de fila de destino se encontra no mesmo bloco onde o evento foi gerado, ele é dito evento interno. Porém, se a camada ou o sistema de fila de destino se encontra em outro bloco, o evento é dito externo. Esta classificação é importante porque o procedimento de encaminhamento de eventos nos blocos é diferenciado para eventos internos e externos.

6.4 - Fila de Eventos

A fila de eventos armazena os eventos que aguardam por execução no *kernel*. Os eventos são ordenados conforme o tempo agendado para a execução. O evento com menor tempo de execução é o primeiro a ser executado. Se vários eventos estiverem agendados para um mesmo tempo de execução, o primeiro evento que foi agendado na fila será executado.

6.5 - Parâmetros

Parâmetros são variáveis utilizadas para configurar ou modificar o comportamento de modelos de equipamentos, aplicativos, sistemas de fila, redes ATM, etc. O **SimATM** possui dois tipos de parâmetros: parâmetros de simulação e parâmetros de *design*. Parâmetros de simulação são usados para configurar variáveis de estado, como por exemplo o tipo de amostragem de resultados estatísticos a ser utilizado, enquanto parâmetros de *design* são usados para configurar o valor de variáveis específicas, como por exemplo a taxa de transmissão de dados em um enlace. Os parâmetros de *design* permitem a modificação do comportamento lógico/matemático dos blocos.

Os parâmetros do **SimATM** possuem a mesma estrutura dos parâmetros do **SimNT** e, portanto, permitem o armazenamento de valores escalares, vetoriais e matriciais. A Figura 6.5 mostra a estrutura dos parâmetros no **SimATM**.

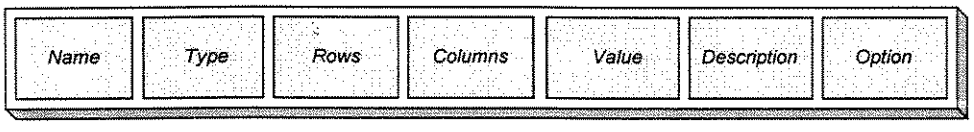


Figura 6.5 – Estrutura geral do parâmetros do SimATM

Os parâmetros do **SimATM** possuem os seguintes campos de informação:

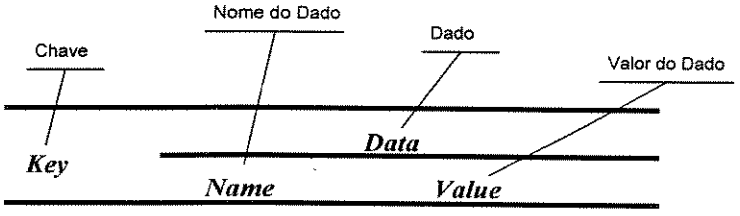
- *Name* (Nome) – Nome do parâmetro.
- *Type* (Tipo) – Este campo serve para identificar o tipo de dado do valor que será armazenado.
- *Rows* (Linhas) – Este campo é utilizado para descrever o número de linhas de um parâmetro. Parâmetros escalares possuem somente uma linha e uma coluna.
- *Columns* (Colunas) – Este campo é utilizado para descrever o número de colunas do parâmetro.

- *Value* (Valor) – Valor do parâmetro.
- *Description* (Descrição) – Descrição do parâmetro.
- *Option* (Opção) – Este campo será utilizado para permitir ou não a modificação de um parâmetro quando o mesmo for amostrado em uma interface gráfica.

6.6 - Tabela de Dados

Os equipamentos reais de uma rede ATM (chaveadores, terminais faixa larga, etc.) utilizam tabelas para armazenar informações relacionadas com as suas conexões, como por exemplo: identificadores de conexão virtual (VPI e VCI), categorias de serviço, larguras de faixa, parâmetros de qualidade de serviço, parâmetros de performance, etc. A tabela de dados é uma estrutura de dados [1][2] que foi desenvolvida com o objetivo de possibilitar o armazenamento, a consulta e a procura eficiente e flexível destas e de outras informações, não só para os equipamentos ATM, mas também para os aplicativos, camadas e para a própria rede ATM.

Para cada dado armazenado na tabela é associado um nome e um valor, que são identificados a partir de uma chave (*key*). Desta forma é possível armazenar vários dados em uma mesma chave, que por exemplo poderia ser o nome de uma conexão ATM ou de uma conexão de dados. A Figura 6.6 mostra o exemplo de uma tabela de dados para o modelo do equipamento terminal faixa larga do **SimATM**.



"DC_1"	"NCI"	"VCC_1"
"VCC_1"	"*Output* VCI"	33
"VCC_1"	"*Output* VPI"	0
"VCC_1"	"*Service categorie* "	"VBR"
"VCC_1"	"*Output port*"	"PHY"
"DLN"	"SW_1"	"PHY"
"DBN"	"PHY"	"SW_1"
"PHY"	"PHY_QS"	"PHY_QS"

Figura 6.6 – Exemplo de tabela de dados

Portanto, tomando como base a Figura 6.6, cada nova linha ou registro na tabela é composta por uma chave, que pode ser nova ou já existente, um nome para o novo dado e o valor desse dado. Se já existir um dado com o mesmo nome na tabela, o valor associado a esse dado será sobreposto.

6.7 - Unidades de Dados

6.7.1 - CPCS-SDU

As AAL-SDUs, conforme a nomenclatura vista no item 2.3.1, são as unidades de dados de serviço de uma camada de adaptação ATM. O modelo atual da camada AAL 5 desenvolvido para o **SimATM**, que será descrito no item 6.13.1.1, assume que a subcamada de convergência de serviços específicos (SSCS – *Service Specific Convergence Sublayer*) da AAL 5 é nula. Desta forma, as AAL-SDUs são mapeadas diretamente para as CPCS-SDUs, que são as SDUs da subcamada de convergência de serviços específicos (CPCS – *Common Part Convergence Sublayer*). A Figura 6.7 mostra a estrutura da CPCS-SDU no **SimATM**.

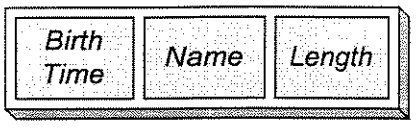


Figura 6.7 – Estrutura da CPCS-SDU no SimATM

Os campos de informações da CPCS-SDU são:

- *Birth Time* (Tempo de Criação) – Este campo contém o instante de tempo em que uma CPCS-SDU é criada. Quando uma CPCS-SDU é entregue ao seu destino na outra extremidade da rede, esse campo é utilizado para calcular o tempo de permanência total dessa CPCS-SDU na rede.
- *Name* (Nome) – Este campo serve para identificar a CPCS-SDU.
- *Length* (Tamanho) – Este campo contém o tamanho da CPCS-SDU em *bytes*.

É importante observar aqui que, na versão atual do **SimATM** qualquer unidade de dados de protocolo procedente das camadas superiores à AAL será modelado de acordo com a estrutura apresentada na Figura 6.7. Portanto, no **SimATM** não é possível inserir o campo de dados (*payload*) de uma CPCS-SDU.

6.7.2 - CPCS-PDU

A CPCS-PDU, conforme a nomenclatura vista no item 2.3.1, é a unidade de dados de protocolo da subcamada de convergência de serviços específicos (CPCS) da AAL. Esta unidade de dados é composta das informações de controle de protocolo dessa subcamada e da CPCS-SDU. A Figura 6.8 mostra a estrutura da CPCS-PDU no **SimATM**.

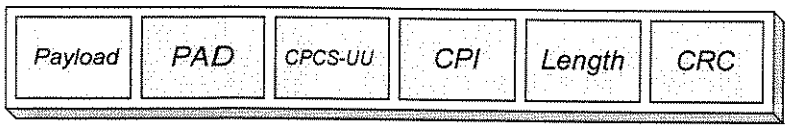


Figura 6.8 – Estrutura do CPCS-PDU no SimATM

Os campos de informações da CPCS-PDU são:

- *Payload* (Carga) – Este campo equivale ao CPCS-PDU *payload* da CPCS-PDU da camada AAL 5.
- PAD (Enchimento) – Este campo equivale ao campo *Padding* da CPCS-PDU da camada AAL 5.
- CPCS-UU (Indicação de Usuário para Usuário CPCS) – Este campo equivale ao campo CPCS *User-to-User Indication* da CPCS-PDU da camada AAL 5.
- CPI (Indicador de Parte Comum) – Este campo equivale ao campo *Common Part Indicator* da CPCS-PDU da camada AAL 5.
- *Length* (Comprimento) – Este campo equivale ao campo *Length* da CPCS-PDU da camada AAL 5.
- CRC (*Cyclic Redundancy Check*) (Checagem de Redundância Cíclica) – Este campo equivale ao campo CRC da CPCS-PDU da camada AAL 5.

Devido a versão atual do **SimATM** implementar somente a camada de adaptação ATM tipo 5, conforme veremos no item 6.13.1.1, o formato da CPCS-PDU do simulador é baseado na estrutura da CPCS-PDU para essa camada.

6.8 - Primitivas

Primitivas são requisições de serviço passadas entre camadas que contém informações de controle de protocolo e dados de usuário. Todas as primitivas apresentadas anteriormente, no Capítulo 2, fazem parte da estrutura do **SimATM**. A troca de primitivas entre as camadas dos blocos da rede é feita através de eventos.

As primitivas implementadas são: CPCS-UNITDATA *Invoke*, CPCS-UNITDATA *Signal*, SAR-UNITDATA *Invoke*, SAR-UNITDATA *Signal*, ATM-DATA *Request*, ATM-DATA *Indication*, PHY-DATA *Request* e PHY-DATA *Indication*.

6.9 - Células ATM

Conforme já apresentamos no Capítulo 2, todas as informações trafegam pela rede ATM no formato de pequenos pacotes de tamanho fixo, chamados de células ATM. A Figura 6.9 mostra a estrutura das células no **SimATM**.

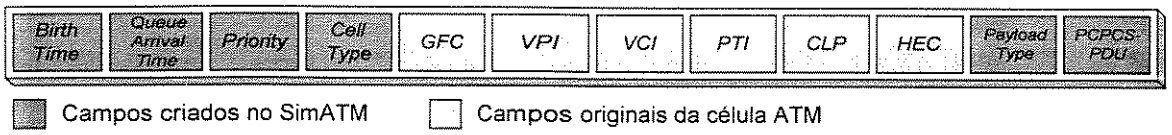


Figura 6.9 – Estrutura das Células no SimATM

As células do **SimATM** possuem os seguintes campos de informações:

- *Birth Time* (Tempo de Criação) – Este campo contém o instante de tempo em que uma célula ATM é criada. Quando uma célula chega ao seu destino, esse campo é utilizado para calcular o tempo de permanência total dessa célula na rede.
- *Queue Arrival Time* (Tempo de Chegada na Fila) – Este campo contém o instante de tempo em que uma célula é armazenada no *buffer* de um sistema de fila. Quando a célula é retirada, esse campo é utilizado para calcular o tempo de permanência da célula nesse *buffer*.
- *Priority* (Prioridade) – Este campo define a prioridade de atendimento da célula. Essa prioridade é calculada nas camadas dos equipamentos ATM, e pode ter significado local.
- *Cell Type* (Tipo) – Este campo define o formato da célula. Os formatos possíveis são: células "UNI" e células "NNI".
- GFC – *Generic Flow Control* (Controle de Fluxo Genérico) – Este campo equivale ao campo GFC das células UNI. Para células NNI este campo é configurado para – 1.
- VPI – *Virtual Path Identifier* (Identificador de Caminho Virtual) – Este campo equivale ao campo VPI do cabeçalho das células ATM.
- VCI – *Virtual Channel Identifier* (Identificador de Canal Virtual) – Este campo equivale ao campo VCI do cabeçalho das células ATM.

- PTI – *Payload Type Identifier* (Identificador de Tipo de Carga) – Este campo equivale ao campo PT do cabeçalho das células ATM.
- CLP – *Cell Loss Priority* (Prioridade de Perda de Célula) – Este campo equivale ao campo CLP do cabeçalho das células ATM.
- HEC – *Header Error Control* (Controle de Erro de Cabeçalho) – Este campo equivale ao campo HEC do cabeçalho das células ATM.
- *Payload Type* (Tipo de Carga) – Este campo é usado para descrever o tipo de de carga que a célula está transportando. Os tipos possíveis são: célula vazia ("EMPTY_CELL") e célula de usuário ("USER_CELL").
- PCPCS-PDU – Este campo serve para identificar a CPCS-PDU da qual a célula ATM foi gerada. No **SimATM**, a CPCS-PDU original é guardada para ser reutilizada ao final do processo de remontagem na AAL de destino.

6.10 - Fila de Células

A fila de células armazena as células que aguardam por atendimento no servidor de um sistema de fila. A célula de maior prioridade tem preferência para ser atendida. Se varias células tiverem a mesma prioridade, elas serão atendidas conforme a sua ordem de chegada, ou seja, segundo a disciplina FIFO.

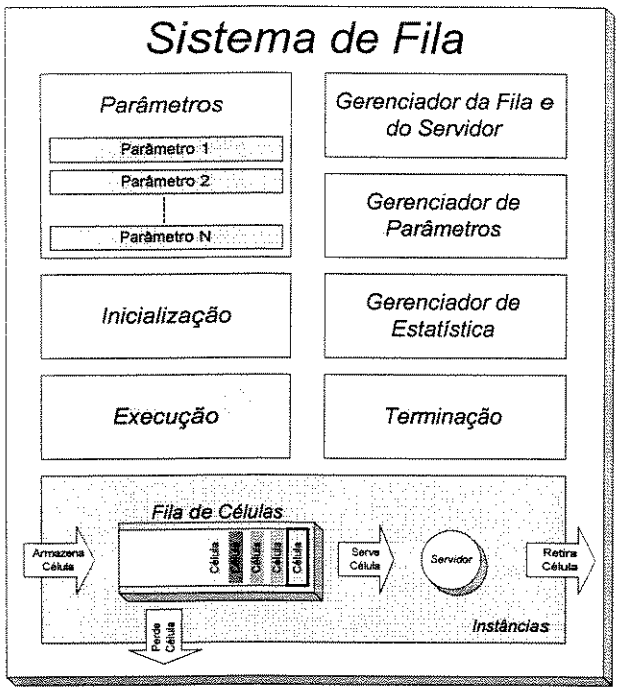
6.11 - Sistemas de Fila

Um sistema de fila [4] é um modelo para um sistema de atendimento de clientes. Os clientes aguardam em filas pelo serviço de um dos servidores do sistema. Um exemplo de um sistema de filas é um banco. Os clientes aguardam pelo atendimento, na maioria das vezes em uma única fila. Quando um caixa está disponível, o cliente de maior prioridade será atendido, e deixará o sistema após um certo período de tempo.

O sistema de fila do **SimATM** (*Queueing System*) possui somente um servidor e uma fila com prioridades. Os clientes do sistema, no caso as células ATM, aguardam pelo serviço na fila (*buffer*) e são atendidas uma de cada vez. O serviço executado varia em função da camada que utiliza o sistema de fila. Na camada física, o servidor executa a transmissão das células através de um enlace físico. Na camada ATM, o servidor executa a retirada das células dos *buffers* de saída. Em ambos os casos o tempo de serviço é constante.

A versão atual do **SimATM** somente utiliza os sistemas de fila para o armazenamento e atendimento de células ATM. Entretanto, em versões futuras pretendemos generalizar esse sistema a fim de armazenar qualquer tipo de mensagem ou pacote.

6.11.1 - Descrição do Modelo



Cada sistema de fila possui um nome, vários parâmetros, vários processos, uma fila de células, um módulo gerenciador de parâmetros, um módulo gerenciador de estatísticas, um módulo gerenciador da fila e do servidor, um módulo de inicialização, um módulo de execução e um módulo de terminação. Os sistemas de fila não possuem tabela de dados. A Figura 6.10 ilustra a estrutura do modelo de sistema de fila do **SimATM**, e a Figura 6.11 ilustra o contexto do modelo de sistema de fila em um equipamento ATM.

Figura 6.10 – Estrutura dos sistemas de filas do SimATM

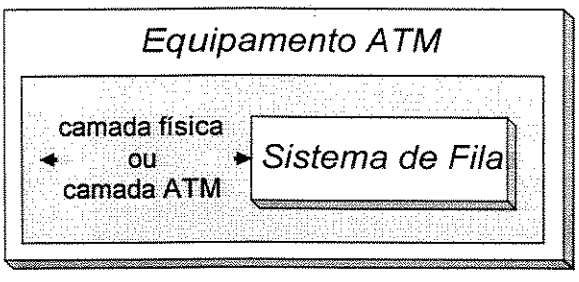


Figura 6.11 – Contexto do sistema de fila em um equipamento ATM

Os processos do sistema de fila são responsáveis pela armazenagem em arquivo das variáveis de estado do sistema de fila, e são disparados através de eventos. O sistema de fila pode agendar seus próprios eventos na fila de eventos junto ao *kernel*. A fila de células armazena as células que aguardam por serviço. O módulo gerenciador de parâmetros permite a manipulação de parâmetros do bloco e da rede a que o sistema de fila pertence. O módulo gerenciador de estatística coordena a amostragem de variáveis para arquivo. O módulo gerenciador da fila e do servidor é responsável pela manipulação de células na fila e no

servidor. O módulo de inicialização é responsável pela atualização dos valores dos parâmetros do sistema de fila, pela criação dos arquivos de amostragem de variáveis de estado e pela geração de amostras iniciais para estes arquivos. O módulo de execução contém os processos do sistema de fila. O módulo de terminação é responsável pelo fechamento dos arquivos de amostragem de variáveis de estado.

6.11.2 - Variáveis de Estado

A fim de caracterizar o comportamento do sistema de fila segundo a teoria de filas [4][5], várias variáveis de estado foram definidas. Estas variáveis se dividem basicamente em três tipos: número de células, tempo de permanência de células e de estado do *buffer* e do servidor. A Figura 6.10 ilustra as variáveis de estado do modelo de sistema de fila do **SimATM**.

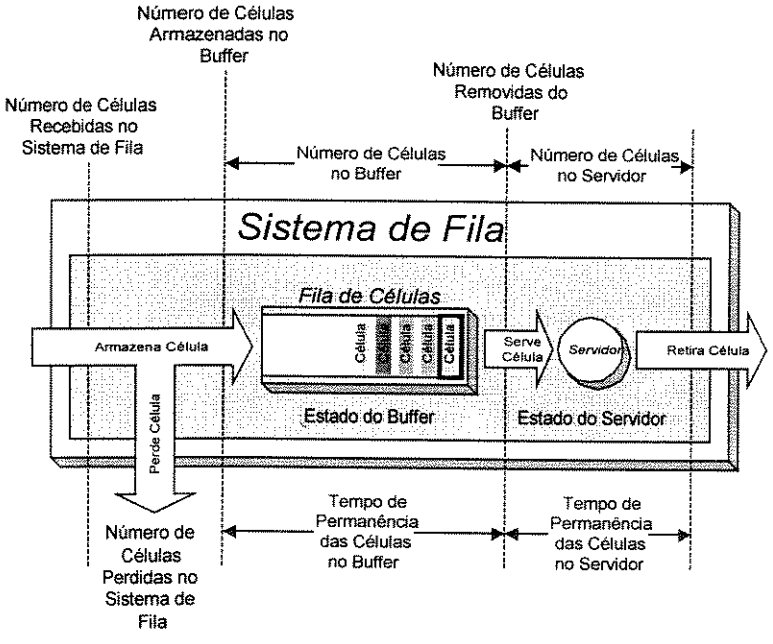


Figura 6.12 – Variáveis de estado do sistema do fila

Toda vez que uma célula é recebida no sistema de fila a variável "número de células recebidas no sistema" é incrementada. Se a célula for armazenada no *buffer* do sistema de fila as variáveis "número de células no *buffer*" e "número de células armazenadas no *buffer*" são incrementadas, caso contrário a variável "número de células perdidas" é incrementada. A variável "estado do *buffer*" é atualizada toda vez que uma célula é armazenada ou retirada do *buffer*. Quando uma célula é retirada do *buffer* a variável "número de células no *buffer*" é decrementada, e a variável "tempo de permanência das células no *buffer*" é atualizada para o tempo de permanência desta célula no *buffer*. A variável "número de células no servidor" é

incrementada toda vez que uma célula é encaminhada para o servidor e decrementada toda vez que uma célula é retirada do serviço. Entretanto, no modelo atual do sistema de fila somente uma célula pode ser servida por vez e, portanto, esta variável se iguala a variável "estado do servidor". A variável "tempo de permanência das células no servidor" é atualizada com o tempo de permanência da última célula retirada de serviço. A variável "estado do *buffer*" pode assumir três valores: "BUFFER_EMPTY", "BUFFER_BUSY" e "BUFFER_FULL". O valor "BUFFER_EMPTY" indica que o *buffer* está vazio. O valor "BUFFER_BUSY" indica que o *buffer* está ocupado. E finalmente, o valor "BUFFER_FULL" indica que o *buffer* está cheio. A variável "estado do servidor" pode assumir dois valores: "SERVER_EMPTY", para servidor vazio, e "SERVER_BUSY", para servidor ocupado.

Tipo	Variável	Tradução	Abreviatura
Variáveis Instantâneas	Número de Células Recebidas no Sistema de Fila	*Number of System Received Cells*	NoSRC
	Número de Células Armazenadas no Buffer	*Number of Buffer Queued Cells*	NoBQC
	Número de Células Perdidas no Sistema de Fila	*Number of Buffer Dropped Cells*	NoBDC
	Estado do Buffer	*Buffer Status*	BS
	Número de Células no Buffer	*Number of Buffer Cells*	NoBC
	Tempo de Permanência das Células no Buffer	*Delay of Buffer Cells*	DoBC
	Estado do Servidor	*Server Status*	SS
	Número de Células no Servidor	*Number of Server Cells*	NoSC
	Tempo de Permanência das Células no Servidor	*Delay of Server Cells*	DoSC
Variáveis Médias	Média da Amostragem do Número de Células no Buffer	*Sample Mean NoBC*	SMNoBC
	Variância da Amostragem do Número de Células no Buffer	*Sample Variance NoBC*	SVNoBC
	Média da Amostragem do Tempo de Permanência das Células no Buffer	*Sample Mean DoBC*	SMDoBC
	Variância da Amostragem do Tempo de Permanência das Células no Buffer	*Sample Variance DoBC*	SVDoBC
	Média da Amostragem do Número de Células no Servidor	*Sample Mean NoSC*	SMNoSC
	Variância da Amostragem do Número de Células no Servidor	*Sample Variance NoSC*	SVNoSC

Tabela 6.1 – Variáveis de estado instantâneas e médias

As variáveis de estado mostradas na Figura 6.12 são variáveis instantâneas. É, portanto, possível acompanhar o regime transitório do sistema de fila em qualquer instante de tempo durante a simulação. Entretanto, o acompanhamento da evolução dos valores médios das variáveis de estado é fundamental. Para isto, foram definidas variáveis de estado médias, que são os valores médios de algumas variáveis de estado chaves (segundo a teoria de filas) do sistema. A Tabela 6.1 sintetiza as variáveis de estado instantâneas e médias de um sistema de fila.

6.11.3 - Amostragem de Variáveis de Estado para Arquivo

Dois tipos de amostragem de variáveis de estado para arquivo são disponíveis no **SimATM**: amostragem por ocorrência e amostragem por tempo. Na amostragem por ocorrência as variáveis de estado instantâneas são gravadas para arquivo toda vez que uma célula é armazenada ou retirada do sistema de fila, o que caracteriza uma ocorrência. Na amostragem por tempo dois arquivos são utilizados: um para as variáveis instantâneas e um para as variáveis médias. A amostragem é feita de tempos em tempos dentro de um intervalo predefinido.

6.11.4 - Cálculo das Variáveis de Estado Médias

O cálculo da média da amostragem de uma variável de estado é feita de duas formas: uma para as variáveis de número de células e outra para as variáveis de tempo de permanência de células. Se a variável de estado for relativa ao número de células, a média é calculada de forma ponderada, ou seja, é levado em conta o tempo que a variável permaneceu em um determinado estado. A fórmula utilizada é a seguinte:

$$\bar{X}_n = \frac{1}{t_n} \sum_{i=1}^{n} X_i . \Delta t_i$$

Onde:

- $\bar{X}_n$ é a média da variável de estado na amostra n.
- t_n é o instante de tempo da amostra n.
- X_i é a amostra de número i.
- Δt_i é o intervalo de tempo entre a amostra i-1 e a amostra i.

Porém, se a variável de estado for relativa ao tempo de permanência de células, a média é calculada de forma simplificada. A fórmula utilizada é a seguinte:

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Onde:

- $\bar{X}_n$ é a média da variável de estado na amostra n.
- X_i é a amostra de número i.

A variância da amostragem é calculada, durante a simulação, utilizando-se o último valor disponível da média. A fórmula é a seguinte:

$$S^2{}_n = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}\,i)^2$$

Onde:

- $S^2{}_n$ é a variância da amostragem da variável de estado na amostra n.
- X_i é a amostra de número i.
- $\bar{X}_i$ é a média da variável de estado na amostra i.

OBS.: É importante observar aqui que a média da variável de estado utilizada no cálculo da variância é a média da amostragem após n amostras e, portanto, a variância calculada é apenas uma estimativa da variância final das amostras, uma vez que a média final ainda não é conhecida. Para a primeira amostra a variância não é calculada.

6.11.5 - Parâmetros

O modelo de sistema de fila possui os seguintes parâmetros: tipo de *buffer* (*BufferType*), tamanho do *buffer* (*BufferSize*), estado da amostragem de variáveis por tempo (*TriggerByTimeStatus*), estado da amostragem de variáveis por ocorrência (*TriggerByCellStatus*), estado do cálculo e amostragem de variáveis médias (*SampleStatisticsStatus*), intervalo de tempo entre amostras por tempo (*LoggingEveryXSecs*), intervalo entre amostras por ocorrência (*LoggingEveryXCells*). Os valores *default* são: *BufferType* = "INFINITE", *BufferSize* = 4294967295, *TriggerByTimeStatus* = "OFF", *TriggerByCellStatus* = "OFF", *SampleStatisticsStatus* = "OFF", *LoggingEveryXSecs* = 1 ms,

LoggingEveryXCells = 1 célula. O valor *default* do parâmetro *BufferSize* é o tamanho máximo que a fila de células pode atingir. O parâmetro *LoggingEveryXCells* determina de quantas em quantas células armazenadas ou retiradas do sistema uma amostra para arquivo deve ser feita.

6.11.6 - Arquivos de Saída

O modelo do sistema de fila possui três arquivos de saída: arquivo de amostragem de variáveis instantâneas por tempo, arquivo de amostragem de variáveis instantâneas por ocorrência e arquivo de amostragem de variáveis médias por tempo. Estes arquivos possuem a extensão *.dat*, e podem ser analisados em ferramentas de visualização científica externas ao **SimATM**, tal como o *Origin*™.

6.11.7 - Funcionamento

As funções desempenhadas pelo modelo de sistema de fila do **SimATM** podem ser divididas em dois procedimentos: *Schedule Cell* (Armazena Célula) e *Remove Cell* (Remove Célula). A seguir detalharemos estes procedimentos.

6.11.7.1 - Procedimento *Schedule Queueing System Cell*

O procedimento *Schedule Queueing System Cell* é responsável pela armazenagem das células no sistema de fila, se possível, e pela amostragem para arquivo das variáveis de estado instantâneas e médias. A Figura 6.13 mostra um fluxograma deste procedimento.

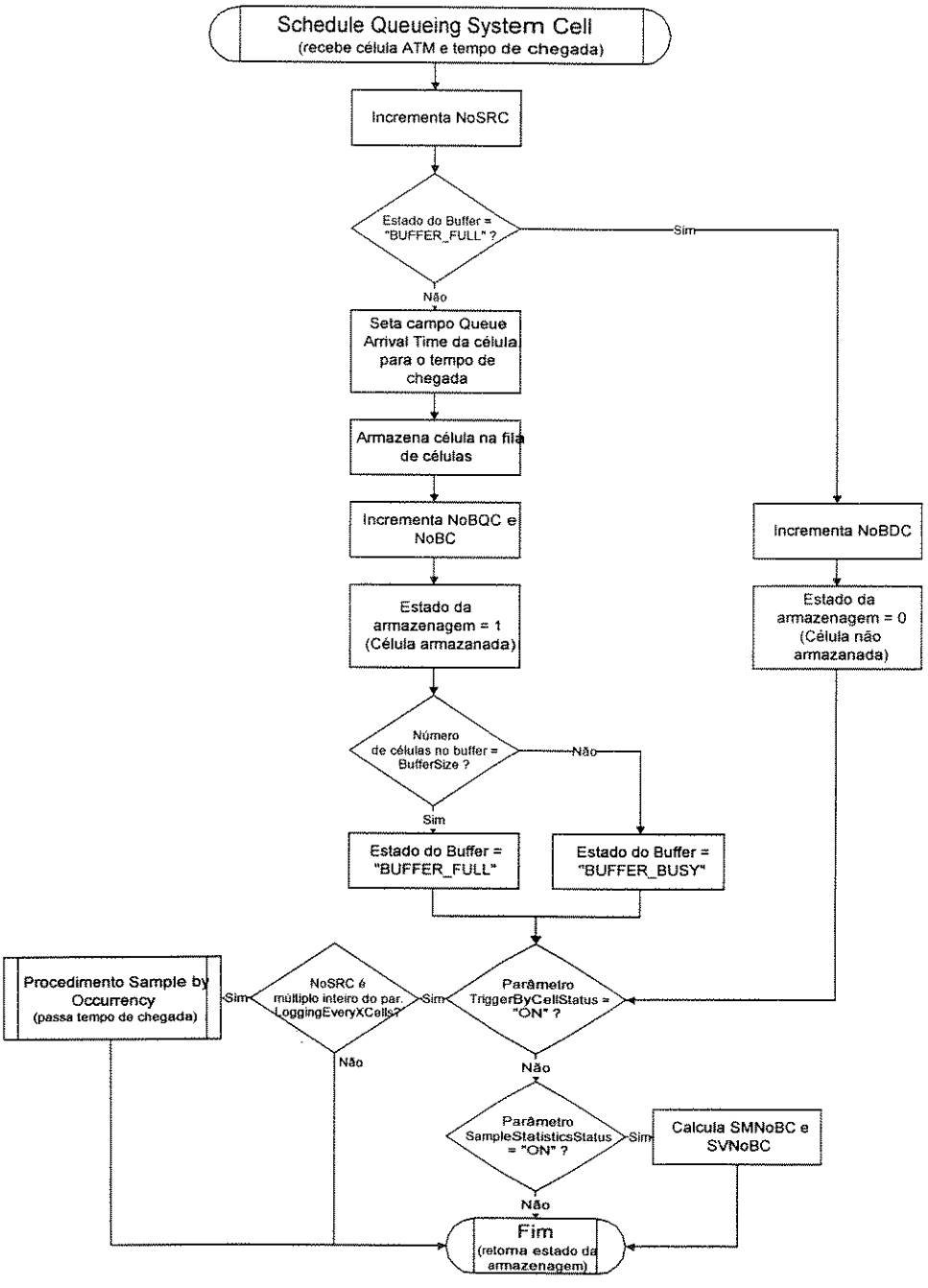


Figura 6.13 – Fluxograma do procedimento *Schedule Queueing System Cell*

Quando uma célula é encaminhada para o sistema de fila a variável NoSRC é incrementada e o estado do *buffer* é verificado. Se o *buffer* estiver cheio, a variável NoBDC é incrementada e o estado da armazenagem é configurado para 0, o que significa que a célula não foi armazenada no *buffer*. Se o *buffer* não estiver cheio, o campo *Queue Arrival Time* da célula é configurado com o tempo de chegada da célula no sistema de fila e a célula é armazenada no *buffer*. As variáveis NoBQC e NoBC são incrementadas e o estado da armazenagem é configurado para 1, o que significa que a célula foi armazenada com sucesso.

É necessário então reconfigurar o estado do *buffer*. Se o número de células no *buffer* for igual ao parâmetro *BufferSize* então o estado do *buffer* é configurado para "BUFFER_FULL", caso contrário o estado do buffer é configurado para "BUFFER_BUSY" (*buffer* ocupado). A seguir é verificado se o parâmetro *TrigerByCellStatus* é igual a "ON", ou seja, está habilitado. Se o parâmetro estiver habilitado é verificado se a variável NoSRC é um múltiplo inteiro do valor do parâmetro *LoggingEveryXCells*. Em caso positivo é feita a amostragem para arquivo das variáveis de estado instantâneas. Finalmente, é verificado o valor do parâmetro *SampleStatisticsStatus*. Se o valor do parâmetro for igual a "ON", é feito o cálculo da média e da variância do número de células no *buffer*, ou seja, das variáveis SMNoBC e SVNoBC.

6.11.7.2 - Procedimento *Remove Queueing System Cell*

O procedimento *Remove Queueing System Cell* é responsável pela remoção das células do sistema de fila, pelo encaminhamento das células armazenadas no *buffer* para o servidor e pela amostragem para arquivo das variáveis de estado instantâneas e médias. A Figura 6.14 mostra um fluxograma deste procedimento.

Quando o procedimento é acionado, inicialmente é verificado se o parâmetro *TriggerByCellStatus* está habilitado. Se o valor do parâmetro for "ON", é verificado se a variável NoSRC é um múltiplo inteiro do parâmetro *LoggingEveryXCells*. Em caso afirmativo, é feita uma amostra das variáveis instantâneas. Em seguida é verificado o estado do servidor. Se o servidor estiver cheio, é decrementada a variável NoSC e verificado o valor do parâmetro *SampleStatisticsStatus*. Se o valor do parâmetro for "ON", é feito o cálculo das variáveis médias: SMNoSC e SVNoSC. Em seguida a célula que está sendo servida é retirada do servidor e o cálculo do tempo de permanência (DoSC) desta célula é feito. O estado do servidor é atualizado para "SERVER_EMPTY" (servidor vazio). Na seqüência é verificado se o *buffer* está vazio, a fim de encaminhar outra célula ao serviço. Se o *buffer* estiver vazio, a variável NoBC é decrementada e a variável NoBRC é incrementada. É ainda verificado o número atual de células no *buffer*. Se não houverem mais células no *buffer*, o seu estado é configurado para "BUFFER_EMPTY", caso contrário o estado do *buffer* é configurado para "BUFFER_BUSY". A célula é então removida do *buffer* e o tempo de permanência desta célula (DoBC) é calculado. O campo *Queue Arrival Time* é configurado para o tempo de partida da célula do *buffer*, a célula ATM é colocada no servidor do sistema e o estado do servidor é configurado para "SERVER_BUSY". São feitas ainda amostras de variáveis

instantâneas para arquivo e cálculos de variáveis médias, a depender dos parâmetros *TriggerByCellStatus* e *SampleStatisticsStatus*, respectivamente.

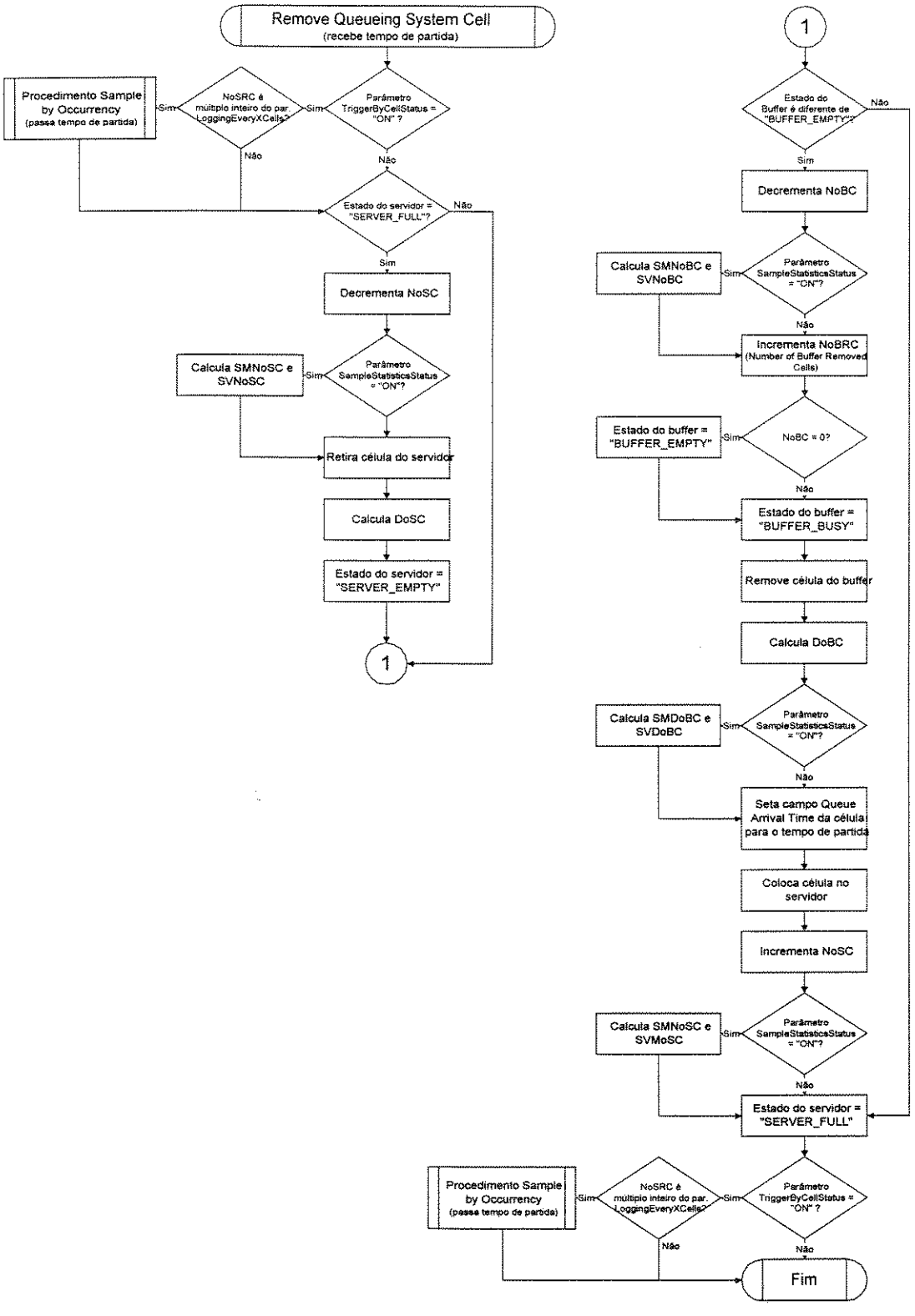


Figura 6.14 – Fluxograma do procedimento *Remove Queueing System Cell*

6.11.8 - Processos

O modelo de sistema de fila possui os processos: LOG_BY_TIME e LOG_SAMPLE_STATISTICS. Estes processos estão relacionados com a amostragem das variáveis de estado para arquivo e são implementados no módulo de execução do modelo.

6.12 - Camadas

Uma camada é um conjunto de protocolos e funções específicas. Várias camadas sobrepostas constituem um modelo de referência. No **SimATM** as camadas estão estruturadas de acordo com o modelo de referência OSI da B-ISDN, apresentado no Capítulo 2.

A camada é um elemento da estrutura do **SimATM** que constitui a estrutura básica dos modelos de *hardware* e/ou *software* das camadas do modelo OSI da B-ISDN. Cada camada possui um nome, um tipo, um ou mais parâmetros, uma tabela de dados, vários processos, um módulo gerenciador de parâmetros, um módulo gerenciador de tabela de dados, um módulo de inicialização, um módulo de execução e um módulo de terminação. Um processo consiste de uma série de tarefas predeterminadas, como por exemplo armazenar uma célula em um sistema de fila, ou ainda fragmentar uma PDU em várias células. Esses processos determinam a funcionalidade de uma camada e são disparados a partir dos eventos. Uma camada pode agendar seus próprios eventos na fila, programando assim o instante de tempo em que seus processos serão executados. Cada camada pertence a um determinado bloco que repassa os eventos que lhe são destinados.

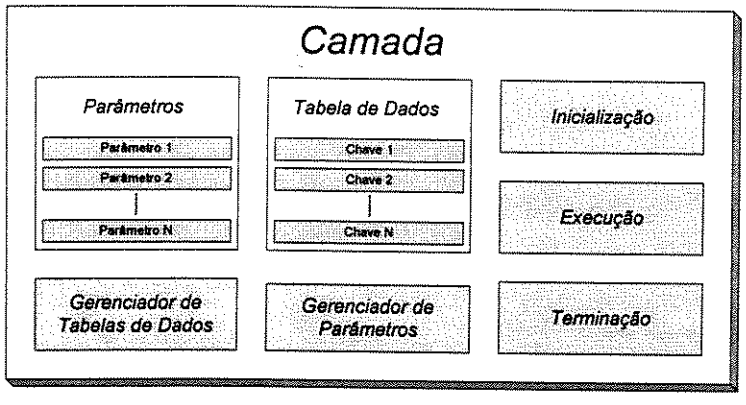


Figura 6.15 – Estrutura das camadas do SimATM

O módulo gerenciador de parâmetros permite consultar, modificar e remover os parâmetros do bloco e da rede ATM a que a camada pertence. O módulo gerenciador de tabelas de dados permite consultar, modificar e remover os dados da tabela do bloco a que a camada pertence. O módulo gerenciador de tabelas permite ainda realizar procuras na tabela deste bloco. Os módulos de inicialização, execução e terminação são definidos nos modelos que herdam a estrutura básica **camada**.

6.13 - Modelos de Camadas

Os modelos de camadas do **SimATM** foram desenvolvidos a partir da estrutura básica **camada**. Dois tipos de modelos foram desenvolvidos: modelos de camadas para os equipamentos ATM e modelos de camadas para o aplicativo ATM. Os modelos de camada para os equipamentos ATM são: Camada de Adaptação ATM Tipo 5, Camada ATM do BTE, Camada ATM do *Switch* e Camada Física para a Interface Baseada em Células. A razão para a separação da camada ATM em dois modelos está na considerável diferença funcional entre a camada ATM presente no equipamento BTE e no equipamento chaveador. Os modelos de camadas para o aplicativo ATM são: Fonte de Tráfego CBR, Fonte de Tráfego VBR Surtuoso, Fonte de Tráfego Genérica e Receptor de Tráfego. No plano de usuário do modelo de protocolos da B-ISDN, apresentado no item 2.3., as camadas acima da camada de adaptação ATM são chamadas camadas superiores. Estas camadas abrigam pilhas de protocolos e funções específicas de usuários dos serviços providos pela rede ATM. No **SimATM**, estas pilhas de protocolos e funções são modeladas como fontes de tráfego. Cada fonte de tráfego é implementada no **SimATM** como um modelo de camada para o aplicativo ATM do simulador. Estes modelos de camadas são os geradores e receptores de tráfego da rede, e foram desenvolvidos inspirados nas fontes de tráfego do aplicativo ATM do simulador do NIST, anteriormente descrito no Capítulo 5. A seguir descreveremos os modelos de camadas do **SimATM**.

6.13.1 - Modelos de Camadas para os Equipamentos ATM

6.13.1.1 - Camada de Adaptação ATM Tipo 5

A versão atual do **SimATM** somente dispõe de um modelo para os protocolos da camada de adaptação ATM: o modelo da AAL 5. Modelos para outras AALs serão desenvolvidos em versões futuras do simulador. A AAL 5 foi escolhida como o primeiro modelo de AAL a ser implementada devido a sua popularidade. Algumas das razões que levaram a tal popularidade foram citadas anteriormente no item 2.3.4.5.

6.13.1.1.1 - Descrição do Modelo

O modelo da camada AAL tipo 5 foi desenvolvido de acordo com a Recomendação I.363 do ITU-T, anteriormente descrita no item 2.3.4.5. Este modelo constitui o modelo de *hardware* para a camada de adaptação ATM tipo 5 do modelo de referência de protocolos da

B-ISDN (B-ISDN PRM), e implementa parcialmente as funções de transferência e de gerenciamento de camadas correspondentes a camada AAL da arquitetura funcional geral de um elemento de rede ATM, apresentada no Capítulo 3.

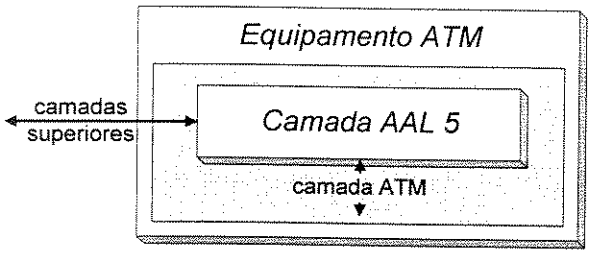


Figura 6.16 – Contexto do modelo da camada AAL 5 em um equipamento ATM

6.13.1.1.2 - Modelo Funcional

A Figura 6.17 mostra uma comparação entre o modelo funcional da AAL 5 apresentado no anexo E da Recomendação I.363 [6] e o modelo funcional da AAL 5 implementada no **SimATM**.

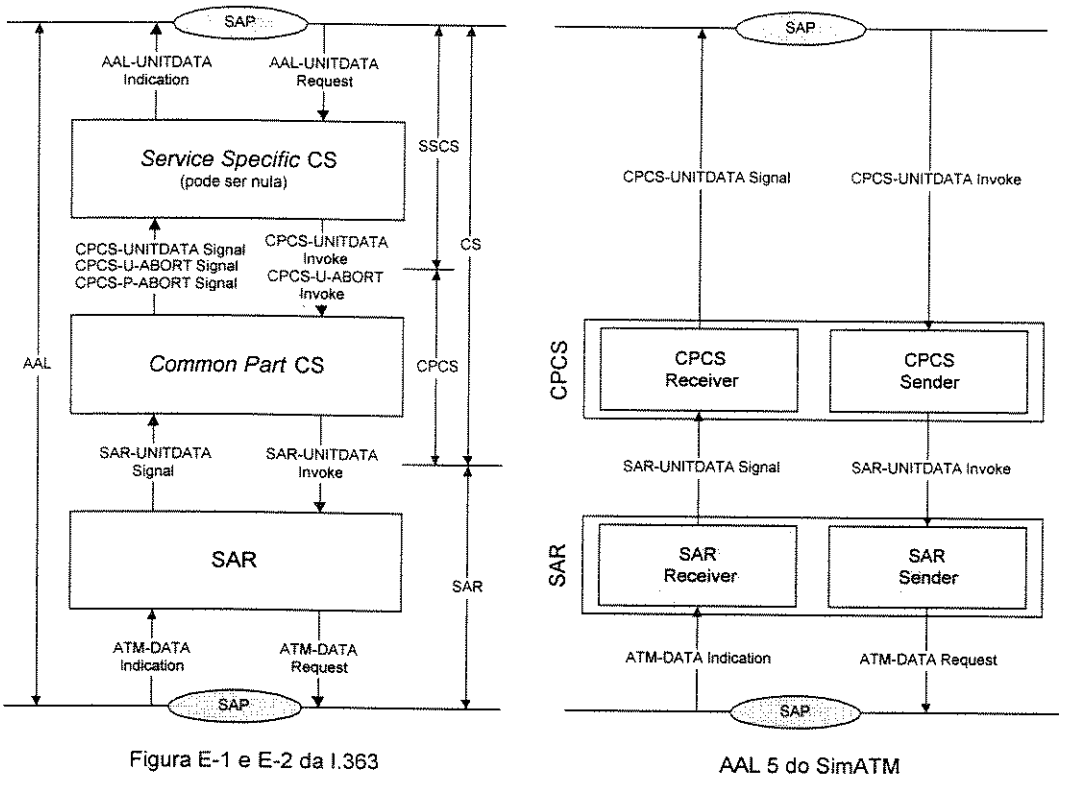


Figura 6.17 – Modelo funcional da AAL 5 na recomendação I.363 e no SimATM

O modelo da AAL 5 do **SimATM** possui a subcamada de convergência de serviços específicos (SSCS – *Service Specific Convergence Sublayer*) nula. Neste caso, as primitivas

trocadas com as camadas superiores a AAL 5 são as próprias primitivas da subcamada de convergência de serviços comuns (CPCS – *Common Part Convergence Sublayer*).

São implementadas todas as primitivas que transportam informações de usuário entre a camada AAL e a camada ATM e entre a camada AAL e as camadas superiores.

As funções da CPCS e da subcamada de segmentação e remontagem (SAR – *Segmentation and Reassembly Sublayer*) são particionadas em quatro procedimentos: CPCS *Sender*, SAR *Sender*, SAR *Receiver* e CPCS *Receiver*. Os dois primeiros agrupam as funções no sentido de transmissão de dados, enquanto os dois últimos agrupam as funções no sentido de recepção.

Conforme apresentamos no Capítulo 2, a AAL 5 suporta dois modos de serviço: modo de serviço de mensagens e modo de serviço de fluxo, bem como dois tipos de transmissão: assegurada e não assegurada. O modelo da AAL 5 do **SimATM** suporta somente o modo de serviço de mensagem com transmissão não assegurada.

6.13.1.1.3 - Parâmetros

O modelo da AAL 5 possui somente um parâmetro: atraso de encapsulamento de célula (*CellPacketizationDelay*). Toda vez que uma SAR-SDU (CPCS-PDU) é segmentada em várias SAR-PDUs, é incluído o atraso de encapsulamento dos 48 *bytes* de informação contidos na SAR-SDU, que são usados para montar cada uma das SAR-PDUs. O valor *default* é 1ns.

6.13.1.1.4 - Dados de Tabela

O modelo da AAL 5 não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.1.1.5 - Funcionamento

A Figura 6.18 ilustra o funcionamento do modelo da AAL 5 no sentido de transmissão de dados. Uma camada geradora de tráfego, pertencente a um aplicativo ATM, gera as TS-PDUs (*Traffic Source* PDUs) que serão enviadas para a camada AAL 5 de um equipamento terminal ATM. Cada TS-PDU é acomodada em uma primitiva CPCS-UNITDATA *Invoke*, que é enviada para a camada AAL 5 de destino através de um evento. Na AAL 5 de destino, o procedimento CPCS *Sender* retira do evento a primitiva CPCS-UNITDATA *Invoke* e utiliza os campos desta primitiva para gerar uma CPCS-PDU.

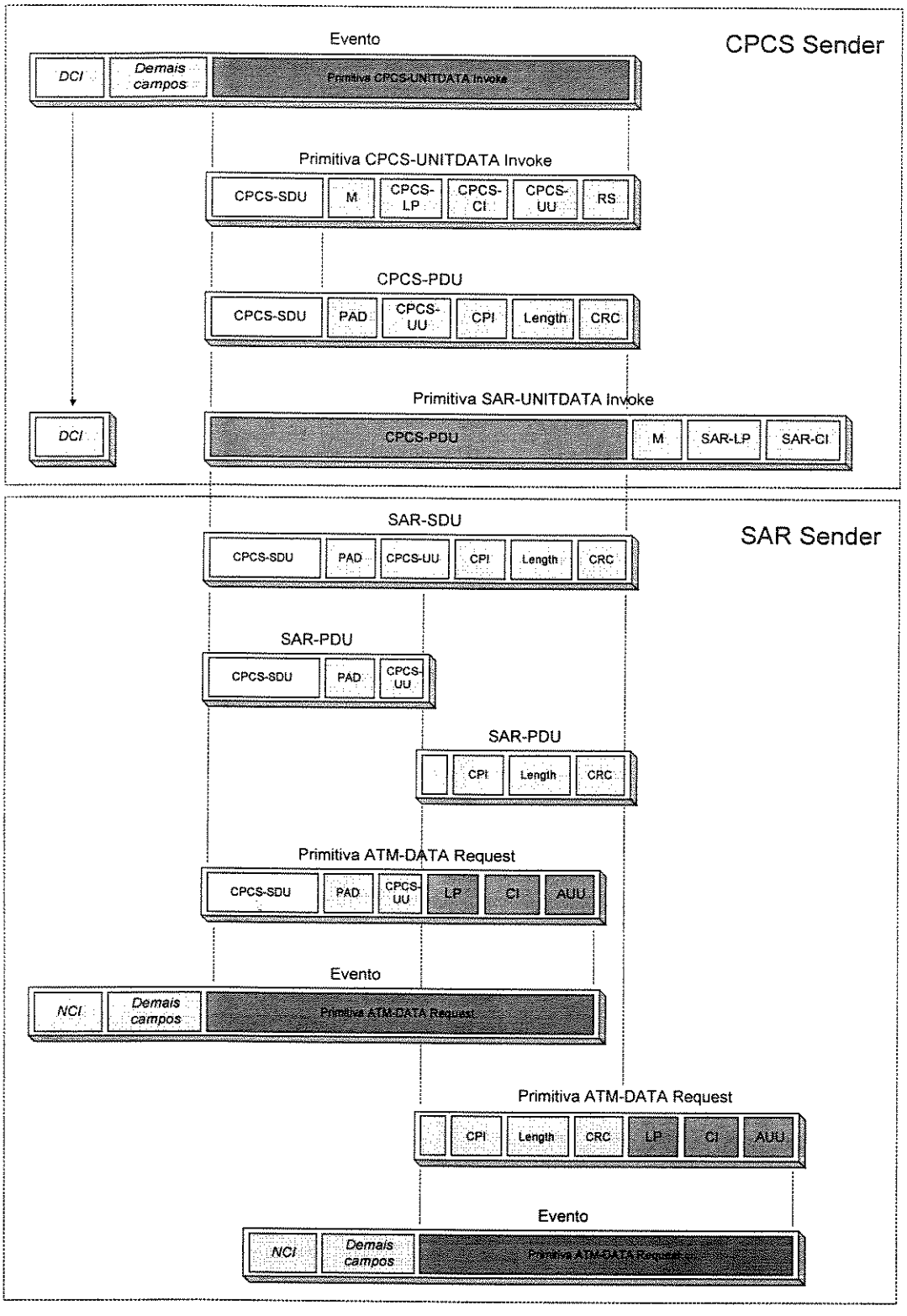


Figura 6.18 – Funcionamento do modelo da AAL 5 no sentido de transmissão de dados

O evento recebido contém um identificador de conexão de dados (DCI – *Data Connection Identifier*), que é utilizado para identificar a qual conexão de dados pertencem as informações de usuário que estão sendo transmitidas. O campo *Payload* desta CPCS-PDU

passa a acomodar a TS-PDU (CPCS-SDU) da camada geradora de tráfego. A CPCS-PDU possui um campo de enchimento (PAD) que é utilizado para adaptar o tamanho da CPCS-PDU a um múltiplo de 48 *bytes*. Então, a CPCS-PDU é passada da subcamada CPCS para a subcamada SAR através da primitiva SAR-UNITDATA *Invoke*. Neste caso não é utilizado um evento, a passagem é feita através de chamada a função.

Na subcamada SAR, o procedimento SAR *Sender* retira da primitiva SAR-UNITDATA *Invoke* a SAR-SDU (CPCS-PDU) e fragmenta-a em SAR-PDUs de 48 *bytes*. Cada SAR-PDU é enviada para a camada ATM em uma primitiva ATM-DATA *Request,* por meio de um evento. A subcamada SAR utiliza os campos da primitiva SAR-UNITDATA *Invoke* para configurar os campos da primitiva ATM-DATA *Request*. O campo AUU da primitiva ATM-DATA *Request* é utilizado para indicar se uma SAR-PDU corresponde a última porção da SAR-SDU. A subcamada SAR não acrescenta nenhuma informação de controle de protocolo (PCI). O evento enviado para a camada ATM possui um identificador de conexão de rede (NCI – *Network Connection Identifier*) que é obtido a partir de uma consulta à tabela de dados do bloco a que a camada pertence. Esta consulta é realizada a partir do identificador de conexão de dados conhecido.

A Figura 6.19 ilustra o funcionamento do modelo da AAL 5 no sentido de recepção de dados. Uma camada ATM envia o *payload* de cada célula de usuário recebida para a camada AAL 5 do mesmo equipamento. Isto é feito utilizando-se uma primitiva ATM-DATA *Indication*, que é entregue a AAL 5 de destino por meio de um evento. Os campos de cabeçalho de cada célula são utilizados para configurar a primitiva. Na AAL 5 de destino, o procedimento SAR *Receiver* retira a primitiva ATM-DATA *Indication* do evento e utiliza os campos desta primitiva para configurar uma primitiva SAR-UNITDATA *Signal*. O campo *More* da primitiva SAR-UNITDATA *Signal* é utilizado para indicar se uma SAR-PDU recebida corresponde a última porção da SAR-SDU original. Então, cada SAR-PDU é passada da subcamada SAR para a subcamada CPCS através da primitiva SAR-UNITDATA *Signal*. Neste caso também não é utilizado um evento. Na subcamada CPCS, o procedimento CPCS *Receiver* retira da primitiva SAR-UNITDATA *Signal* cada uma das parcelas da CPCS-PDU original. Quando a última parcela é recebida, a CPCS-PDU original é remontada. Então, a CPCS-SDU é retirada da CPCS-PDU e enviada em uma primitiva CPCS-UNITDATA *Signal* para uma camada receptora de tráfego de um aplicativo ATM de destino. A primitiva CPCS-UNITDATA *Signal* é entregue ao receptor de tráfego por meio de um evento.

Na versão atual do **SimATM**, se uma ou mais células ATM forem descartadas devido a um congestionamento, a CPCS-PDU original só será remontada se sua última parcela tiver sido recebida. Neste caso, a camada receptora de tráfego no aplicativo ATM de destino será avisada sobre a corrupção na TR-PDU (*Traffic Receiver* PDU) recebida.

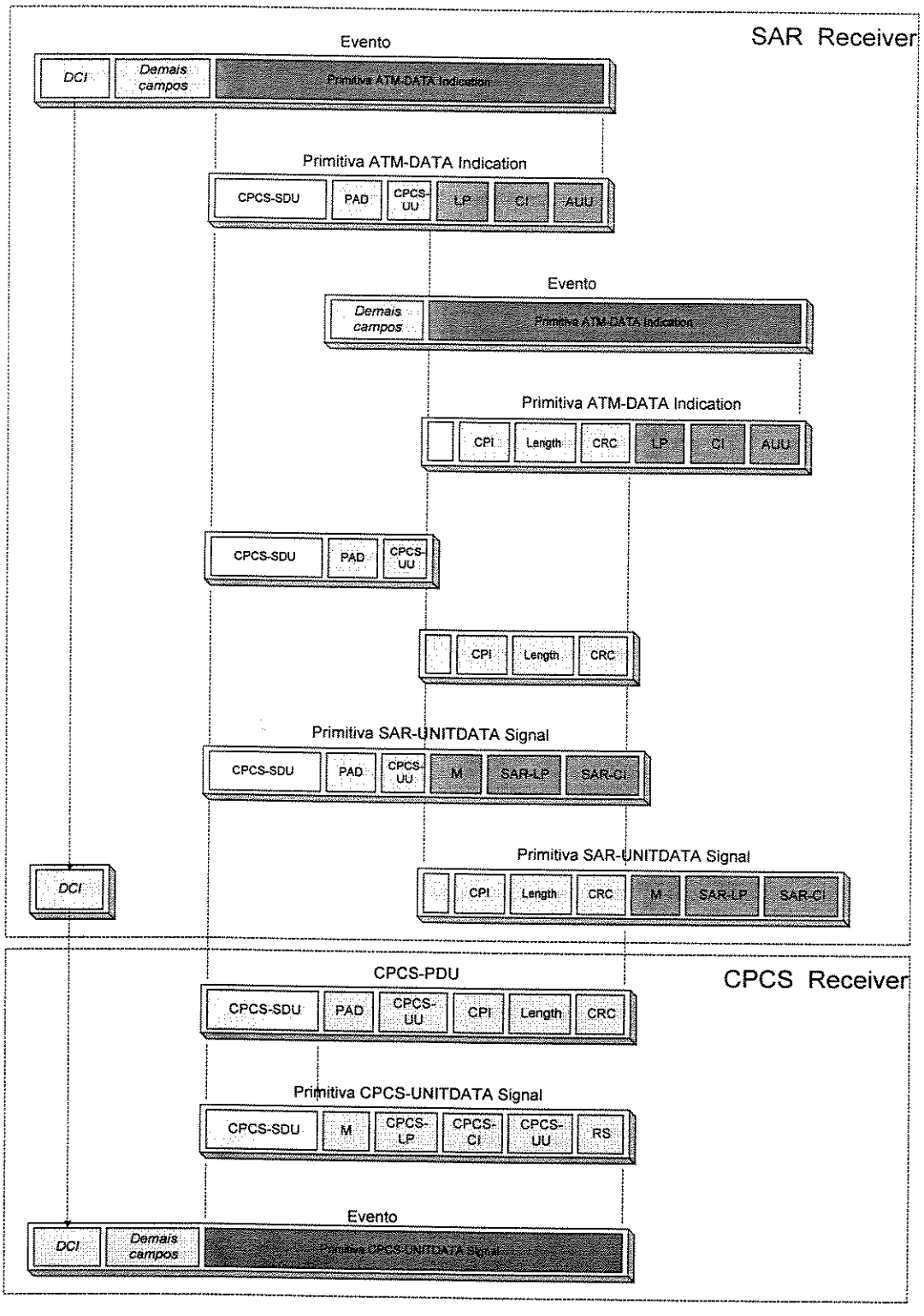


Figura 6.19 – Funcionamento do modelo da AAL 5 no sentido de recepção de dados

6.13.1.1.6 - Processos

O modelo da AAL 5 possui os processos: CPCS_UNITDATA_INVOKE e ATM_DATA_INDICATION. Estes processos são implementados no módulo de execução do modelo e são responsáveis pela execução dos procedimentos CPCS *Sender*, SAR *Sender*, SAR *Receiver* e CPCS *Receiver*. A Figura 6.20 mostra um fluxograma dos processos do modelo da AAL 5. Os fluxogramas completos dos processos do modelo da camada AAL 5 encontram-se no Apêndice A.

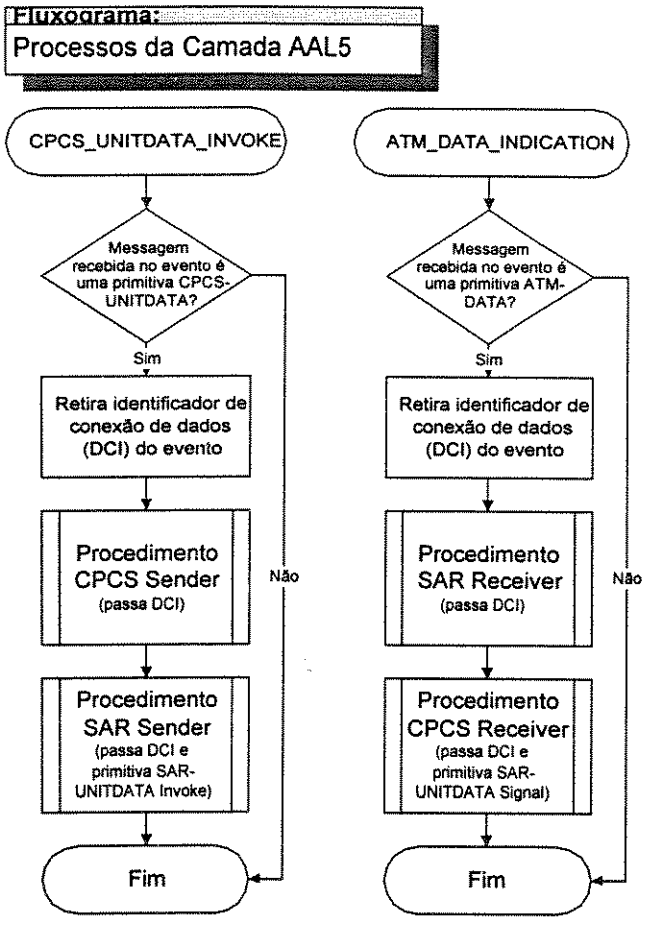


Figura 6.20 – Fluxograma dos processos da camada AAL 5

6.13.1.2 - Camada ATM do Terminal Faixa Larga

A versão atual do **SimATM** possui dois modelos de camada ATM: camada ATM do terminal faixa larga (BTE) e camada ATM do chaveador (*Switch*). Como já abordamos anteriormente, a razão para esta larga separação está na considerável diferença funcional entre a camada ATM do terminal faixa e a camada ATM do chaveador.

6.13.1.2.1 - Descrição do Modelo

O modelo da camada ATM do BTE foi desenvolvido de acordo com as Recomendações: I.361, I.150 e I.610 do ITU-T, anteriormente descritas no Capítulo 2. Este modelo constitui o modelo parcial de *hardware* para a camada ATM do modelo de referência de protocolos da B-ISDN (B-ISDN PRM), e implementa parcialmente as funções de transferência e de gerenciamento de camadas correspondentes a camada ATM da arquitetura funcional geral de um elemento de rede ATM.

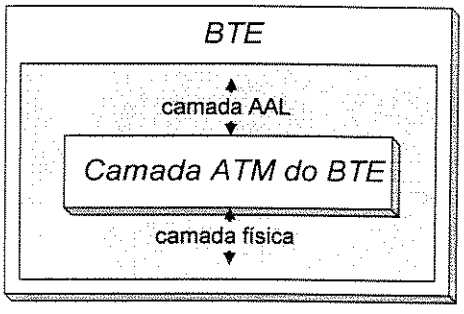


Figura 6.21 – Contexto do modelo da camada ATM do BTE em um equipamento terminal faixa larga

6.13.1.2.2 - Modelo Funcional

A Figura 6.22 mostra uma comparação entre o modelo funcional da camada ATM para o plano de usuário, interpretado a partir da Recomendação I.361 [7], e o modelo funcional da camada ATM do BTE implementado no **SimATM**.

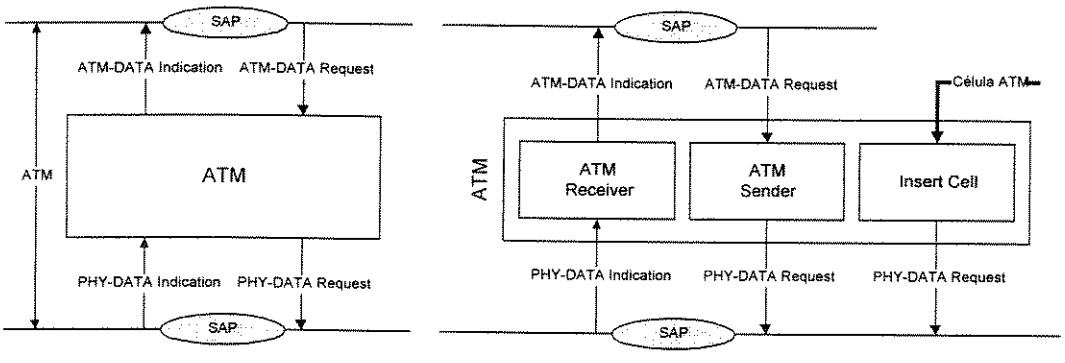


Figura 6.22 – Modelo funcional da camada ATM do SimATM

As funções executadas pela camada ATM são particionadas em dois procedimentos: ATM *Sender* e ATM *Receiver*. O primeiro agrupa as funções no sentido de transmissão de dados, enquanto o segundo agrupa as funções no sentido de recepção de dados. Um terceiro procedimento chamado *Insert Cell* possui uma fração das funções do procedimento ATM *Sender*, e tem por objetivo possibilitar a inserção de células de OAM (*Operation Administration and Maintenance*), RM (*Resource Management*) e de metasinalização, junto

ao fluxo de células ATM de usuário. Entretanto, a versão atual do modelo da camada ATM do BTE não insere tais células, apenas provê esta funcionalidade para versões futuras.

São implementadas todas as primitivas trocadas no plano de usuário entre a camada ATM e a camada física e entre a camada ATM e a camada de adaptação ATM.

6.13.1.2.3 - Parâmetros

O modelo da camada ATM do BTE possui os seguintes parâmetros: estado da amostragem de variáveis por célula (*TriggerByCellStatus*), estado do cálculo e amostragem de variáveis estatísticas (*SampleStatisticsStatus*) e intervalo de tempo entre amostras de variáveis estatísticas (*LoggingEveryXSecs*). Estes parâmetros são parâmetros de simulação. O parâmetro *TriggerByCellStatus* permite a amostragem para arquivo do tempo de permanência total de uma célula ATM na rede. Esta amostragem pode ser feita toda vez que uma célula chega a camada ATM de um equipamento de egresso da rede. O valor *default* deste parâmetro é "OFF" (desligado). O parâmetro *SampleStatisticsStatus* habilita ou não o cálculo da média e da variância do tempo de permanência de células na rede. Toda vez que uma célula chega a camada ATM de um equipamento de egresso da rede, as variáveis estatísticas são atualizadas se o parâmetro *TriggerByCellStatus* estiver habilitado. O valor *default* deste parâmetro também é "OFF" (desligado). O parâmetro *LoggingEveryXSecs* configura o intervalo de tempo entre amostras de variáveis estatísticas para arquivo. O valor *default* é 1ms. Este parâmetro só é utilizado quando o parâmetro *SampleStatisticsStatus* = "ON" (ligado).

6.13.1.2.4 - Arquivos de Saída

O modelo de camada ATM do BTE, de acordo com o estado dos seus parâmetros, pode possuir até dois arquivos de saída: um arquivo de amostragem de variáveis por chegada de célula e um arquivo de amostragem de variáveis estatísticas. Estes arquivos também possuem a extensão *.dat*.

6.13.1.2.5 - Dados de Tabela

O modelo da camada ATM do BTE não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.1.2.6 - Funcionamento

A Figura 6.23 ilustra o funcionamento do modelo da camada ATM do BTE no sentido de transmissão de dados. Os dados de usuário chegam a camada ATM do BTE através de eventos enviados a partir da camada de adaptação ATM. Estes eventos contém a primitiva

ATM-DATA *Request* e um identificador de conexão de rede ATM (NCI – *Network Connection Identifier*). O NCI identifica a qual conexão preestabelecida pertence os dados de usuário que estão sendo transmitidos pela rede. O procedimento ATM *Sender* retira esta primitiva do evento e utiliza os seus campos para configurar o cabeçalho de um célula ATM. O campo *Interface Data* da primitiva ATM-DATA *Request* contém a ATM-SDU de 48 bytes que será acomodada no *payload* da célula ATM. Os campos VPI e VCI da célula são configurados a partir de uma consulta feita com o NCI na tabela de dados do BTE. A célula ATM é então acomodada no campo *Interface Data* de uma primitiva PHY-DATA *Request*. Esta primitiva é então enviada para a camada física do BTE, por meio de um evento.

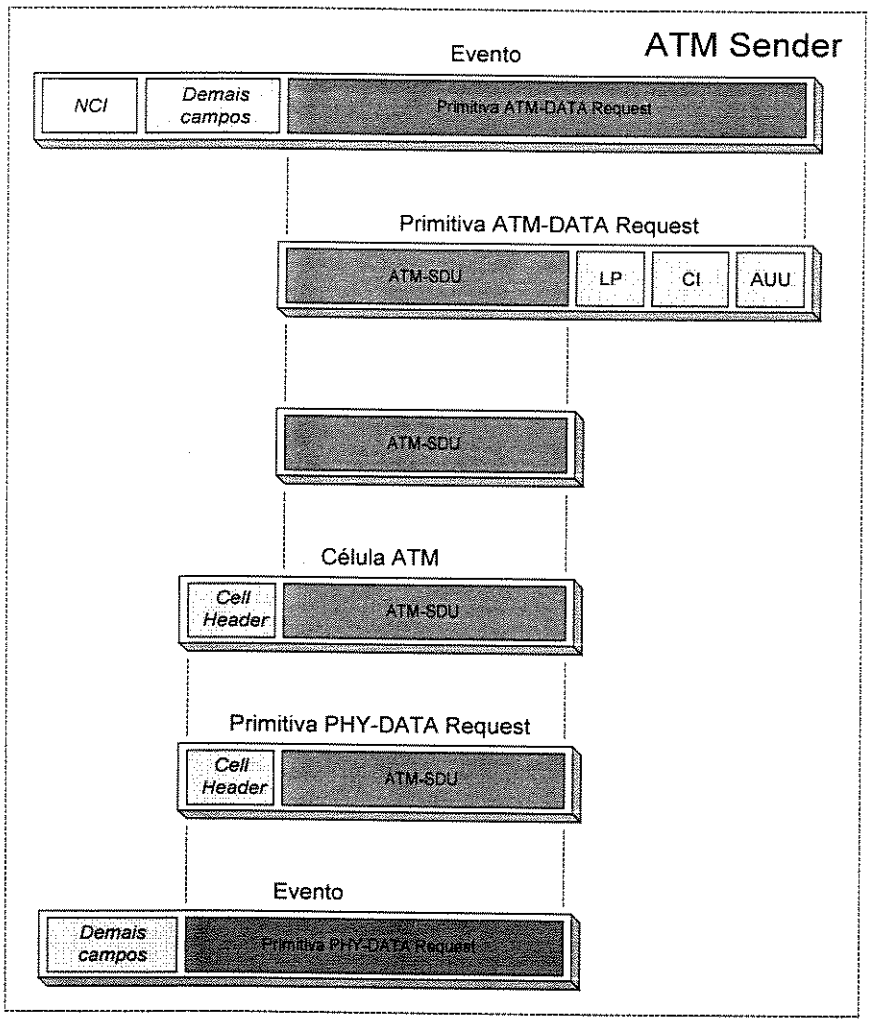


Figura 6.23 – Funcionamento do modelo da camada ATM do BTE no sentido de transmissão de dados

O funcionamento do modelo da camada ATM do BTE no sentido de recepção de dados é ilustrado na Figura 6.24. Os dados de usuário chegam a camada ATM do BTE de egresso da rede através de eventos enviados a partir da camada de física. Estes eventos

contém a primitiva PHY-DATA *Indication*. O procedimento ATM *Receiver* retira esta primitiva do evento e a célula ATM do campo *Interface Data* desta primitiva. O procedimento ATM *Receiver* verifica ainda se a célula recebida é uma célula de OAM, RM ou de metasinalização. Se a célula recebida for de qualquer um destes tipos, ela será encaminhada para um procedimento de tratamento específico. Na versão atual do modelo desta camada, estes procedimentos apenas retiram de circulação estas células. O objetivo é permitir que versões futuras implementem toda a funcionalidade destes procedimentos.

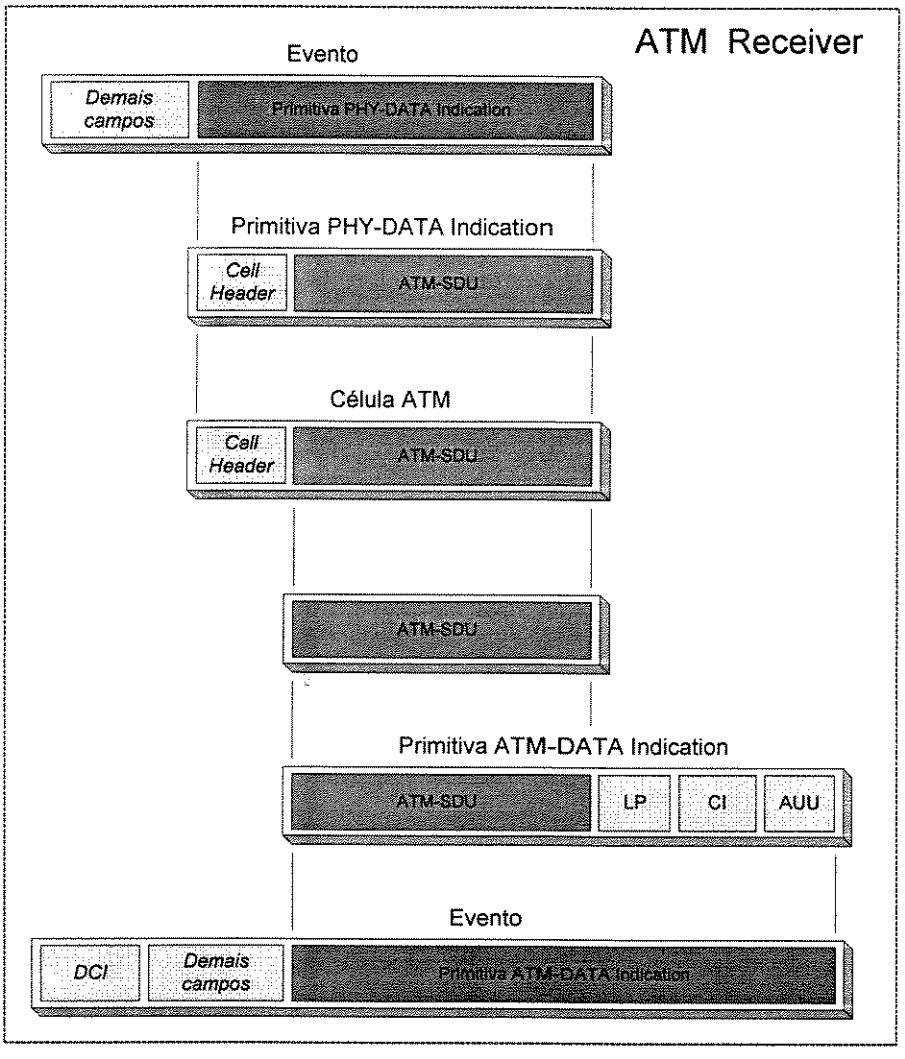


Figura 6.24 – Funcionamento do modelo da camada ATM do BTE no sentido de recepção de dados

Caso a célula recebida seja uma célula de usuário, o procedimento ATM *Receiver* irá realizar uma série de consultas na tabela de dados do BTE antes de enviá-la para a camada de adaptação de destino. A primeira consulta visa descobrir o identificador de conexão de rede (NCI) associado a esta célula de usuário. Para isso, é feita uma consulta utilizando-se os campos VPI e VCI da célula recebida. A segunda consulta visa descobrir o identificador de

conexão de dados (DCI) associado a conexão de rede identificada pelo NCI. A última consulta visa descobrir o nome da camada de adaptação de destino associado a conexão de dados identificada pelo DCI. A ATM-SDU é então acomodada em uma primitiva ATM-DATA *Indication*, que será enviada para a AAL de destino através de um evento. Este evento contém o identificador DCI. Os campos da primitiva ATM-DATA *Indication* são configurados a partir do cabeçalho da célula ATM. O procedimento ATM *Receiver* realiza ainda, a depender do estado do parâmetro de simulação *TriggerByCellStatus*, a amostragem para arquivo do tempo de permanência desta célula na rede.

6.13.1.2.7 - Processos

O modelo da camada ATM do BTE possui quatro processos: ATM_DATA_REQUEST, PHY_DATA_INDICATION, INSERT_ATMCELL e LOG_SAMPLE_STATISTICS. Estes processos são implementados no módulo de execução do modelo e são responsáveis pela execução dos procedimentos ATM *Sender*, ATM *Receiver*, *Insert Cell* e de um outro procedimento responsável pela amostragem para arquivo das variáveis estatísticas de permanência de células na rede. Este procedimento é chamado *Log Sample Statistics*. A Figura 6.25 mostra um fluxograma dos processos do modelo da camada ATM do BTE. Os fluxogramas completos dos processos do modelo da camada ATM do BTE encontram-se no Apêndice A.

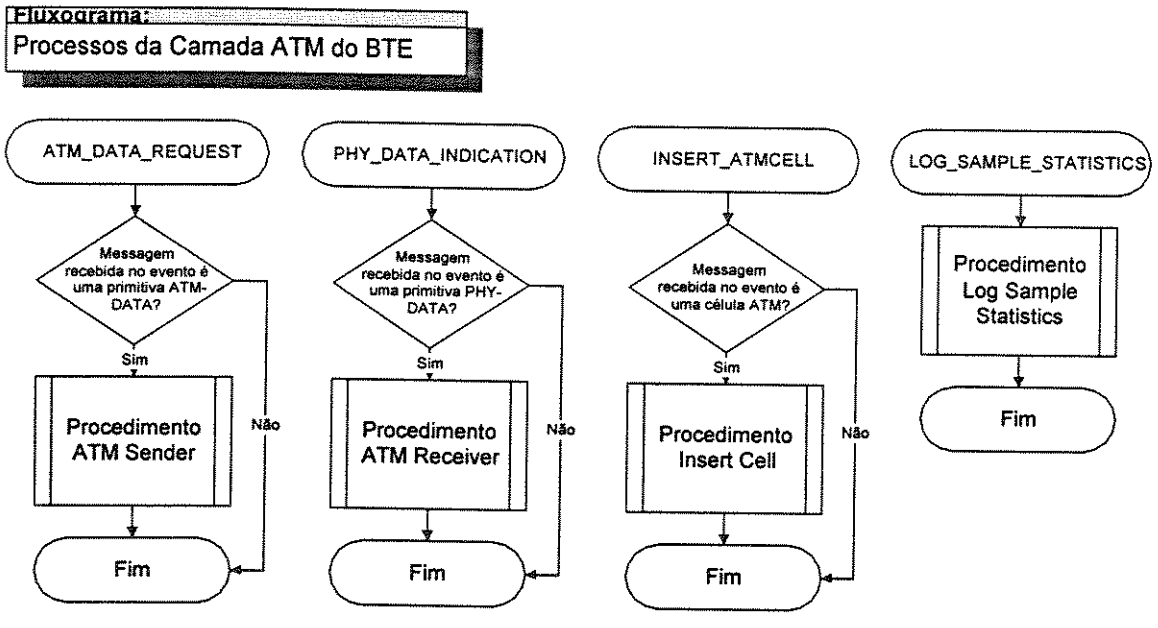


Figura 6.25 – Fluxograma dos processos da camada ATM do BTE

6.13.1.3 - Camada Física para a Interface Baseada em Células

A versão atual do **SimATM** dispõem somente de um modelo de camada física: camada física para a interface baseada em células. Outros modelos de camada física, como por exemplo a camada física para a interface SDH (*Synchronous Digital Hierarchy*) ou a camada física para a interface PDH (*Plesiochronous Digital Hierarchy*), poderão ser desenvolvidas para futuras versões do simulador. A camada física para a interface baseada em células foi escolhida como primeira opção de modelo de camada física devido a sua simplicidade.

6.13.1.3.1 - Descrição do Modelo

O modelo da camada física para a interface baseada em células foi desenvolvido de acordo com a Recomendações: I.432 e I.610 do ITU-T, anteriormente descritas nos itens 2.3.2.4 e 2.3.6.3. Este modelo constitui o modelo de *hardware* para a camada física do modelo de referência de protocolos da B-ISDN (B-ISDN PRM), e implementa parcialmente as funções de transferência e de gerenciamento de camadas correspondentes à camada física da arquitetura funcional geral de um elemento de rede ATM.

As instâncias do modelo da camada física para a interface baseada em células são conectadas entre si através de enlaces físicos (fibra ótica, cabo coaxial, etc.), e fazem parte da estrutura dos equipamentos ATM da rede. Chamaremos de **porta** cada instância de modelo da camada física que se conecta através de um enlace. E cada porta possui uma instância de modelo de sistema de fila associado. O modelo de sistema de fila é responsável pelo armazenamento e transmissão de células ATM.

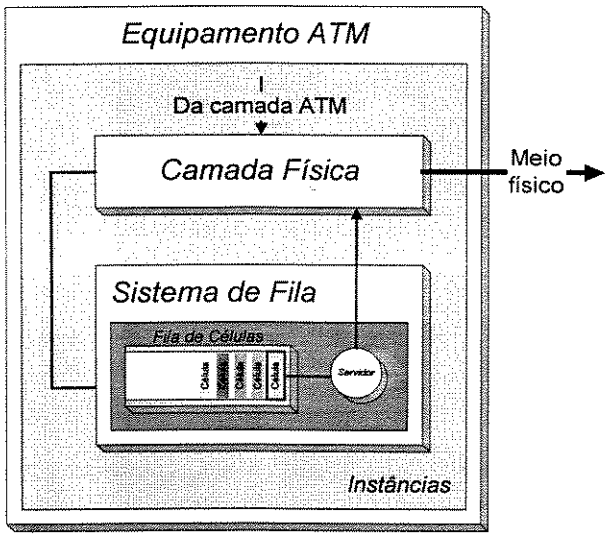


Figura 6.26 – Contexto do modelo da camada física em um equipamento ATM

6.13.1.3.2 - Modelo Funcional

A Figura 6.27 mostra uma comparação entre o modelo funcional da camada física para o plano de usuário, interpretado a partir da Recomendação I.361 [7] , e o modelo funcional do modelo de camada física implementada no **SimATM**.

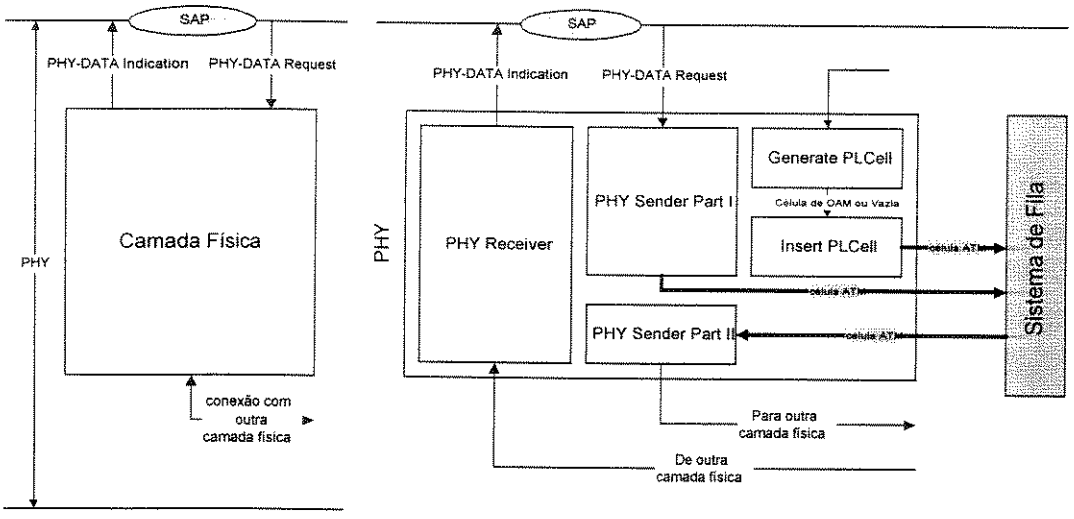


Figura 6.27 – Modelo funcional da camada física do SimATM

As funções executadas pela camada física são particionadas em três procedimentos: PHY *Sender Part I*, PHY *Sender Part II* e PHY *Receiver*. Os dois primeiros agrupam as funções no sentido de transmissão de dados, enquanto o terceiro agrupa as funções no sentido de recepção de dados. Um quarto procedimento chamado *Insert PLCell* possui uma fração das funções do procedimento PHY *Sender Part I*, e tem por objetivo possibilitar a inserção de células de OAM da camada física (PLOAM) e células vazias, junto ao fluxo de células de usuário. O modelo da camada física do **SimATM** possui ainda um quinto procedimento chamado *Generate PLCell*, que é responsável pela geração de células PLOAM e células vazias, de acordo com a freqüência estipulada na Recomendação I.432 do ITU-T.

São implementadas todas as primitivas trocadas no plano de usuário entre a camada física e a camada ATM.

6.13.1.3.3 - Interação com o Sistema de Fila Associado

Cada instância do modelo da camada física necessita associar-se com uma instância de modelo de sistema de fila. Como já abordamos no item 6.11 - Sistemas de Fila, o modelo do sistema de fila do **SimATM** é responsável pelo armazenamento das células ATM que aguardam por serviço e pelo modelamento da execução desses serviços nas células ATM.

A Figura 6.27 mostra a iteração do modelo da camada física com o seu sistema de fila associado. O procedimento PHY *Sender Part I* armazena as células ATM recebidas da camada ATM no sistema de fila associado e agenda eventos para encaminhar cada célula ATM ao serviço e/ou retirar células já servidas. No caso do modelo da camada física, o serviço executado pelo sistema de fila equivale à transmissão de uma célula ATM através de

um enlace até o próximo equipamento da rede. Somente uma célula ATM pode ser servida (transmitida) por vez. O procedimento *Insert PLCell* armazena células de OAM e células vazias no sistema de fila associado e agenda eventos para encaminhar estas células ao serviço e/ou retirar células já servidas. O procedimento PHY *Sender Part II* é acionado através dos eventos agendados pelo procedimento PHY *Sender Part I* e *Insert PLCell*, e é responsável por retirar as células que já foram servidas e encaminhar outras células ao serviço.

6.13.1.3.4 - Agendando a Retirada de Células de um Sistema de Fila

Como já abordamos no item 6.11 - Sistemas de Fila, o ritmo de encaminhamento de células ATM ao serviço em um sistema de fila é ditado através de solicitações de retirada de célula executadas pela instância de camada associada.

A Figura 6.28 mostra o fluxograma do esquema de agendamento de eventos utilizado pelos procedimentos PHY *Sender Part I* e *Insert PLCell*, para acionar o procedimento PHY *Sender Part II*.

O tempo de serviço de uma célula (ST – *Service Time*) no sistema de fila associado a camada física é fixo e igual a:

$$ST = \frac{X}{BitRate}$$

Onde:

- X é igual ao tamanho de uma célula em *bits* (424 *bits*)
- *BitRate* é igual a taxa de transmissão em *bits*/segundo.

O tempo de serviço de uma célula é igual a duração de um *frame* na interface da camada física.

As variáveis apresentadas no fluxograma da Figura 6.28 tem o seguinte significado:

- ESCRT (*Estimated System Cell Remove Time*) – Tempo estimado para a remoção de uma célula do sistema de fila.
- LRSCT (*Last Removed System Cell Time*) – Tempo da última célula removida do sistema de fila.
- *Time* – Instante de tempo da chegada de uma célula nos procedimentos PHY *Sender Part I* e *Insert PLCell*.

- *FramePeriod* – Período de duração de um *frame* na camada física. Equivale a janela de tempo necessária para transmitir *bit* a *bit* uma célula ATM através de um enlace.

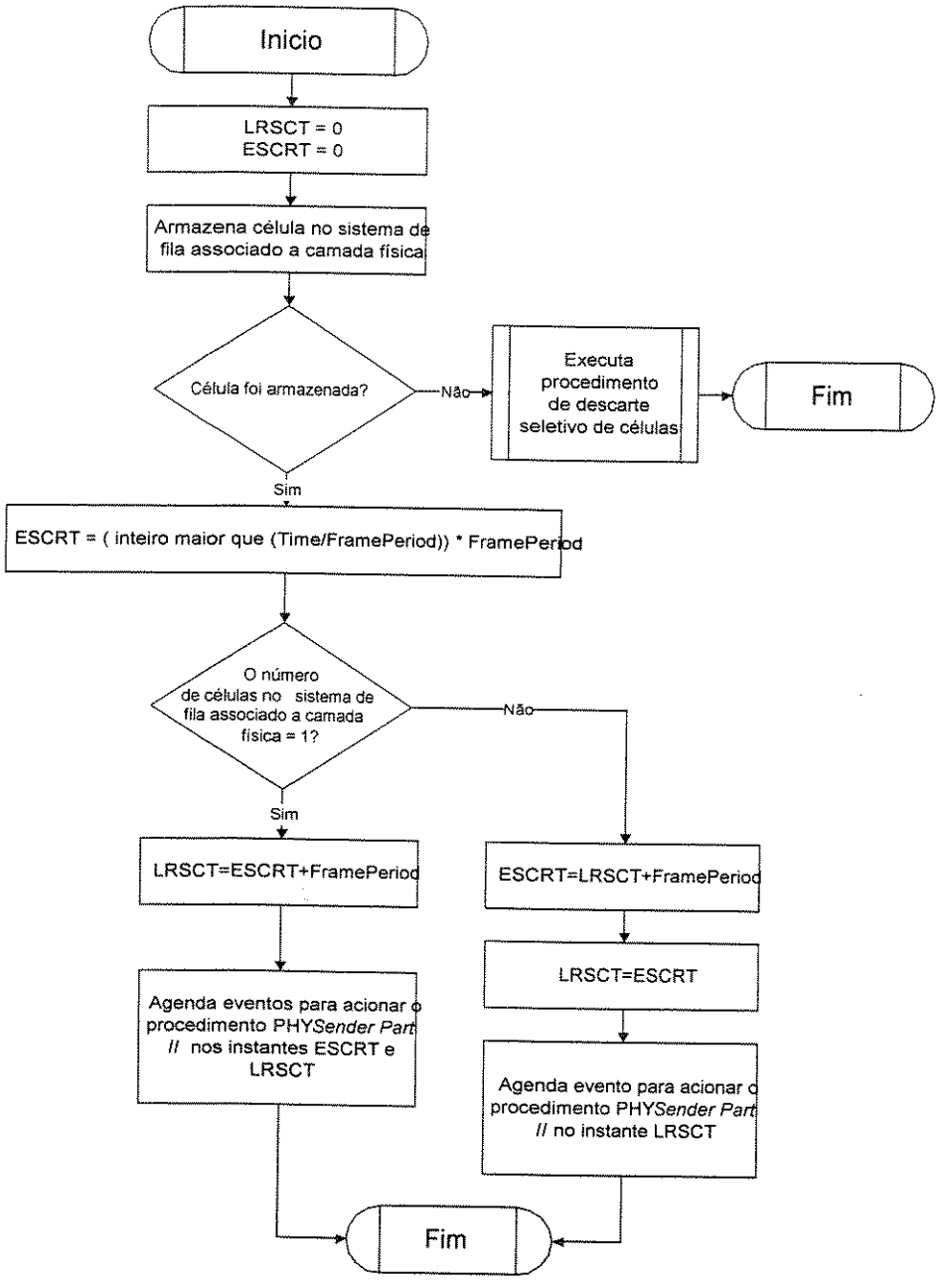


Figura 6.28 – Fluxograma do esquema de agendamento de eventos utilizados pelos procedimentos PHY *Sender Part I* e *Insert PLCell*

O funcionamento do esquema é o seguinte: toda vez que os procedimentos PHY *Sender Part I* ou *Insert PLCell* recebem uma célula ATM, eles tentam armazenar esta célula no sistema de fila associado a camada física. Se o *buffer* deste sistema de fila estiver cheio, a

célula será então encaminhada para um procedimento de descarte seletivo. Na versão atual do modelo da camada física para a interface baseada em células, este procedimento de descarte seletivo simplesmente descarta as células que não foram armazenadas no sistema de fila. Se a célula ATM foi armazenada com sucesso no sistema de fila associado, calcula-se então o tempo de inicio do próximo *frame* de transmissão e verifica-se o número de células presentes no sistema de fila (*buffer*+servidor). Se o número de células for igual a um, dois eventos precisam ser gerados para acionar o procedimento PHY *Sender Part II*: um para retirar a célula do *buffer* e encaminha-la para o servidor do sistema de fila, e outro para remover a célula deste servidor. O primeiro evento é gerado para o inicio do próximo *frame* (ESCRT), enquanto o segundo é gerado para um *frame* adiante do próximo *frame* (LRCST). Se o número de células for maior que 1 somente um evento precisa ser agendado para acionar o procedimento PHY *Sender Part II* e retirar a célula do sistema de fila, uma vez que quando a célula de prioridade imediatamente superior a esta célula for retirada do sistema, automaticamente esta célula será retirada do *buffer* e encaminhada para o servidor. O evento é agendado para o inicio do próximo *frame* a frente do tempo da última célula removida do sistema de fila (LRSCT).

6.13.1.3.5 - Parâmetros

O modelo da camada física do **SimATM** possui os seguintes parâmetros: taxa de transmissão em *bits*/segundo (*BitRate*), descrição do tipo de taxa (*SignalType*), distância até o próximo equipamento ATM em metros (*Distance*), velocidade de propagação do sinal no meio de transmissão em metros/segundo (*PropagationSpeed*), estado da transmissão de OAM e células vazias (*OAMTransmitionStatus*). Os parâmetros *BitRate, SignalType, Distance* e *PropagationSpeed* são parâmetros de design, enquanto o parâmetro *OAMTransmitionStatus* é um parâmetro de simulação. Para cada célula transmitida é acrescido um atraso de propagação igual a razão entre os parâmetros *PropagationSpeed* e *Distance*. Os valores *default* dos parâmetros são: *BitRate* = 155.52 *Mbps*, *SignalType* = "155.52 *Mbps* OC-3", *Distance* = 10 metros, *PropagationSpeed* = 300 x 10^6 metros/segundo e *OAMTransmitionStatus* = "ON" (ativado).

6.13.1.3.6 - Dados de Tabela

O modelo da camada física não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.1.3.7 - Funcionamento

A Figura 6.29 ilustra o funcionamento do modelo da camada física no sentido de transmissão de dados, enquanto a Figura 6.30 ilustra o funcionamento do modelo da camada física no sentido de recepção de dados.

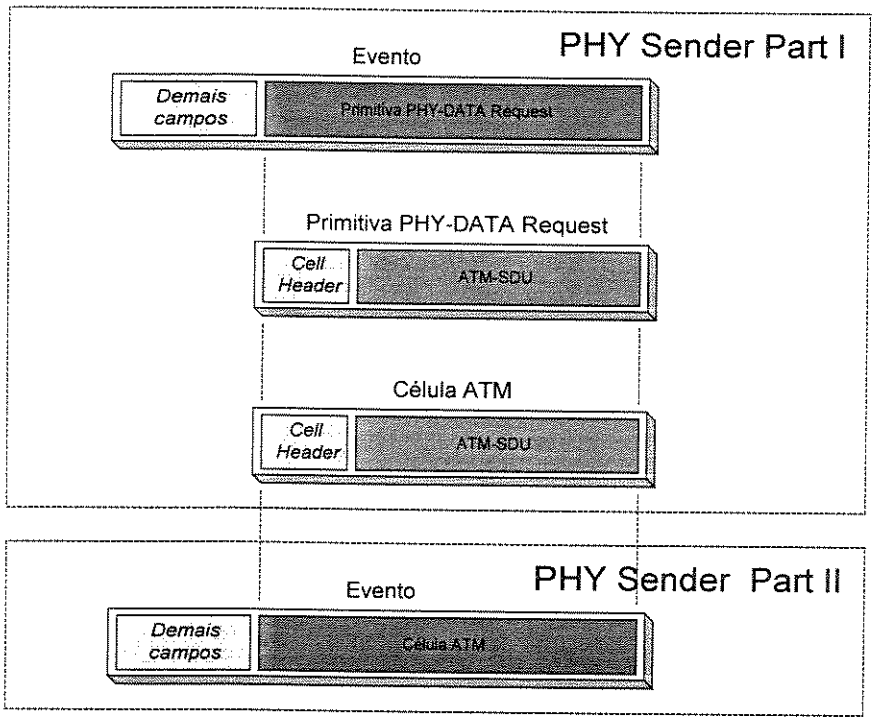


Figura 6.29 – Funcionamento do modelo da camada física no sentido de transmissão de dados

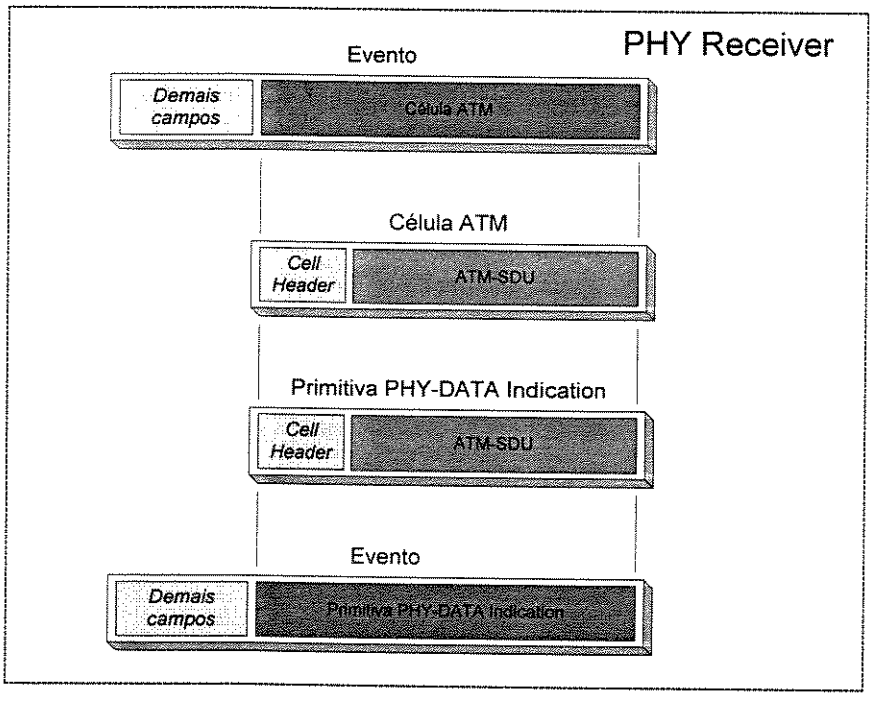


Figura 6.30 – Funcionamento do modelo da camada física no sentido de recepção de dados

As células ATM chegam à camada física através de eventos enviados a partir da camada ATM. Estes eventos contém a primitiva PHY-DATA *Request*. No procedimento PHY *Sender Part I* cada célula é retirada da primitiva e armazenada no sistema de fila associado a camada física. No procedimento PHY *Sender Part II* a célula é retirada do sistema de fila e alojada em um evento, que é enviado à camada física do próximo equipamento da rede. Na camada física deste equipamento, o procedimento PHY *Receiver* retira a célula do evento e aloja esta célula em uma primitiva PHY-DATA *Indication*. Esta primitiva é enviada através de um evento para a camada ATM deste equipamento.

6.13.1.3.8 - Processos

O modelo da camada física do **SimATM** possui cinco processos: PHY_DATA_REQUEST, PHY_SEND_CELL, RECEIVE_CELL, INSERT_PLCELL e PLCELL_LOOP. Estes processos são implementados no módulo de execução do modelo e são responsáveis pela execução dos procedimentos PHY *Sender Part I*, PHY *Sender Part II*, PHY *Receiver*, *Insert PLCell* e *Generate PLCell*. A Figura 6.31 mostra um fluxograma dos processos do modelo da camada física do **SimATM**. Os fluxogramas completos dos processos do modelo da camada física encontram-se no Apêndice A.

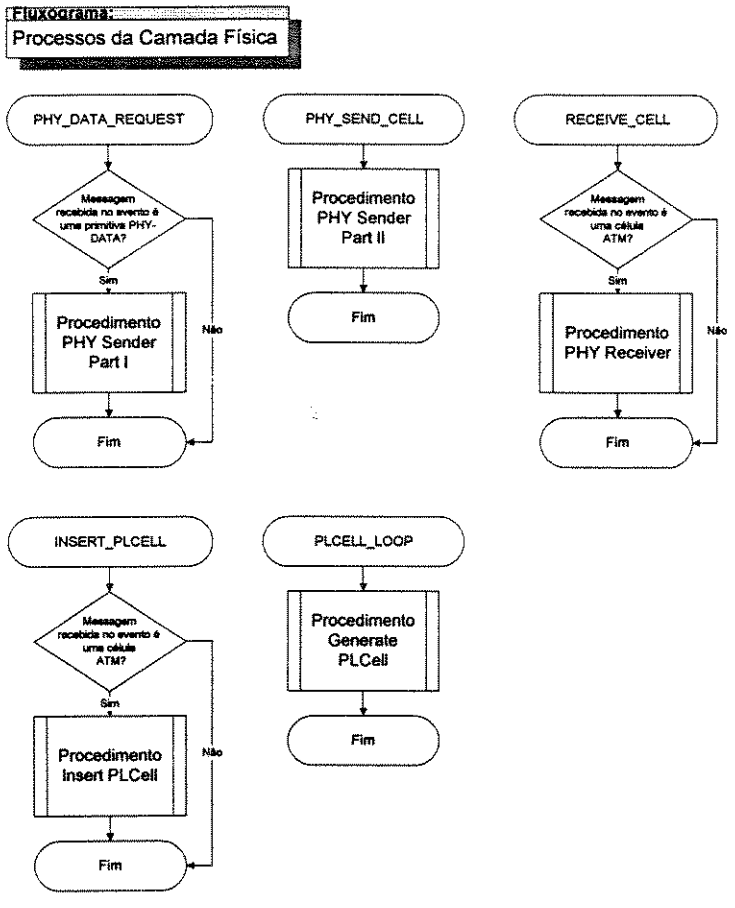


Figura 6.31 – Fluxograma dos processos do modelo da camada física

6.13.1.4 - Camada ATM do Chaveador

A versão atual do **SimATM** possui um modelo específico para a camada ATM do equipamento chaveador.

6.13.1.4.1 - Descrição do Modelo

Assim como o modelo da camada ATM do BTE, o modelo da camada ATM do chaveador foi desenvolvido de acordo com as Recomendações: I.361, I.150 e I.610 do ITU-T, anteriormente descritas no Capítulo 2. Este modelo constitui o modelo parcial de *hardware* para a camada ATM do modelo de referência de protocolos da B-ISDN (B-ISDN PRM), e implementa parcialmente as funções de transferência e de gerenciamento de camadas correspondentes a camada ATM da arquitetura funcional geral de um elemento de rede ATM.

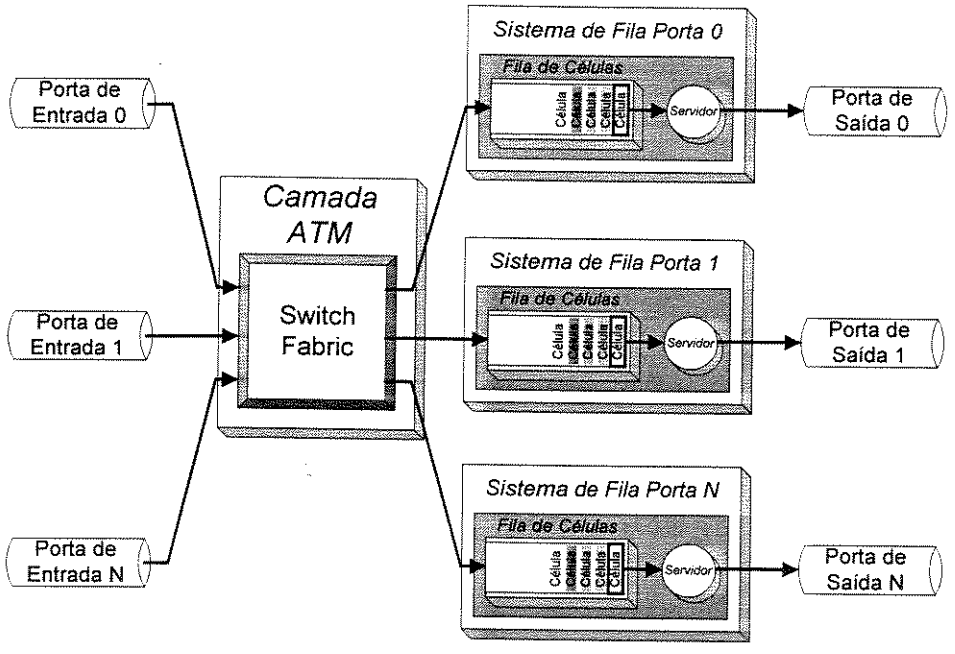


Figura 6.32 – Contexto do modelo da camada ATM em um chaveador ATM

O modelo da camada ATM do chaveador executa o roteamento das células ATM de uma porta de entrada para um porta de saída, passando por uma matriz de chaveamento (SF – *Switch Fabric*) e por um *buffer* de saída. Para cada porta do chaveador existe um sistema de fila associado, onde são armazenadas as células ATM. A *switch fabric* do modelo é perfeita, uma vez que nenhuma célula é perdida no seu interior e todas as células são retiradas dos *buffers* dos sistemas de fila a uma taxa constante igual ao tamanho de uma célula em *bits* (424 *bits*) dividido pelo tempo de ciclo (*SlotTime*). Todas as células são encaminhadas para as portas de saída em paralelo. A Figura 6.32 ilustra o modelo da camada ATM do chaveador.

6.13.1.4.2 - Modelo Funcional

A Figura 6.33 mostra uma comparação entre o modelo funcional da camada ATM para o plano de usuário, interpretado a partir da Recomendação I.361 [7], e o modelo funcional da camada ATM do chaveador implementado no **SimATM**.

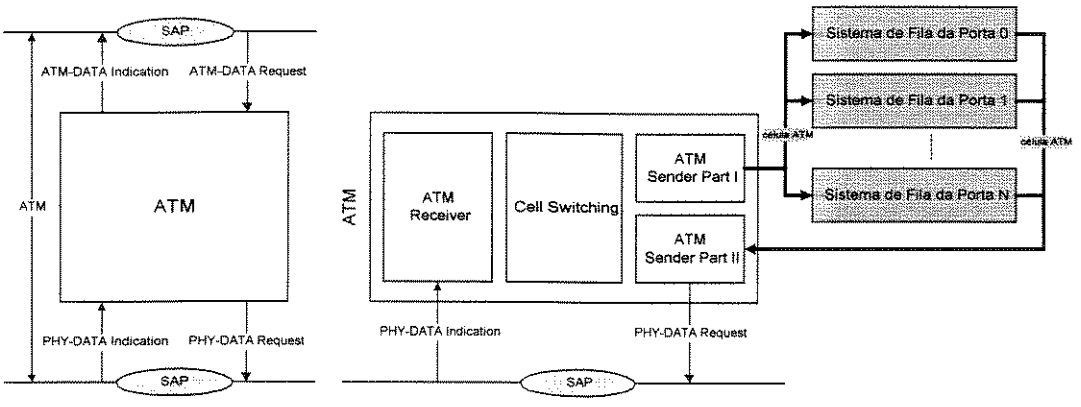


Figura 6.33 – Modelo funcional da camada ATM do chaveador

As funções executadas pela camada ATM do chaveador são particionadas em quatro procedimentos: ATM *Receiver*, *Cell Switching*, ATM *Sender Part I* e ATM *Sender Part II*. O procedimento ATM *Receiver* agrupa as funções de recepção de células ATM da camada física. O procedimento *Cell Switching* agrupa as funções relacionadas com o chaveamento de células ATM. Os procedimentos ATM *Sender Part I* e *Part II* agrupam as funções de transmissão de células com destino às várias instâncias de camadas físicas do chaveador (portas).

São implementadas todas as primitivas trocadas no plano de usuário entre a camada ATM e a camada física. As primitivas trocadas entre a camada ATM e a camada de adaptação ATM não são implementadas.

6.13.1.4.3 - Interação com os Sistemas de Fila Associados

Cada instância do modelo da camada ATM do chaveador necessita associar-se com várias instâncias do modelo de sistema de fila. O número de sistemas de fila associados é igual ao número de portas do chaveador. A Figura 6.33 mostra a iteração do modelo da camada ATM do chaveador com os seus sistemas de fila associados. O procedimento ATM *Sender Part I* armazena as células ATM destinadas a uma porta de saída no sistema de fila correspondente a esta porta, e agenda eventos para encaminhar cada célula ATM ao serviço e/ou retirar células já servidas. No caso do modelo da camada ATM do chaveador, o serviço executado pelo servidor do sistema de fila equivale a retirada de uma célula ATM do *buffer*

deste sistema e a entrega na camada física correspondente. Somente uma célula ATM pode ser retirada por vez em cada sistema de fila. Entretanto, no conjunto de sistemas de fila associados, várias células ATM são retiradas simultaneamente. O procedimento ATM *Sender Part II* é acionado através dos eventos agendados pelo procedimento ATM *Sender Part I*, e é responsável por retirar as células que já foram servidas e encaminhar outras células ao serviço.

6.13.1.4.4 - Agendando a Retirada de Células de um Sistema de Fila

Foi utilizado o mesmo esquema apresentado no item 6.13.1.3.4. Entretanto, neste caso o sincronismo do esquema é dado pela variável *SlotTime* ao invés da variável *FramePeriod.*

6.13.1.4.5 - Parâmetros

O modelo da camada ATM do chaveador possui apenas um parâmetro: tempo de ciclo (*SlotTime*). Este parâmetro estabelece o intervalo de tempo necessário para retirar as células dos servidores dos sistemas de fila associados e encaminhar para a camada física correspondente. O valor *default* do parâmetro *SlotTime* é 2,7261 μs, o que representa uma taxa interna de transmissão de 155.52 *Mbps* por porta. Se o chaveador possui oito portas por exemplo, a taxa interna total de transmissão é de 1,244 *Gbps*.

6.13.1.4.6 - Dados de Tabela

O modelo da camada ATM do chaveador não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.1.4.7 - Funcionamento

A Figura 6.34 ilustra o funcionamento do modelo da camada ATM do chaveador. As células chegam a camada ATM através de eventos enviados a partir das instâncias de camada física do chaveador. Estes eventos contém a primitiva PHY-DATA *Indication.* O procedimento ATM *Receiver* retira a célula ATM desta primitiva e verifica se esta célula é uma célula de OAM, de gerenciamento de recursos ou de metasinalização. Se a célula recebida for de qualquer um destes tipos, ela será encaminhada para um procedimento específico. Na versão atual do modelo desta camada, estes procedimentos apenas retiram de circulação tais células. O objetivo é permitir que versões futuras do modelo da camada ATM do chaveador implementem toda a funcionalidade destes procedimentos. No procedimento *Cell Switching* é feita a translação dos identificadores virtuais VPI e VCI presentes no cabeçalho da célula ATM. A translação dos campos VPI e VCI da célula é feita a partir de

consultas à tabela de dados do chaveador. Primeiramente, é feita uma consulta para determinar o identificador de conexão de rede ATM (NCI) a partir dos campos VPI e VCI atuais da célula ATM. Em seguida é feita outra consulta para determinar os novos campos VPI e VCI da célula, a partir do NCI conhecido. Por último, é feita uma consulta para determinar o nome da porta de saída para qual a célula deve ser enviada. Esta consulta também é feita a partir do NCI. Então o procedimento ATM *Sender Part I* armazena a célula ATM no sistema de fila desta porta de saída e agenda eventos para acionar o procedimento ATM *Sender Part II*. Este procedimento encaminha a célula ATM ao serviço e posteriormente retira a célula do sistema de fila. Após a célula ser retirada do sistema de fila associado ela é acomodada no campo *Interface Data* de uma primitiva PHY-DATA *Request*. Esta primitiva é então enviada para a respectiva porta de saída do chaveador através de um evento.

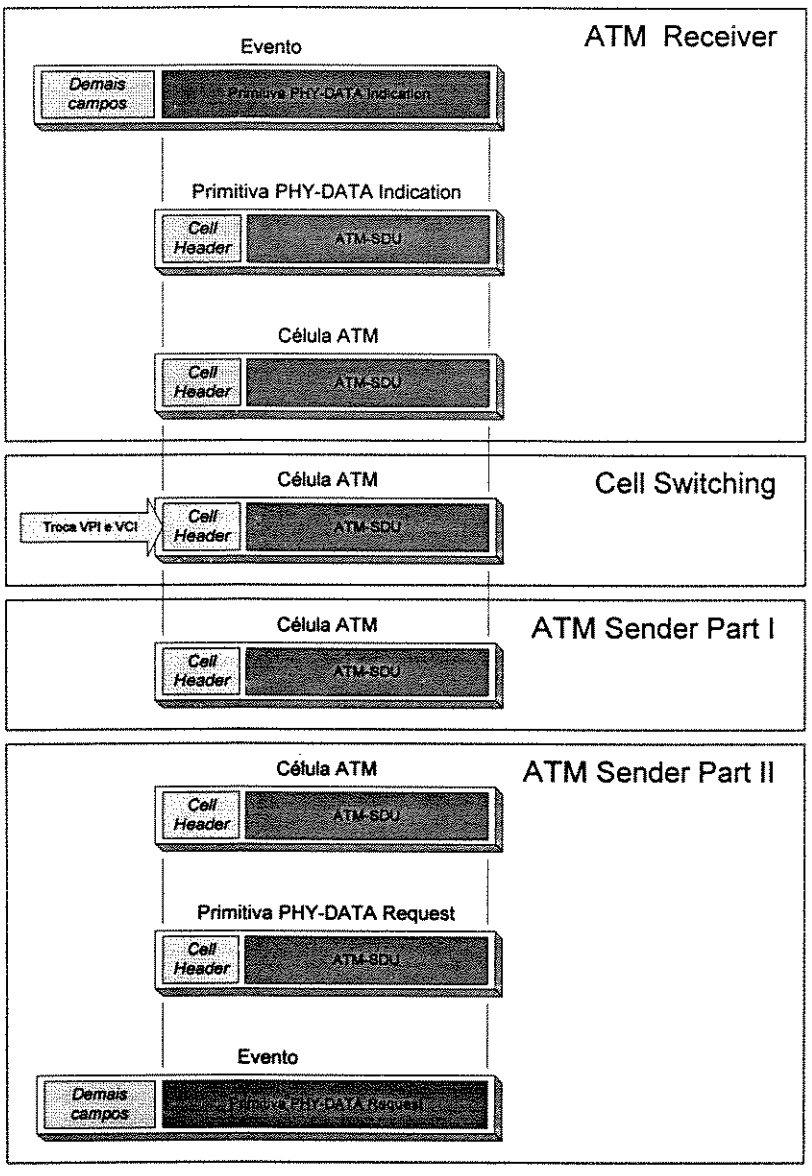


Figura 6.34 – Funcionamento do modelo da camada ATM do chaveador

6.13.1.4.8 - Processos

O modelo da camada ATM do chaveador possui dois processos: PHY_DATA_INDICATION e ATM_SEND_CELL. Estes processos são implementados no módulo de execução do modelo e são responsáveis pela execução dos procedimentos ATM

Sender Part I, ATM *Sender Part II*, *Cell Switching* e ATM *Receiver*. A Figura 6.35 mostra um fluxograma dos processos do modelo da camada ATM do chaveador.

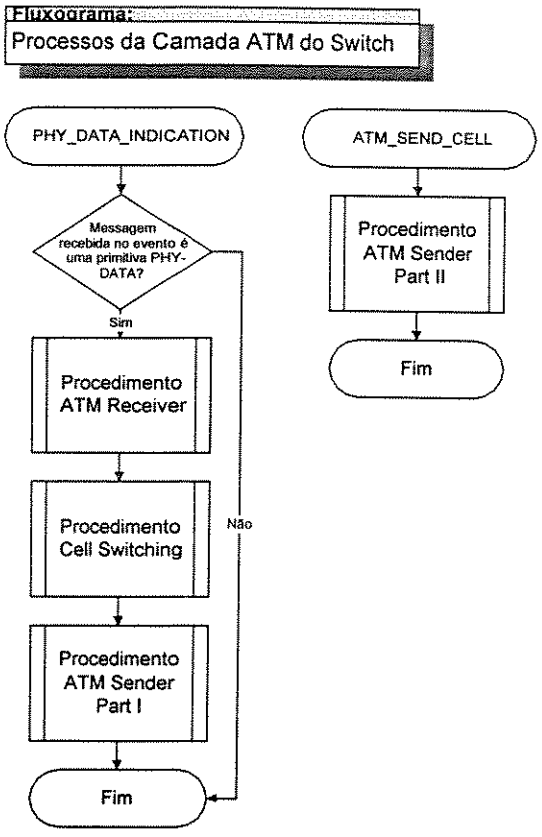


Figura 6.35 – Fluxograma dos processos da camada ATM do chaveador

6.13.2 - Modelos de Camadas para o Aplicativo ATM

6.13.2.1 - Fonte de Tráfego CBR

Como já abordamos anteriormente, no **SimATM** cada fonte de tráfego é implementada como um modelo de camada para o aplicativo ATM do simulador. Estas fontes de tráfego modelam as camadas superiores do B-ISDN PRM.

6.13.2.1.1 - Descrição do Modelo

O modelo da fonte de tráfego CBR envia TS-PDUs (*Traffic Source PDUs* ou CPCS-SDUs) de tamanho fixo, a uma taxa de *bits* constante, para a camada de adaptação ATM de um equipamento da rede. A fonte de tráfego CBR permanece transmitindo até que um número total de *bytes* seja atingido. É possível ainda especificar o instante de inicio de transmissão e o tipo de AAL a ser utilizada no equipamento ATM.

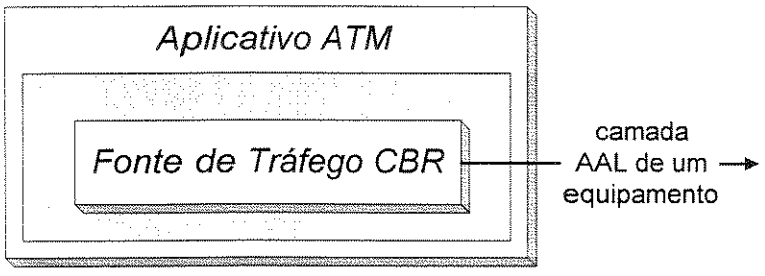


Figura 6.36 – Contexto do modelo da fonte de tráfego CBR em um aplicativo ATM

6.13.2.1.2 - Modelo Funcional

A Figura 6.37 mostra o modelo funcional da camada fonte de tráfego CBR do aplicativo ATM implementado no simulador.

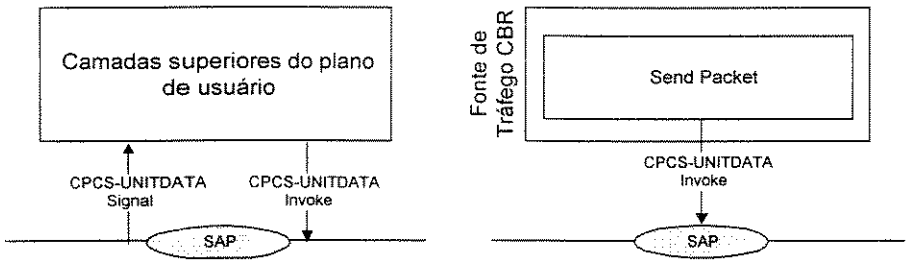


Figura 6.37 – Modelo funcional da camada fonte de tráfego CBR do aplicativo ATM

As funções executadas pela camada encontram-se no procedimento *Send Packet*. Este procedimento envia TS-PDUs para a camada de adaptação ATM de um equipamento ATM.

6.13.2.1.3 - Parâmetros

O modelo da camada fonte de tráfego CBR possui apenas oito parâmetros: taxa de transmissão em *bits*/segundo (*BitRate*), tamanho das TS-PDUs (*TSPDULength*), número de *bytes* a serem transmitidos (*NumberOfBytesToBeSent*), tempo de inicio de transmissão (*StartTime*), nome das TS-PDUs (*TSPDUName*), tipo de AAL a ser utilizada (AAL), categoria de serviço a ser utilizada (*ServiceCategorie*) e amostragem de tempo de criação de TS-PDUs (*LogTSPDUTimeStatus*). O parâmetro *LogTSPDUTimeStatus* é um parâmetro de simulação. Os demais parâmetros são parâmetros de *design*. A partir dos parâmetros *BitRate* e *TSPDULength* é definido um intervalo de tempo entre a transmissão de cada TS-PDU. O parâmetro AAL especifica o tipo de AAL que deve existir no equipamento a que o aplicativo está conectado. O parâmetro *ServiceCategorie* especifica a categoria de serviço que deve ser utilizada para transmitir as TS-PDUs da fonte CBR. O parâmetro *LogTSPDUTimeStatus* habilita ou desabilita a amostragem para arquivo do instante de tempo de transmissão de TS-PDUs. Os valores *default* dos parâmetros são: *BitRate* = 1 *Mbps*, *TSPDULength* = 40 *bytes*,

NumberOfBytesToBeSent = 1 *Megabytes*, *StartTime* = 0, *TSPDUName* = "CBR *Constant Length Packet*", AAL = "AAL5", *ServiceCategorie* = "CBR" e *LogTSPDUTimeStatus* = "OFF".

6.13.2.1.4 - Dados de Tabela

O modelo da camada fonte de tráfego CBR não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.2.1.5 - Funcionamento

O modelo da camada fonte de tráfego CBR gera TS-PDUs (CPCS-SDUs) que são alojadas em primitivas CPCS-UNITDATA *Invoke*. Estas primitivas são enviadas através de eventos para a camada AAL do equipamento ATM imediatamente conectado ao aplicativo a qual a fonte CBR pertence. Estes eventos possuem um identificador de conexão de dados (DCI – *Data Connection Identifier*), que é utilizado para identificar a qual conexão de dados pertence a TS-PDU que está sendo transmitida. Como veremos no item 6.20 - Conexões de Dados, o modelo de aplicativo ATM pode estabelecer este tipo de conexões com mais de um aplicativo ATM de destino. O modelo da camada fonte de tráfego CBR envia eventos contendo a primitiva CPCS-UNITDATA *Invoke* para cada conexão de dados estabelecida a partir do aplicativo fonte. As TS-PDUs são enviadas até que o número de *bytes* a serem transmitidos seja atingido.

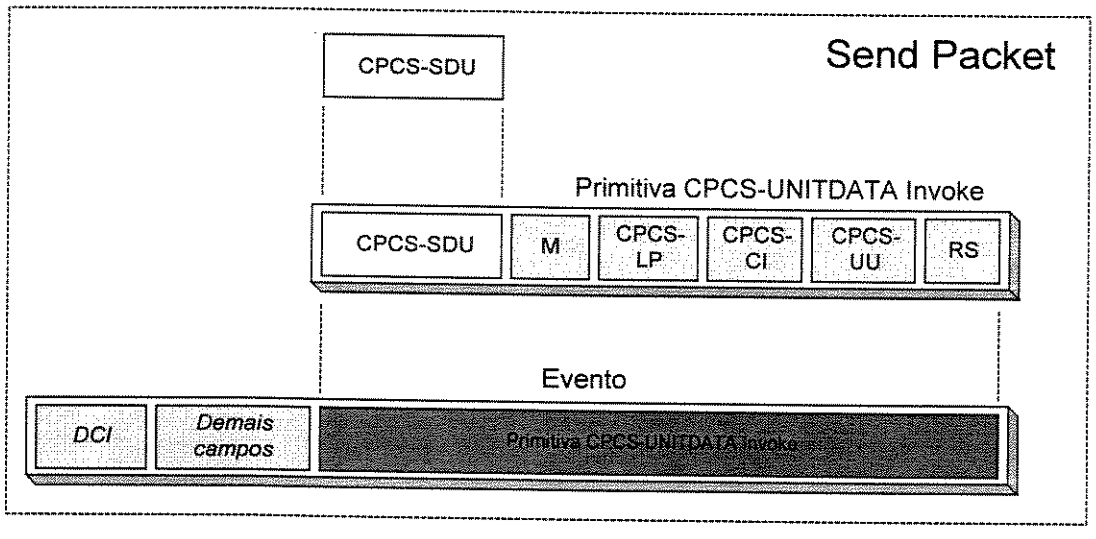


Figura 6.38 – Funcionamento da camada fonte de tráfego CBR

6.13.2.1.6 - Processos

O modelo da camada fonte de tráfego CBR possui apenas um processo: SEND_PACKET. Este processo é responsável pela execução do procedimento *Send Packet*. A Figura 6.39 mostra um fluxograma do processo do modelo da camada fonte de tráfego CBR.

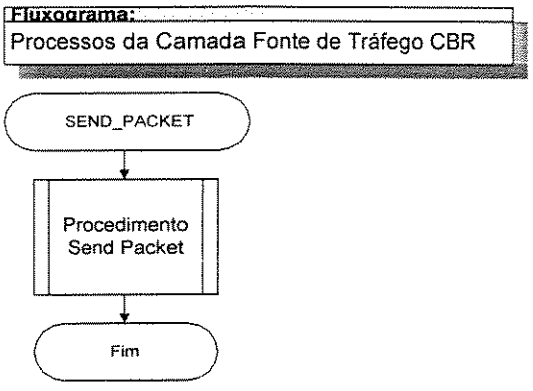


Figura 6.39 – Fluxograma dos processos da camada fonte de tráfego CBR

6.13.2.2 - Fonte de Tráfego VBR Surtuoso

O **SimATM** conta com um modelo de fonte de tráfego de taxa de bits variável (VBR – *Variable Bit Rate*) surtuosa. Esta fonte de tráfego também foi implementada como um modelo de camada para o aplicativo ATM do simulador.

6.13.2.2.1 - Descrição do Modelo

O modelo da fonte de tráfego VBR surtuoso (*batch*) envia TS-PDUs (*Traffic Source PDUs* ou CPCS-SDUs) de tamanho médio igual a média de uma distribuição de *poisson*, a cada intervalo médio de tempo dado pela média de uma distribuição exponencial negativa. É possível definir o número de *bytes* totais a serem transmitidos, o instante de tempo de inicio de transmissão, o tipo de AAL a ser utilizado no equipamento ATM e a categoria de serviço da conexão ATM a ser utilizada.

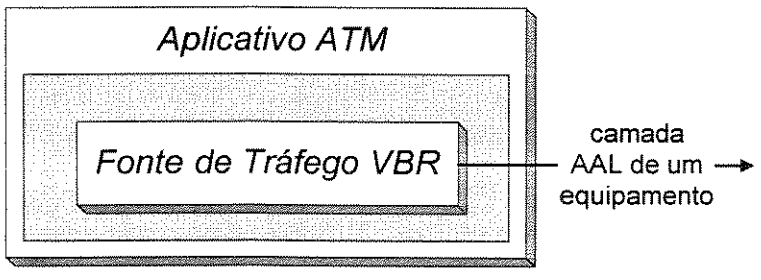


Figura 6.40 – Contexto do modelo da fonte de tráfego VBR *Batch* em um aplicativo ATM

6.13.2.2.2 - Modelo Funcional

A Figura 6.41 mostra o modelo funcional da camada fonte de tráfego VBR surtuoso do aplicativo ATM implementado no simulador.

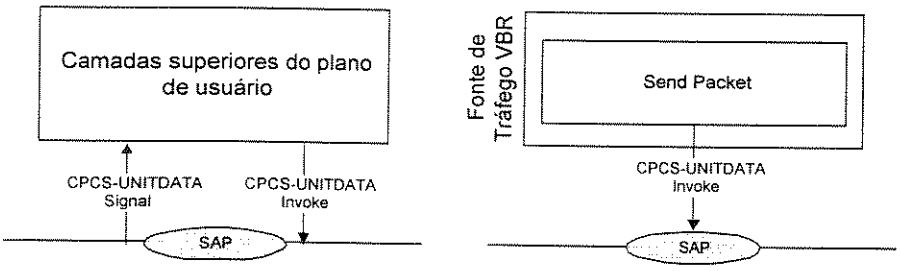


Figura 6.41 – Modelo funcional da fonte de tráfego VBR do aplicativo ATM

As funções executadas pela camada encontram-se no procedimento *Send Packet*. Este procedimento envia TS-PDUs para a camada de adaptação ATM de um equipamento ATM.

6.13.2.2.3 - Parâmetros

O modelo da camada fonte de tráfego VBR surtuoso possui nove parâmetros: número de *bytes* a serem transmitidos (*NumberOfBytesToBeSent*), tamanho médio da TS-PDU em células ATM (*MeanNumberOfATMCells*), intervalo médio de tempo entre surtos (*MeanIntervalBetweenBatches*), tempo de inicio de transmissão (*StartTime*), nome das TS-PDUs (*TSPDUName*), tipo de AAL a ser utilizada (AAL), categoria de serviço a ser utilizada (*ServiceCategorie*), taxa de transmissão em *bits*/segundo (*BitRate*) e estado da amostragem do instante de tempo de transmissão de TS-PDUs (*LogTSPDUTimeStatus*). Os valores *default* dos parâmetros são: *NumberOfBytesToBeSent* = 1 *Megabyte*, *MeanNumberOfATMCells* = ½ célula, *MeanIntervalBetweenBatches* = 2.7263 μseg., *StartTime* = 0 seg., *TSPDUName* = "VBR *Batch*", AAL = "AAL5", *ServiceCategorie* = "VBR", *BitRate* = 20 *Mbps*, *LogTSPDUTimeStatus* = "OFF".

O parâmetro *BitRate* deve ser configurado com o valor aproximado da taxa resultante da transmissão de TS-PDUs de tamanho médio igual a *MeanNumberOfATMCells* e com intervalo médio de transmissão *MeanIntervalBetweenBatches*.

6.13.2.2.4 - Dados de Tabela

O modelo da camada fonte de tráfego VBR surtuoso não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.2.2.5 - Funcionamento

O modelo da camada fonte de tráfego VBR surtuoso funciona da mesma forma que o modelo da camada fonte de tráfego CBR. As TS-PDUs (CPCS-SDUs) também são alojadas

em primitivas CPCS-UNITDATA *Invoke* até que o número de *bytes* a serem transmitidos seja atingido. O identificador de conexão de dados (DCI) também é inserido no evento.

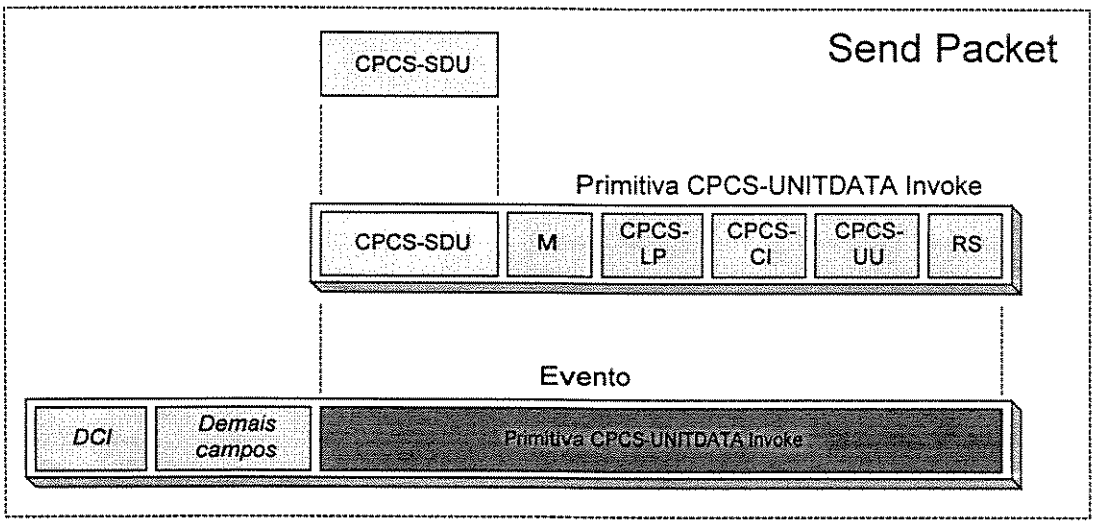


Figura 6.42 – Funcionamento da camada fonte de tráfego VBR *Batch*

6.13.2.2.6 - Processos

O modelo da camada fonte de tráfego VBR surtuoso possui apenas um processo: SEND_PACKET. Este processo é implementado no módulo de execução do modelo e é responsável pela execução do procedimento *Send Packet*. A mostra um fluxograma do processo do modelo da camada fonte de tráfego VBR.

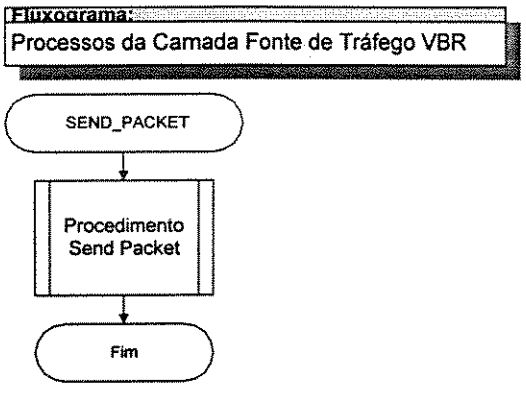


Figura 6.43 - Fluxograma dos processos da camada fonte de tráfego VBR

6.13.2.3 - Fonte de Tráfego Genérica

O **SimATM** possui ainda um modelo de fonte de tráfego genérica. Esta fonte é genérica no sentido de que através dela é possível transmitir TS-PDUs (*Traffic Source PDUs* ou CPCS-SDUs) cujo tamanho e instante de tempo de transmissão são lidos de um arquivo. Desta forma é possível simular o efeito de um padrão de tráfego real ou gerado externamente

sobre uma rede ATM. Esta fonte de tráfego também foi implementada como um modelo de camada para o aplicativo ATM do simulador.

6.13.2.3.1 - Descrição do Modelo

O modelo da fonte de tráfego genérica envia TS-PDUs cujo tamanho e instante de transmissão são lidos de um arquivo. É possível definir o número de *bytes* ou de células ATM a serem transmitidas, o tipo de AAL a ser utilizado no equipamento ATM de ingresso na rede e a categoria de serviço da conexão ATM a ser utilizada.

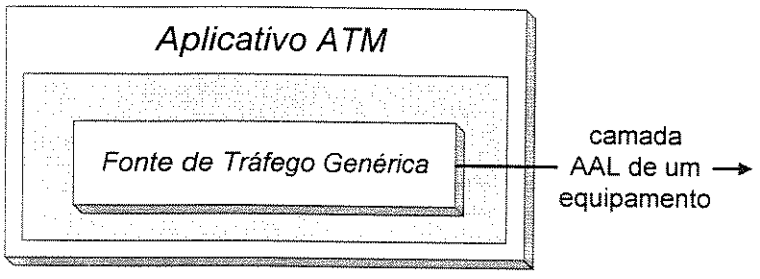


Figura 6.44 – Contexto do modelo da fonte de tráfego genérica em um aplicativo ATM

6.13.2.3.2 - Modelo Funcional

A Figura 6.45 mostra o modelo funcional da camada fonte de tráfego genérica do aplicativo ATM implementado no simulador.

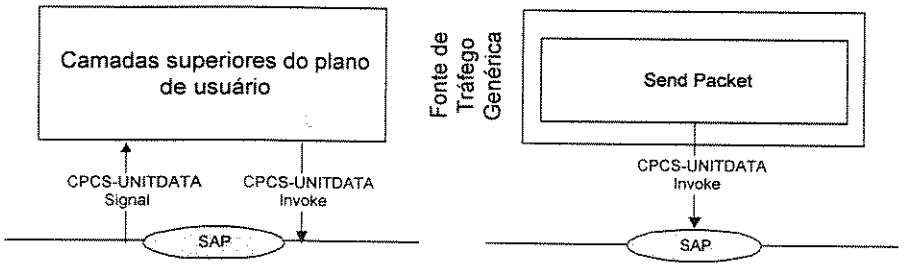


Figura 6.45 – Modelo funcional da fonte de tráfego genérica do aplicativo ATM

As funções executadas pela camada encontram-se no procedimento *Send Packet*. Este procedimento envia TS-PDUs para a camada de adaptação ATM de um equipamento ATM.

6.13.2.3.3 - Parâmetros

O modelo da camada fonte de tráfego genérica possui oito parâmetros: número de *bytes* a serem transmitidos (*NumberOfBytesToBeSent*), número de células a serem transmitidas (*NumberOfCellsToBeSent*), nome das TS-PDUs (*TSPDUName*), tipo de AAL a ser utilizada (AAL), categoria de serviço a ser utilizada (*ServiceCategorie*), taxa de transmissão em *bits*/segundo (*BitRate*), nome do arquivo fonte de tráfego (*SourceFileName*) e configuração do instante de parada de transmissão (*StopToSendAt*). Os valores *default* são:

NumberOfBytesToBeSent = 1 *Megabyte*, *NumberOfCellsToBeSent* = 1000000 de células, *TSPDUName* = "*Packet read from file*", AAL = "AAL5", *ServiceCategorie* = "VBR", *BitRate* = 10 Mbps, *SourceFileName* = "", *StopToSendAt* = "END".

O parâmetro *StopToSendAt* pode assumir três valores: "END", "MAX_BYTES" e "MAX_CELLS". Se o valor for "END", a transmissão será terminada no final do arquivo fonte de tráfego. Se o valor for "MAX_BYTES", a transmissão será terminada quando o número de *bytes* transmitidos for maior ou igual ao valor do parâmetro *NumberOfBytesToBeSent*. Finalmente, se o valor "MAX_CELLS" a transmissão será terminada quando o número de células transmitidas for maior ou igual ao valor do parâmetro *NumberOfCellsToBeSent*.

O parâmetro *BitRate* deve ser configurado com o valor aproximado da taxa resultante da transmissão das TS-PDUs lidas a partir do arquivo.

6.13.2.3.4 - Dados de Tabela

O modelo da camada fonte de tráfego VBR surtuoso não armazena nenhum dado na sua tabela, entretanto realiza a consulta de dados na tabela do bloco a que pertence.

6.13.2.3.5 - Funcionamento

O modelo da camada fonte genérica funciona da mesma forma que os modelos das camadas CBR e VBR. As TS-PDUs (CPCS-SDUs) também são alojadas em primitivas CPCS-UNITDATA *Invoke*. O identificador de conexão de dados (DCI) também é inserido no evento.

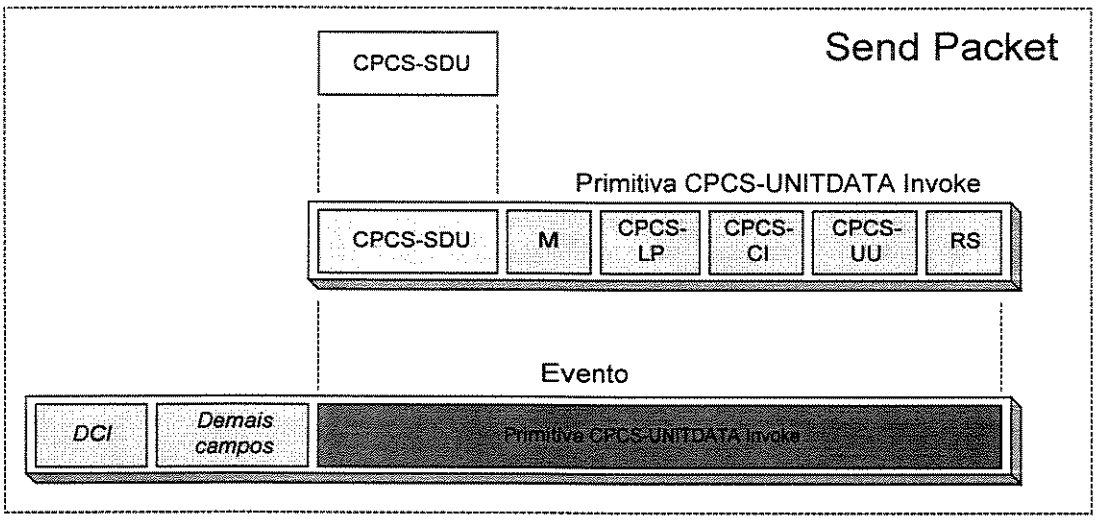


Figura 6.46 – Funcionamento da camada fonte de tráfego VBR *Batch*

6.13.2.3.6 - Processos

O modelo da camada fonte de tráfego genérica também possui apenas um processo: SEND_PACKET. Este processo é implementado no módulo de execução do modelo e é responsável pela execução do procedimento *Send Packet*. A Figura 6.47 mostra um fluxograma do processo do modelo da camada fonte de tráfego genérica.

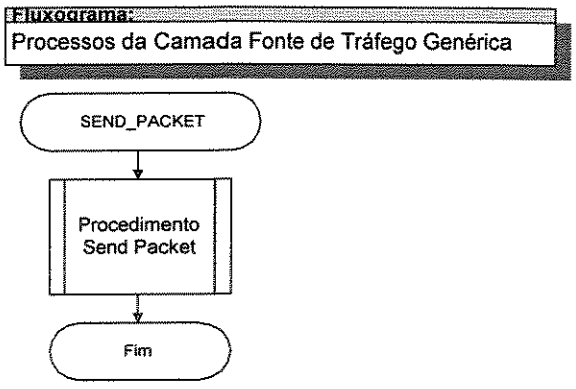


Figura 6.47 – Fluxograma dos processos da camada fonte de tráfego genérica

6.13.2.4 - Receptor de Tráfego

O **SimATM** possui um modelo de receptor de tráfego. Este receptor também foi implementado como um modelo de camada para o aplicativo ATM do simulador.

6.13.2.4.1 - Descrição do Modelo

O modelo receptor de tráfego recebe as TS-PDUs enviadas por modelos de fontes de tráfego e coleta estatísticas de tempo de permanência destas TS-PDUs na rede.

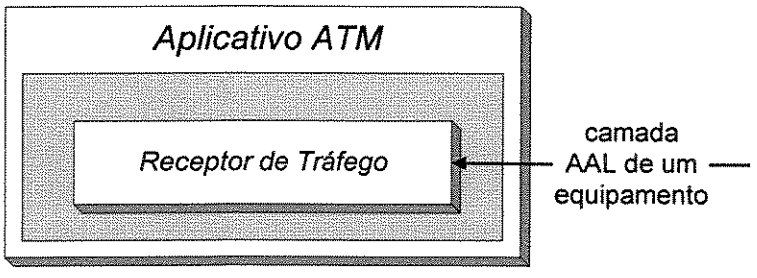


Figura 6.48 – Contexto do modelo de receptor de tráfego em um aplicativo ATM

6.13.2.4.2 - Modelo Funcional

A Figura 6.49 mostra o modelo funcional da camada receptora de tráfego do aplicativo ATM implementado no simulador.

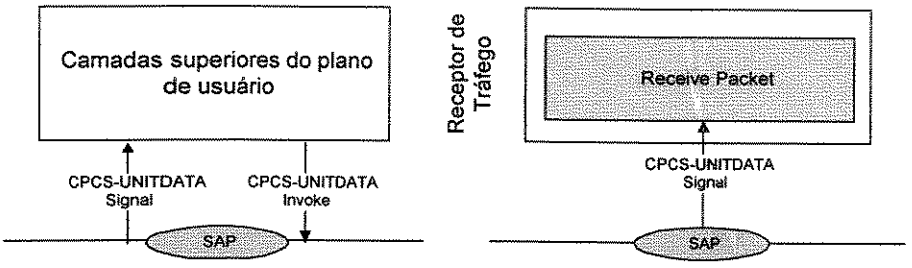


Figura 6.49 – Modelo funcional da camada receptora de tráfego do aplicativo ATM

As funções executadas pela camada encontram-se no procedimento *Receive Packet*. Este procedimento recebe TS-PDUs da camada de adaptação ATM de um equipamento ATM.

6.13.2.4.3 - Parâmetros

O modelo da camada receptora de tráfego possui três parâmetros: estado da amostragem de variáveis por célula (*TriggerByCellStatus*), estado do cálculo e amostragem de variáveis estatísticas (*SampleStatisticsStatus*) e intervalo de tempo entre amostras de variáveis estatísticas (*LoggingEveryXSecs*). Estes parâmetros são parâmetros de simulação. O parâmetro *TriggerByCellStatus* permite a amostragem para arquivo do tempo de permanência total de uma TS-PDU na rede ATM. Esta amostragem pode ser feita toda vez que uma TS-PDU chega a camada receptora de tráfego do aplicativo ATM de destino. O valor *default* deste parâmetro é "OFF" (desligado). O parâmetro *SampleStatisticsStatus* habilita ou não o cálculo da média e da variância do tempo de permanência de TS-PDUs na rede. Toda vez que uma TS-PDU chega à camada receptora de tráfego de um aplicativo ATM de destino, as variáveis estatísticas são atualizadas se o parâmetro *TriggerByCellStatus* estiver habilitado. O valor *default* deste parâmetro também é "OFF" (desligado). O parâmetro *LoggingEveryXSecs* configura o intervalo de tempo entre amostras de variáveis estatísticas para arquivo. O valor *default* é 1ms. Este parâmetro só é utilizado quando o parâmetro *SampleStatisticsStatus* = "ON" (ligado).

6.13.2.4.4 - Arquivos de Saída

O modelo de camada do receptor de tráfego, de acordo com o estado dos seus parâmetros, pode possuir até dois arquivos de saída: arquivo de amostragem de variáveis por chegada de TS-PDU e arquivo de amostragem de variáveis estatísticas. Estes arquivos possuem a extensão *.dat*. No primeiro arquivo a variável amostrada é o tempo de permanência de TS-PDUs na rede (regime transitório), enquanto no segundo arquivo as variáveis amostradas consistem da média e da variância estimada do tempo de permanência de TS-PDUs na rede (regime permanente).

6.13.2.4.5 - Dados de Tabela

O modelo da camada fonte de tráfego VBR surtuoso não armazena nenhum dado na sua tabela.

6.13.2.4.6 - Funcionamento

O modelo da camada receptora de tráfego recebe da AAL de um equipamento ATM eventos que contém a primitiva CPCS-UNITDATA *Signal* e o identificador de conexão de dados DCI. Através de deste identificador é possível determinar a que conexão de dados pertence uma TS-PDU recebida. A primitiva CPCS-UNITDATA *Signal* contém a TS-PDU (CPCS-SDU) original que foi enviada pela rede ATM. A versão atual desta camada não utiliza os campos da primitiva para verificar as condições atuais da CPCS-SDU recebida. Futuras versões do modelo receptor de tráfego poderão fazer melhor uso destas informações.

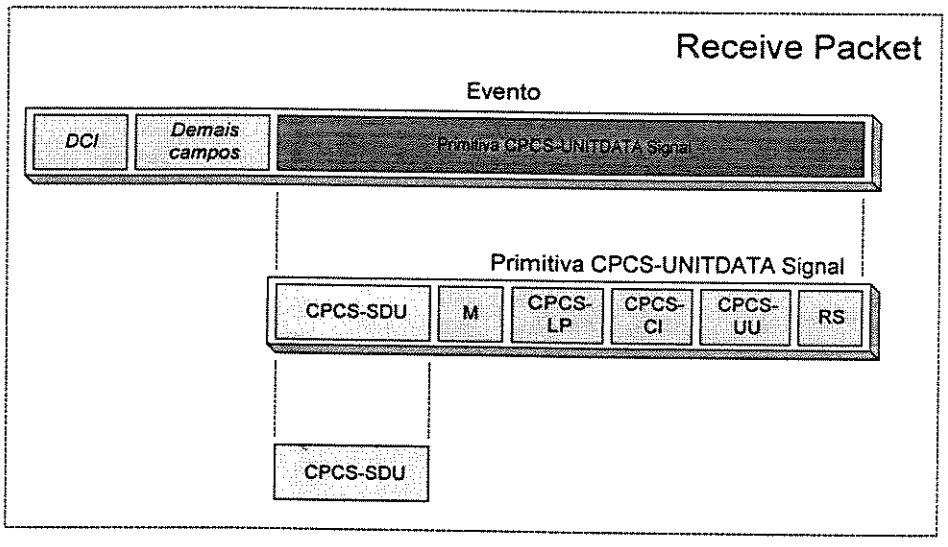


Figura 6.50 – Funcionamento da camada receptora de tráfego

6.13.2.4.7 - Processos

O modelo da camada receptora de tráfego possui apenas um processo: RECEIVE_PACKET. Este processo é implementado no módulo de execução do modelo e é responsável pela execução do procedimento *Receive Packet*. A Figura 6.51 mostra um fluxograma do processo do modelo da camada receptora de tráfego.

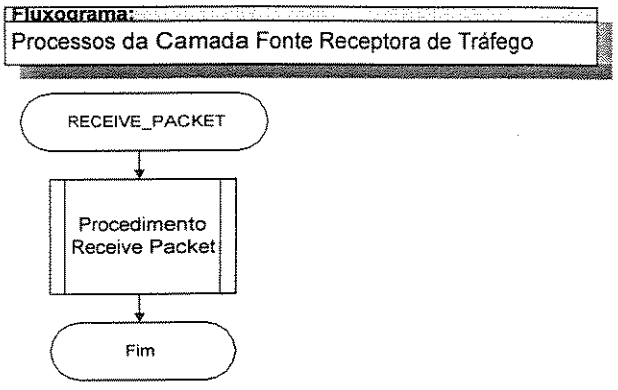


Figura 6.51 – Fluxograma dos processos da camada receptora de tráfego

6.14 - Blocos

O bloco é um elemento da estrutura do **SimATM** que constitui a estrutura básica dos equipamentos e aplicativos da rede ATM. Cada bloco possui um nome, um tipo, um ou mais parâmetros, uma tabela de dados, um módulo gerenciador de parâmetros, um módulo gerenciador de tabelas de dados, um módulo gerenciador de camadas, um módulo gerenciador de sistemas de fila, um módulo de inicialização, um módulo de conexão, um módulo de execução, um módulo de desconexão e um módulo de terminação. A Figura 6.52 ilustra a estrutura **bloco** do simulador.

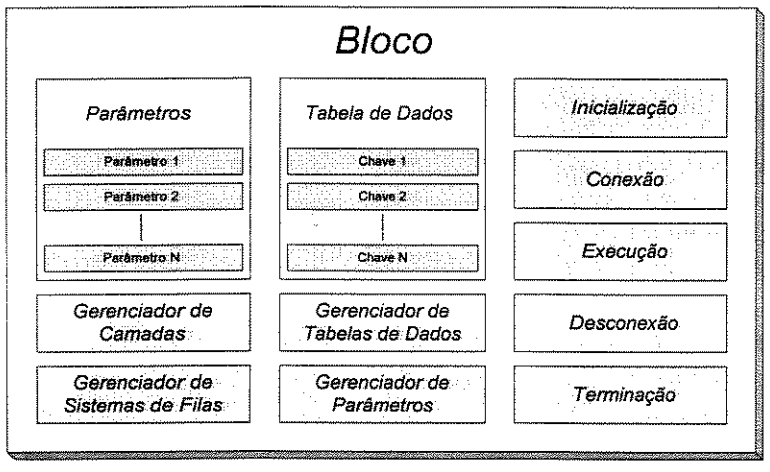


Figura 6.52 – Estrutura dos blocos do SimATM

O módulo gerenciador de parâmetros permite consultar, modificar e remover os parâmetros das camadas e sistemas de fila de um bloco, bem como da rede ATM a que o bloco pertence. O módulo gerenciador de tabelas de dados permite consultar, modificar e remover os dados das tabelas das camadas de um bloco. O módulo gerenciador de tabelas permite ainda realizar procuras nessas tabelas. O módulo gerenciador de camadas permite acrescentar e remover camadas em um bloco, bem como acionar os módulos de inicialização, execução e terminação das camadas de um bloco. O módulo gerenciador de camadas permite

ainda verificar a existência de uma dada camada e consultar o nome de todas as camadas de um bloco. O módulo gerenciador de sistemas de fila permite acrescentar e remover sistemas de fila em um bloco, bem como acionar os módulos de inicialização, execução e terminação dos sistemas de fila de um bloco. O módulo gerenciador de sistemas de fila permite ainda verificar a existência de um dado sistema de fila, consultar o nome de todos os sistema de um bloco, armazenar e remover células em um sistema de fila e consultar o número de células total, no *buffer* e no servidor de um sistema de fila. Os módulos de inicialização, conexão, execução, desconexão e terminação são definidos nos modelos que herdam a estrutura básica **bloco**.

6.15 - Equipamentos ATM

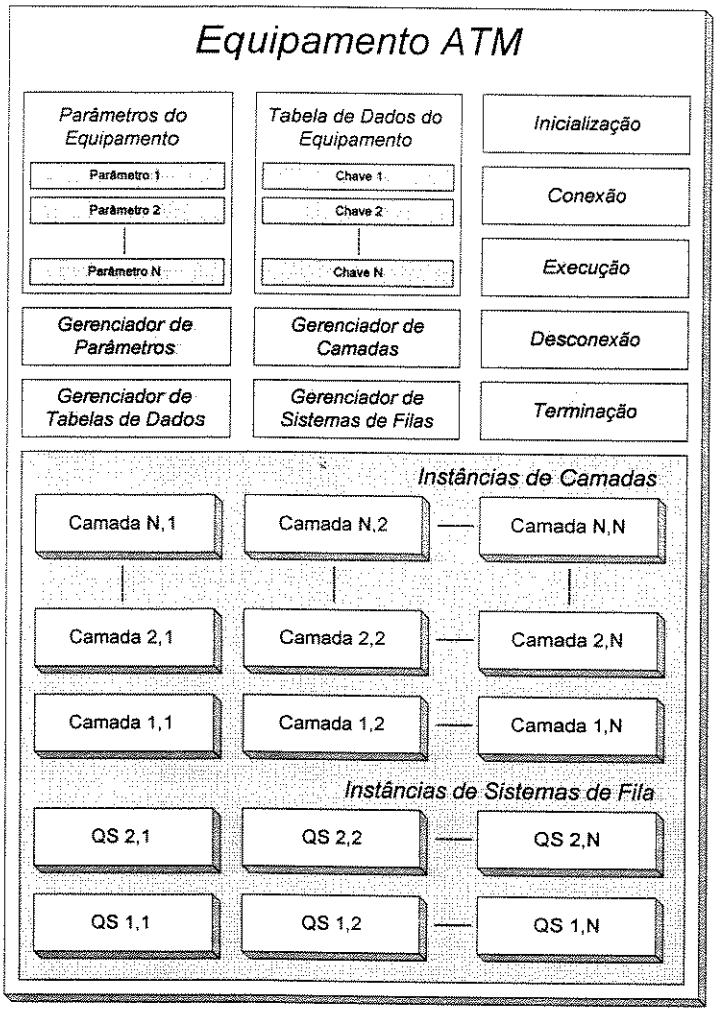


Figura 6.53 – Estrutura dos equipamentos ATM

O equipamento ATM é um elemento da estrutura do **SimATM** que constitui a estrutura básica dos modelos de *hardware* da rede ATM. É implementado a partir da estrutura **bloco**, e portanto herda toda a sua funcionalidade. Os equipamentos ATM são compostos por várias instâncias de camadas e sistemas de fila. Sua estrutura permite facilmente acrescentar, substituir e remover essas camadas e sistemas. A Figura 6.53 mostra a estrutura dos equipamentos ATM.

6.16 - Modelos de Equipamentos ATM

Os modelos de *hardware* implementados no **SimATM** são: terminal faixa larga ATM (ATM B-TE – ATM *Broadband Terminal Equipment*) e chaveador ATM (ATM *switch*). Esses modelos são implementados a partir da estrutura básica **equipamento ATM**. A seguir detalharemos estes modelos.

6.16.1 - Terminal Faixa Larga ATM

O modelo terminal faixa larga ATM simula um equipamento terminal da B-ISDN. Este modelo se conecta a um ou mais aplicativos ATM, e a um outro equipamento ATM que pode ser um BTE ou um chaveador. A Figura 6.54 ilustra o contexto do modelo BTE em uma rede ATM.

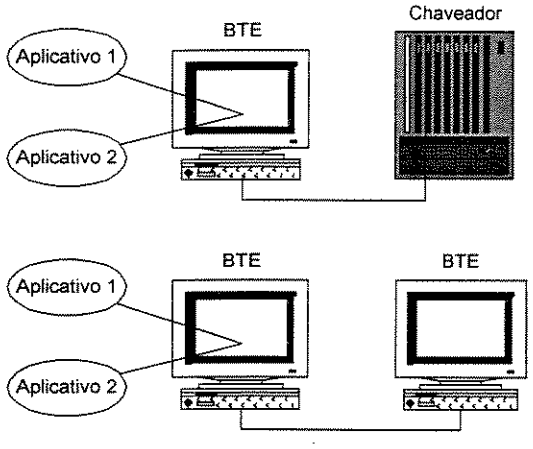


Figura 6.54 – Contexto do modelo BTE em uma rede ATM

6.16.1.1 - Camadas e Sistemas de Filas

O modelo do BTE possui uma instância das camadas: AAL 5, camada ATM do BTE e camada física para a interface baseada em células. O modelo do BTE possui ainda uma instância de modelo de sistema de fila, que é associado a instância da camada física. A Figura 6.55 ilustra a estrutura do BTE.

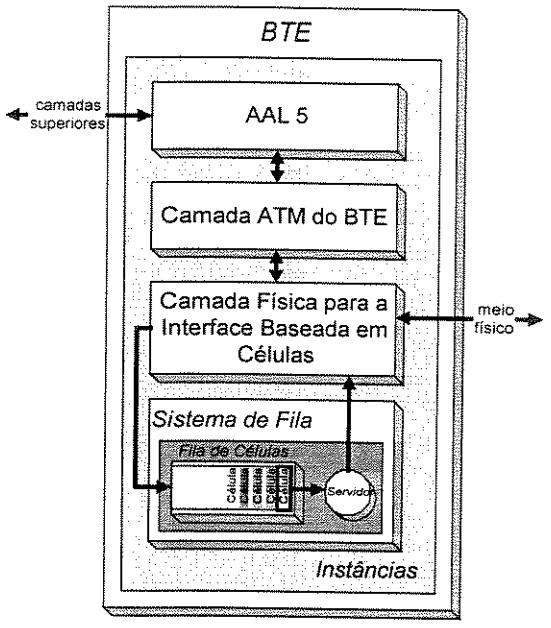


Figura 6.55 – Camadas e sistema de fila do modelo do BTE

As PDUs provenientes das camadas superiores são fragmentados na AAL 5 e passados para a camada ATM, onde são encapsulados em células e enviados para a camada física. Na camada física as células são armazenadas no *buffer* do sistema de fila associado e após serem servidas, são enviadas pelo meio físico até o próximo equipamento da rede. As camadas superiores se encontram no aplicativo ATM.

6.16.1.2 - Parâmetros

O modelo BTE possui o parâmetro: número de aplicativos (*NumberOfApplications*), que define a quantia máxima de aplicativos ATM que podem ser conectados ao BTE.

6.16.1.3 - Dados de Tabela

O modelo BTE possui uma tabela de dados que armazena informações relacionadas com as conexões ATM e de dados que passam através dele. O VPI e o VCI de ingresso para as células ATM de cada conexão ATM são armazenados, bem como o categoria de serviço de cada conexão ATM. São ainda armazenadas informações relacionadas com o encaminhamento de eventos externos e internos ao BTE.

6.16.1.4 - Funcionamento

O funcionamento do modelo BTE será descrito a partir das funções desempenhadas por cada um dos módulos da sua estrutura.

6.16.1.4.1 - Módulo de Inicialização

Este módulo é acionado pela **rede ATM** logo após a criação do BTE. Quando este módulo é acionado uma instância da camada ATM e da AAL 5 são criadas. A camada ATM é chamada de "ATM" e a camada AAL 5 é chamada de "AAL5".

6.16.1.4.2 - Módulo de Conexão

Este módulo é acionado quando o BTE está sendo conectado a um aplicativo ATM ou a um outro equipamento ATM. O objetivo é determinar se o BTE pode ou não ser conectado. Nesta situação o módulo de conexão é informado a respeito do nome, do tipo e do número de vizinhos do mesmo tipo do bloco a que o BTE será conectado. O tipo de interface a ser utilizada entre os dois blocos também é especificada. No caso da conexão entre o BTE e um outro equipamentos ATM, a interface especifica o modelo da camada física a ser utilizada na conexão. No caso da conexão entre o BTE e um aplicativo ATM, a interface não é utilizada.

Se o BTE estiver sendo conectado a um aplicativo ATM, é verificado se o número de aplicativos já conectados é menor do que o número máximo de aplicativos que podem ser conectados ao BTE (especificado pelo parâmetro *NumberOfApplications*). Se essa condição for atendida a conexão entre os dois blocos poderá ser realizada. Entretanto, a conexão só será realizada se houver a aprovação junto ao módulo de conexão do outro bloco a que o BTE está sendo conectado.

Se o BTE estiver sendo conectado a um equipamento ATM, é verificado se o número de equipamentos já conectados ao BTE é igual a zero. Se essa condição for atendida, a conexão entre os dois equipamentos poderá ser realizada. O módulo de conexão cria então uma instância do modelo de sistema de fila, chamado "PHY_QS", e do modelo de camada física para a interface baseada em células, chamado "PHY". São feitos também alguns registros na tabela do BTE. A Tabela 6.2 mostra tais registros.

Chave	Nome	Valor
"PHY"	"PHY_QS"	"PHY_QS"
"DBN"	"PHY"	Nome do bloco a que o BTE se conectou
"DLN"	Nome do bloco a que o BTE se conectou	"PHY"

Tabela 6.2 – Registros feitos na tabela do BTE

O primeiro registro serve para identificar o nome do sistema de fila associado a camada física. O segundo registro contém o nome do bloco de destino (DBN – *Destination Block Name*) após a camada "PHY" no sentido de transmissão. O terceiro registro contém o

nome da camada de destino (DLN – *Destination Layer Name*) para onde devem ser enviados os eventos procedentes do bloco a que o BTE se conectou (eventos externos).

6.16.1.4.3 - Módulo de Execução

Este módulo é responsável pelo encaminhamento dos eventos às camadas e ao sistema de fila do BTE. Esta função é desempenhada pelo procedimento *Run*. A Figura 6.56 mostra um fluxograma deste procedimento.

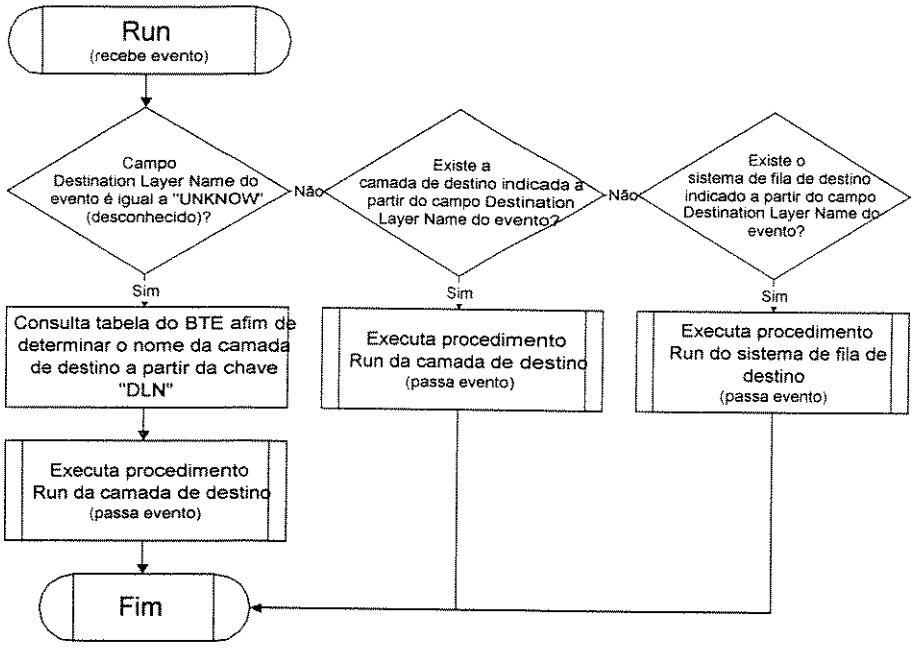


Figura 6.56 – Fluxograma do procedimento *Run*

Quando o *kernel* do simulador possui um evento destinado a um BTE, ele aciona o procedimento *Run* do módulo de execução deste BTE, que irá repassar este evento para uma das suas camadas ou para o seu sistema de fila. Se o campo *Destination Layer Name* do evento for igual a "UNKNOW" (desconhecido), então trata-se de um evento externo, e neste caso o nome da camada de destino é desconhecido. É realizada então uma busca na tabela do BTE, a fim de encontrar o nome desta camada. Esta busca é feita a partir da chave "DLN". Se for encontrado o nome da camada de destino, o evento é repassado para o procedimento *Run* do módulo de execução desta camada. Se o campo *Destination Layer Name* do evento possuir o nome da camada ou sistema de fila de destino, então trata-se de um evento interno. Neste caso, se existir tal camada ou sistema de fila no BTE, o evento será repassado para o respectivo procedimento *Run*.

6.16.1.4.4 - Módulo de Desconexão

Este módulo da estrutura do BTE é acionado quando o BTE está sendo desconectado de outro bloco da rede. Se o BTE estiver sendo desconectado de outro equipamento ATM, então serão removidas as instâncias da camada física e do sistema de fila do bloco. Também serão retirados da tabela do BTE os registros relacionados com esta conexão. Se o BTE estiver sendo desconectado de um aplicativo ATM nenhuma ação precisa ser efetuada.

6.16.1.4.5 - Módulo de Terminação

Quando este módulo é executado são removidas as instâncias das camadas ATM e AAL 5, criadas no módulo de inicialização.

6.16.2 - Chaveador ATM

O modelo chaveador ATM simula um chaveador ATM da B-ISDN. Este modelo se conecta com vários BTEs e com outros chaveadores ATM. A Figura 6.57 ilustra o contexto do modelo chaveador em uma rede ATM.

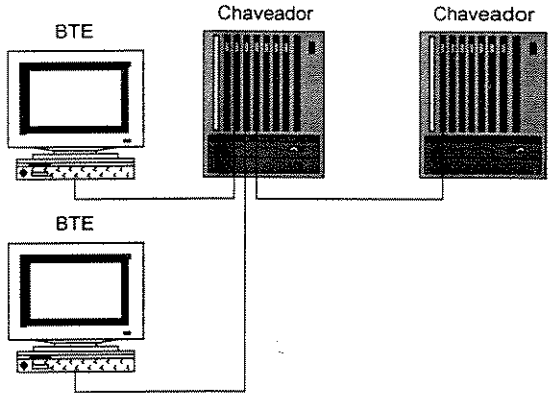


Figura 6.57 – Contexto do modelo chaveador em uma rede ATM

6.16.2.1 - Camadas e Sistemas de Filas

O modelo chaveador ATM possui: uma instância da camada ATM do chaveador, várias instâncias da camada física para a interface baseada em células e várias instâncias de sistemas de fila. A Figura 6.58 ilustra a estrutura do modelo chaveador ATM com duas portas bidirecionais.

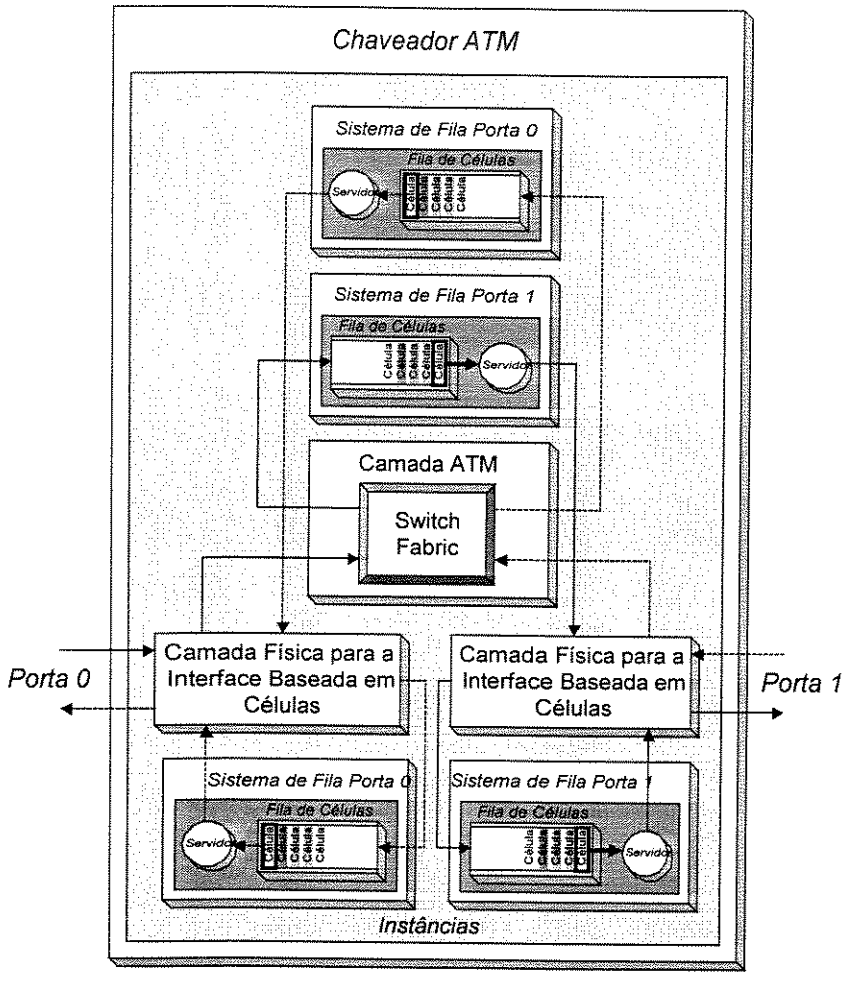


Figura 6.58 – Camadas e sistemas de fila do modelo chaveador ATM

As células que chegam pela porta 0 serão encaminhadas para a porta 1 e vice-versa. Quando uma célula chega em uma das portas ela é encaminhada para a camada ATM do chaveador. Nesta camada os campos VPI e VCI da célula são trocados e a célula é armazenada no sistema de fila associado a camada ATM na direção da porta de saída. Quando a célula é removida deste sistema de fila, ela é encaminhada para a camada física na porta de saída. A célula então é armazenada no sistema de fila desta porta e quando retirada é transmitida para o próximo equipamento ATM.

6.16.2.2 - Parâmetros

O modelo chaveador ATM possui o parâmetro: número de portas (*NumberOfPorts*), que define a quantia máxima de equipamentos ATM que podem ser conectados ao chaveador.

6.16.2.3 - Dados de Tabela

A tabela de dados do modelo chaveador ATM armazena informações de roteamento de células. Os identificadores VPI e VCI de entrada e o nome da porta de entrada, bem como os identificadores VPI e VCI de saída e o nome da porta de saída são armazenados para cada conexão ATM. Também é armazenada na tabela a categoria de serviço de cada conexão ATM.

6.16.2.4 - Funcionamento

O funcionamento do modelo chaveador ATM será descrito a partir das funções desempenhadas por cada um dos módulos da sua estrutura.

6.16.2.4.1 - Módulo de Inicialização

Este módulo é acionado pela **rede ATM** logo após a criação do bloco chaveador ATM. Quando o módulo de inicialização é acionado uma instância do modelo da camada ATM do chaveador é criada.

6.16.2.4.2 - Módulo de Conexão

Este módulo é acionado quando o chaveador está sendo conectado a um outro equipamento ATM. O chaveador não pode ser conectado a aplicativos ATM. O módulo de conexão é informado a respeito do nome e do número de equipamentos ATM vizinhos ao chaveador. O tipo de interface a ser utilizada entre os dois blocos também é informada. Quando acionado, o módulo de conexão verifica se o número de equipamentos ATM conectados ao chaveador é menor que o número máximo de equipamentos que podem ser conectados, que é dado pelo valor do parâmetro *NumberOfPorts*. Se essa condição for atendida, a conexão entre os dois equipamentos poderá ser realizada. Entretanto, a conexão só será realizada se houver a aprovação junto ao módulo de conexão do outro bloco a que o chaveador está sendo conectado. O módulo de conexão cria então uma instância do modelo da camada física para a interface baseada em células, chamada de "PHY_X", onde X é o número da conexão que está sendo realizada. São ainda criadas duas instâncias do modelo de sistema de fila, chamadas de "PHY_QS_X" e "ATM_QS_X". O sistema de fila chamado "ATM_QS_X" é associado a camada ATM do chaveador, enquanto o sistema de fila "PHY_QS_X" é associado a camada física recém criada. São feitos também alguns registros na tabela do chaveador. A Tabela 6.3 mostra tais registros.

Chave	Nome	Valor
"PHY_X"	"PHY_QS"	"PHY_QS_X"
"PHY_X"	"ATM_QS"	"ATM_QS_X"
"DBN"	"PHY_X"	Nome do bloco a que o chaveador se conectou
"DLN"	Nome do bloco a que o chaveador se conectou	"PHY_X"

Tabela 6.3 – Registros feitos na tabela do chaveador

O primeiro registro contém o nome do sistema de fila associado a camada física "PHY_X". O segundo registro contém o nome do sistema de fila associado a camada ATM do chaveador e no sentido de transmissão da camada "PHY_X". O terceiro registro contém o nome do bloco de destino após a camada "PHY_X" no sentido de transmissão. O quarto registro contém o nome da camada de destino ("PHY_X") para onde devem ser enviados os eventos procedentes do bloco a que o chaveador se conectou (eventos externos).

6.16.2.4.3 - Módulo de Execução

O módulo de execução do chaveador funciona da mesma forma que o módulo de execução do BTE, apresentando anteriormente no item 6.16.1.4.3 - Módulo de Execução do BTE.

6.16.2.4.4 - Módulo de Desconexão

Este módulo é acionado quando o chaveador está sendo desconectado de outro equipamento da rede. Quando o módulo de desconexão é acionado, os modelos de camadas e sistemas de fila criados pelo módulo de conexão são removidos. Também são removidos os registros feitos na tabela do chaveador.

6.16.2.4.5 - Módulo de Terminação

Quando este módulo é executado é removida a instância da camada ATM do chaveador, criada no módulo de inicialização.

6.17 - Aplicativos ATM

O aplicativo ATM é um elemento da estrutura do **SimATM** que constitui a estrutura básica dos modelos de *software* da rede ATM. É implementado a partir da estrutura **bloco**, e portanto herda toda a sua funcionalidade. Os aplicativos ATM são compostos por várias instâncias de camadas. Não possuem instâncias de sistemas de fila, uma vez que o modelo atual dos sistemas de fila é voltado especificamente para o armazenamento e serviço de

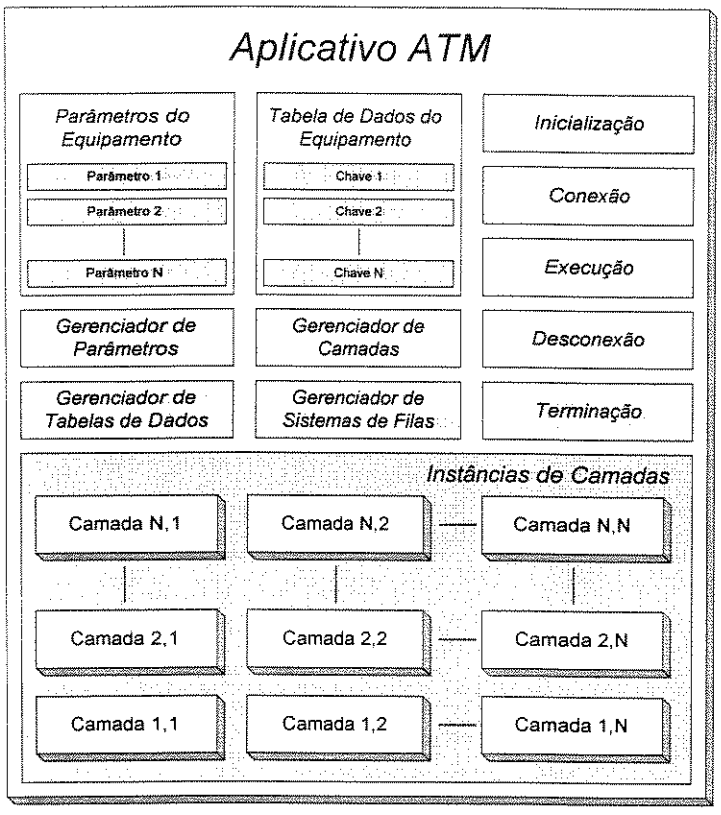


Figura 6.59 – Estrutura dos aplicativos ATM

células ATM. Sua estrutura permite facilmente acrescentar, substituir e remover as suas camadas. A Figura 6.59 mostra a estrutura dos aplicativos ATM.

6.18 - Modelos de Aplicativos ATM

A versão atual do **SimATM** implementa somente um modelo de *software*: aplicativo ATM (ATM *Application*). Esse modelo é construído a partir da estrutura **aplicativo ATM**. A seguir detalharemos o modelo aplicativo ATM.

6.18.1 - Aplicativo ATM

O modelo aplicativo ATM simula a pilha de protocolos da B-ISDN acima da camada de adaptação ATM. Este modelo se conecta a um equipamento terminal faixa larga. É a partir do aplicativo ATM que são estabelecidas as conexões de dados que transmitem informações pela rede ATM. Como veremos no item 6.20 - Conexões de Dados estas conexões são estabelecidas utilizando-se conexões ATM. A Figura 6.60 ilustra o contexto do modelo aplicativo ATM em uma rede.

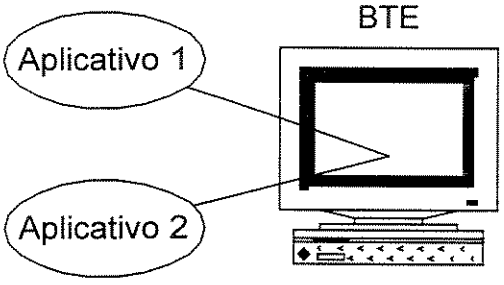


Figura 6.60 – Contexto do modelo aplicativo em uma rede ATM

6.18.1.1 - Camadas

O modelo aplicativo ATM possui uma instância do modelo de uma das camadas geradoras de tráfego: fonte de tráfego CBR, fonte de tráfego VBR surtuoso ou fonte de tráfego genérica. O aplicativo ATM possui também uma instância do modelo da camada receptora de tráfego. A Figura 6.61 mostra as camadas do modelo aplicativo ATM.

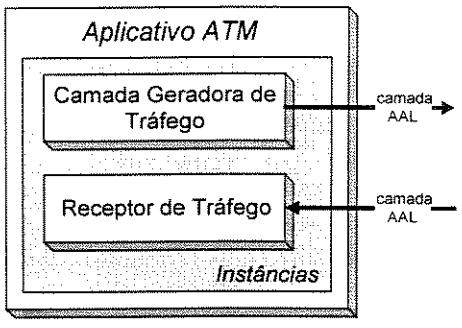


Figura 6.61 – Camadas do modelo aplicativo ATM

TS-PDUs (CPCS-SDUs) são geradas pela camada geradora de tráfego e encaminhadas para a camada AAL do BTE a que o aplicativo está conectado. Na outra extremidade da rede as TS-PDUs são enviadas pela AAL do BTE de egresso para a camada receptora de tráfego do aplicativo, que colhe estatísticas do tempo de permanência das TS-PDUs na rede.

6.18.1.2 - Parâmetros

O modelo aplicativo ATM possui o parâmetro: fonte de tráfego (*TrafficSource*), que define qual modelo de camada geradora de tráfego deve ser utilizado.

6.18.1.3 - Dados de Tabela

O modelo aplicativo ATM possui uma tabela de dados, que armazena a categoria de serviço, a largura de faixa e o nome da conexão ATM a ser utilizada por cada uma das conexões de dados estabelecidas a partir deste aplicativo. Também é armazenada na tabela a

direção de cada conexão de dados, a fim de determinar se uma conexão inicia ou termina em um dado aplicativo.

6.18.1.4 - Funcionamento

O funcionamento do modelo aplicativo ATM será descrito a partir das funções desempenhadas por cada um dos módulos da sua estrutura.

6.18.1.4.1 - Módulo de Inicialização

O modelo aplicativo ATM não utiliza este módulo.

6.18.1.4.2 - Módulo de Conexão

Este módulo é acionado quando o aplicativo ATM está sendo conectado a um BTE. O módulo de conexão é informado a respeito do nome e do número de BTEs a que o aplicativo está conectado. Quando acionado, o módulo de conexão verifica se o aplicativo ATM não está conectado a algum BTE. Se essa condição for atendida, a conexão entre o aplicativo e o BTE poderá ser realizada. Entretanto, a conexão só será realizada se houver a aprovação junto ao módulo de conexão do BTE a que o aplicativo está sendo conectado. O módulo de conexão cria então uma instância da camada geradora de tráfego cujo modelo é definido no parâmetro *TrafficSource*. Esta camada é chamada de "X_TS", onde X é o valor do parâmetro *TrafficSource*. É também criada uma instância do modelo da camada receptora de tráfego, chamada de "TR". É ainda feito um registro na tabela do aplicativo ATM. A Tabela 6.4 mostra este registro.

Chave	Nome	Valor
"DBN"	"X_TS"	Nome do BTE a que o aplicativo se conectou

Tabela 6.4 – Registro feito na tabela do aplicativo

O registro feito contém o nome do bloco de destino (DBN) após a camada "X_TS" no sentido de transmissão.

6.18.1.4.3 - Módulo de Execução

Este módulo é responsável pelo encaminhamento dos eventos às camadas do aplicativo ATM. Esta função é desempenhada pelo procedimento *Run*. A Figura 6.62 mostra um fluxograma deste procedimento.

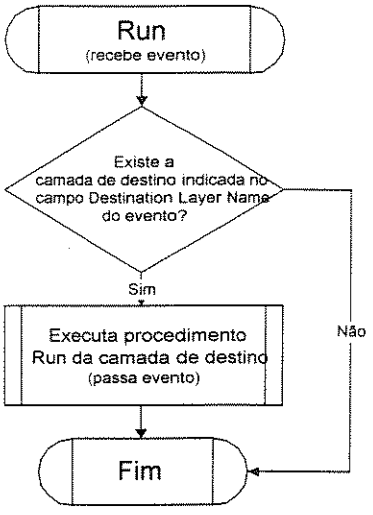


Figura 6.62 – Fluxograma do procedimento *Run*

Quando o *kernel* do simulador possui um evento destinado a um aplicativo, ele aciona o procedimento *Run* do módulo de execução deste aplicativo. Este procedimento irá repassar o evento para uma das camadas do aplicativo. Se a camada de destino do evento existir, então ele será repassado para o procedimento *Run* desta camada.

6.18.1.4.4 - Módulo de Desconexão

Este módulo é acionado quando o aplicativo está sendo desconectado de um BTE. Quando o módulo de desconexão é acionado, os modelos de camadas criados pelo módulo de conexão são removidos. Também é removido o registro feito na tabela do aplicativo.

6.18.1.4.5 - Módulo de Terminação

Este módulo não executa nenhuma função.

6.19 - Conexões ATM

Conforme vimos no Capítulo 2, as conexões ATM podem ser permanentes (PVCs) ou chaveadas (SVCs), ponto à ponto ou ponto para multiponto, unidirecionais ou bidirecionais. No **SimATM**, as conexões ATM são conexões de canal virtual (VCCs – *Virtual Channel Connection*) permanentes, ponto à ponto e unidirecionais. Para cada conexão ATM é possível especificar a categoria de serviço, a largura de faixa e os equipamentos ATM que estão no trajeto da conexão, bem como os identificadores de caminho virtual (VPI) e de canal virtual (VCI) para cada enlace de canal virtual (VCL – *Virtual Channel Link*) da conexão. Cada conexão ATM possui um estado, que pode assumir os valores "EMPTY" (disponível) ou

"BUSY" (ocupada). Uma conexão ATM está disponível quando ela não está sendo utilizada para transportar dados entre aplicativos ATM.

As conexões ATM são construídas a partir de registros feitos nas tabelas dos equipamentos ATM que fazem parte do trajeto das conexões. O estabelecimento de conexões ATM será exemplificado no item 6.21.

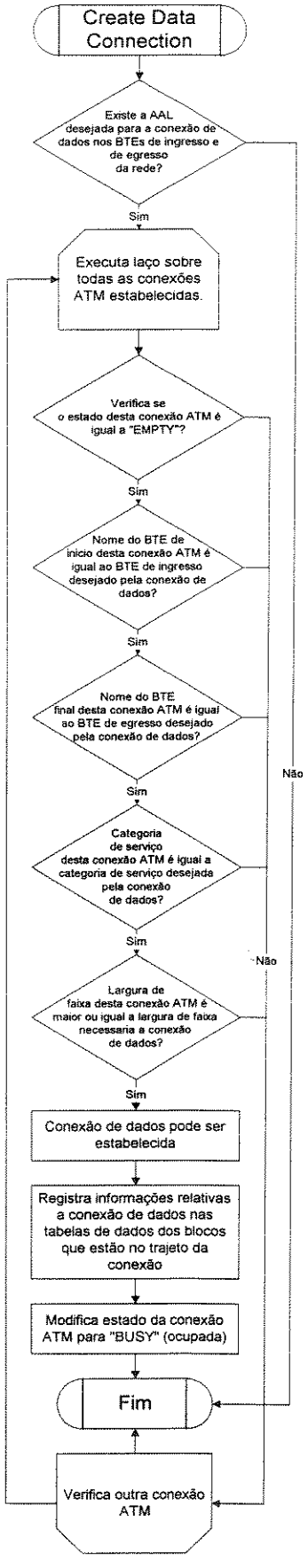
6.20 - Conexões de Dados

As conexões de dados são conexões estabelecidas entre dois aplicativos ATM, que se utilizam de conexões ATM previamente estabelecidas para enviar informações através de uma rede ATM. Uma conexão de dados só é estabelecida se houver uma conexão ATM que possa atender as suas necessidades. Para cada conexão de dados é possível especificar o nome dos aplicativos fonte e destino, o nome dos BTEs de ingresso e egresso, a categoria de serviço, a largura de faixa e o nome da conexão ATM a ser utilizada.

Assim como as conexões ATM, as conexões de dados também são construídas a partir de registros nas tabelas dos blocos que fazem parte do trajeto da conexão. O estabelecimento de conexões de dados será exemplificado no item 6.21.

6.20.1 - Utilizando uma Conexão ATM

Para que uma conexão de dados seja estabelecida é preciso que exista uma conexão ATM capaz de atender os seus pré-requisitos. A Figura 6.63 mostra um fluxograma do procedimento de estabelecimento de uma conexão de dados, chamado de *Create Data Connection*.



Inicialmente é verificado se a AAL a ser utilizada pelos aplicativos fonte e destino da conexão de dados existe nos BTEs de ingresso e de egresso da conexão. Em seguida é executado um laço sobre todas as conexões ATM existes na rede, a fim de encontrar uma conexão que possa ser utilizada para transmitir as informações da conexão de dados. Se o estado de uma destas conexões for "EMPTY" (disponível), então prossegue-se a verificação de outras condições necessárias para a utilização desta conexão. É verificado se o nome do BTE onde inicia a conexão ATM é igual ao nome do BTE de ingresso desejado para a conexão de dados, se o nome do BTE onde termina a conexão ATM é igual ao nome do BTE de egresso desejado para a conexão de dados, se a categoria de serviço da conexão ATM é igual a categoria de serviço desejada para a conexão de dados e se a largura de faixa da conexão ATM é maior ou igual a largura de faixa desejada para a conexão de dados. Se todas estas condições forem atendidas, então está conexão ATM será utilizada para transportar as informações da conexão de dados, que já pode ser estabelecida. O estado desta conexão ATM é então modificado para "BUSY" (ocupada). São feitos ainda vários registros nas tabelas de dados dos blocos que estão no trajeto da nova conexão. Se uma das condições acima não for atendida, uma outra conexão ATM é procurada. Se nenhuma das conexões ATM puder atender os pré-requisitos necessários para o estabelecimento da conexão de dados, esta conexão não poderá ser estabelecida.

Figura 6.63 – Fluxograma do procedimento de estabelecimento de uma conexões de dados

6.21 - Exemplo de Montagem de uma Rede ATM

A montagem de uma rede ATM pode ser dividida em cinco fases: criação dos blocos, conexão dos blocos, criação de conexões ATM, criação de enlaces (conexões de canais virtuais) para as conexões ATM e criação de conexões de dados. Somente após a execução destas fases uma rede ATM poderá ser simulada. A Figura 6.64 mostra um exemplo de uma rede ATM que utilizaremos para exemplificar a montagem de redes no **SimATM**.

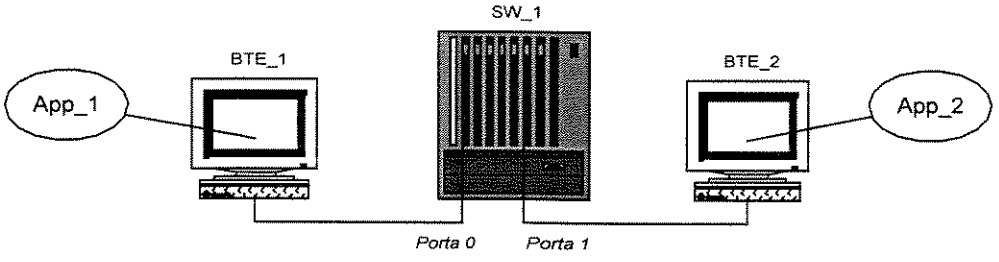


Figura 6.64 – Exemplo de rede ATM

A primeira fase da montagem da rede consiste da criação dos blocos. No caso do nosso exemplo, são criados dois aplicativos ATM chamados App_1 e App_2, dois BTEs chamados BTE_1 e BTE_2 e um chaveador ATM chamado de SW_1.

Na segunda fase de montagem da rede estes blocos devem ser conectados. O aplicativo App_1 é conectado ao terminal BTE_1, que por sua vez é conectado ao chaveador SW_1, na porta 0. O chaveador SW_1 é conectado ao terminal BTE_2, que se conecta ao aplicativo App_2.

Na terceira fase de montagem da rede é feita a criação das conexões ATM. No caso do nosso exemplo, criamos uma conexão ATM chamada VCC_1, que vai do BTE_1 até o BTE_2, passando pelo SW_1. A conexão VCC_1 possui largura de faixa de 155.52 Mbps e utiliza a categoria de serviço CBR.

Na quarta fase de montagem da rede são criados os enlaces (conexões de canal virtual) para as conexões ATM. No caso do nosso exemplo, são criados os enlaces para a conexão VCC_1. Para o enlace entre o BTE_1 e o SW_1, o identificador de caminho virtual é configurado para VPI=0 e o identificador de canal virtual é configurado para VCI=33. Para o enlace entre o SW_1 e o BTE_2, o identificador de caminho virtual é configurado para VPI=0 e o identificador de canal virtual é configurado para VCI=34.

Na quinta e última fase de montagem da rede são criadas as conexões de dados. No caso do nosso exemplo, criamos uma conexão de dados chamada DC_1. Esta conexão inicia

no aplicativo App_1 e termina no aplicativo App_2. A conexão DC_1 possui largura de faixa de 29 Mbps e utiliza a categoria de serviço CBR.

Durante a execução das fases de montagem da rede, as tabelas dos blocos e da rede ATM são modificadas. A Figura 6.65 mostra os registros que são realizados nas tabelas dos blocos da rede da Figura 6.64 durante as fases de montagem da rede, enquanto a Figura 6.66 mostra os registros feitos na tabela da rede ATM da Figura 6.64 também durante as fases de montagem desta rede.

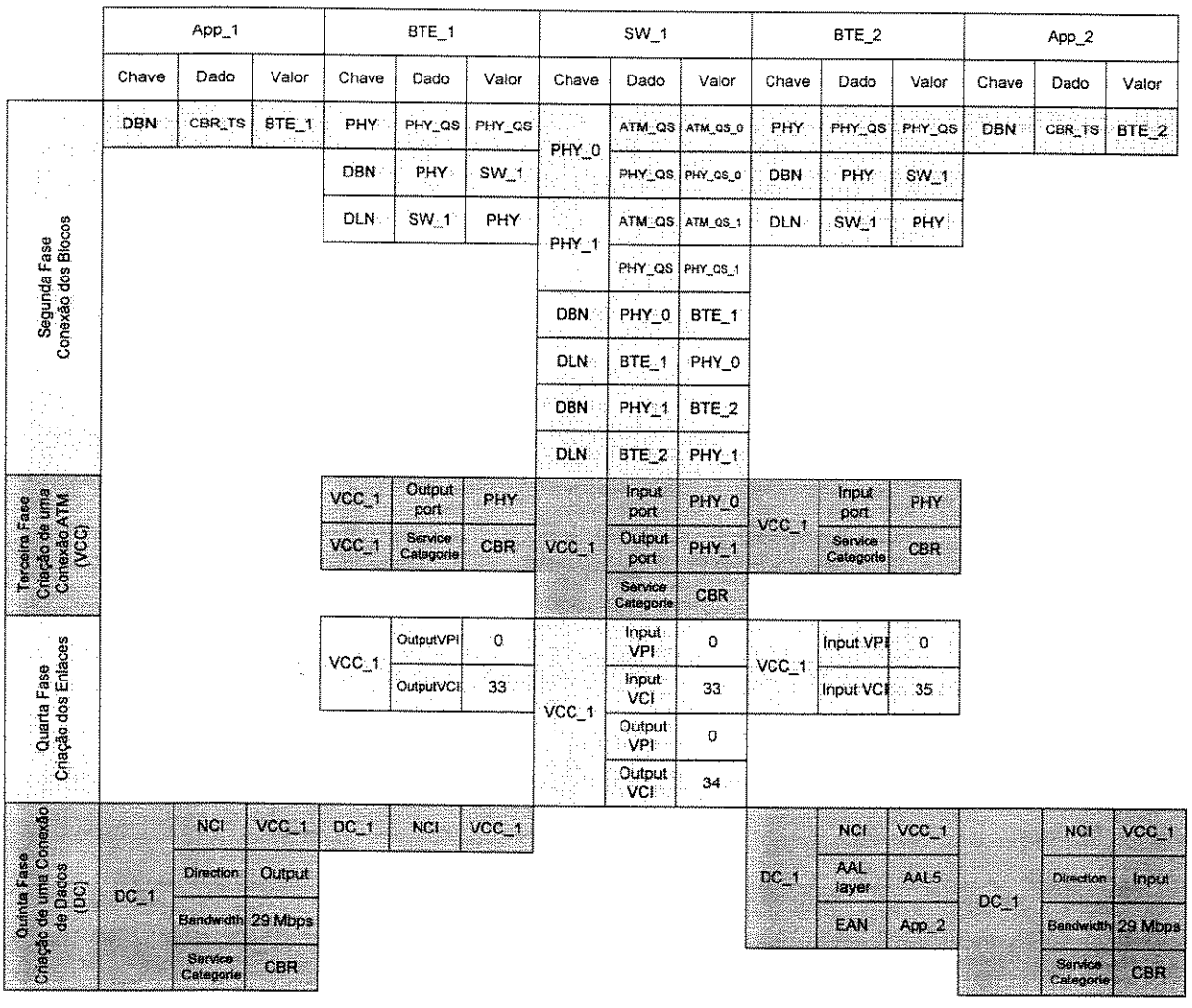


Figura 6.65 – Tabelas dos blocos durante as fases de montagem da rede ATM

	Rede		
	Chave	Dado	Valor
Segunda Fase Conexão dos Blocos	Interfaces	Interface_BTE_2_App_2	-
	Interfaces	Interface_SW_1_BTE_2	CB_PHYSICAL_LAYER
	Interfaces	Interface_BTE_1_SW_1	CB_PHYSICAL_LAYER
	Interfaces	Interface_App_1_BTE_1	-
Terceira Fase Criação de uma Conexão ATM (VCC)	VCC_1	Status	BUSY
	VCC_1	Service Categorie	CBR
	VCC_1	Bandwidth	155.52 Mbps
	VCC_1	Block_1	BTE_1
	VCC_1	Number of Blocks	3
	VCC_1	Number of Links	2
	VCC_1	Block_2	SW_1
	VCC_1	Block_3	BTE_2
Quarta Fase Criação dos Enlaces	VCC_1	Link_1_Begin	BTE_1
	VCC_1	Link_1_End	SW_1
	VCC_1	Link_1_VPI	0
	VCC_1	Link_1_VCI	33
	VCC_1	Link_2_Begin	SW_1
	VCC_1	Link_2_End	BTE_2
	VCC_1	Link_2_VPI	0
	VCC_1	Link_2_VCI	34
Quinta Fase Criação de uma Conexão de Dados (DC)	DC_1	NCI	VCC_1
	DC_1	Begin application name	App_1
	DC_1	End application name	App_2
	DC_1	Begin block name	BTE_1
	DC_1	End block name	BTE_2
	DC_1	Bandwidth	29 Mbps
	DC_1	Service Categorie	CBR

Figura 6.66 – Tabela da rede durante as fases de montagem da rede ATM

A execução das fases de montagem de uma rede ATM deve ser feita através de comandos disponíveis para os usuários do **SimATM**. A seguir descreveremos a linguagem de comandos do simulador.

6.22 - Linguagem de Comandos

O **SimATM** possui uma linguagem de comados. Através desta linguagem é possível montar a rede ATM que será simulada e executar a simulação. Os comandos disponíveis podem ser executados a partir da linha de comando (*prompt*) ou a partir da leitura e interpretação de um arquivo de rede. Este arquivo possui a extensão *.net*. A configuração de uma rede ATM previamente montada pode ser salva para arquivo através do comando *save* e recarregada através do comando *load*. É possível a montagem de uma rede ATM diretamente através do uso do arquivo de rede. Neste caso, os comandos devem ser colocados no arquivo de acordo com as fases de montagem de rede apresentadas anteriormente.

A versão atual do **SimATM** não possui interface gráfica. O desenvolvimento de uma interface gráfica para o simulador permitirá que estes comandos sejam executados de forma transparente para o usuário final do simulador. Desta forma, será possível modificar a configuração da rede e os seus parâmetros mais facilmente.

No Apêndice B apresentaremos os comandos disponíveis na versão atual da linguagem de comandos do **SimATM**.

6.23 - Referências Bibliográficas

[1] W. Ford, W. Topp, "*Data Structures with* C++", *Prentice-Hall, Inc.*, 1996.

[2] Timothy A. Budd, "*Classic Data Structures in* C++", *Addison-Wesley*, 1994.

[3] Douglas Comer, "*Interneworking with* TCP/IP", *Third Edition, Prentice-Hall*, 1995.

[4] Leonard Kleinrock, "*Queueing Systems – Volume* I: *Theory*", *John Wiley & Sons*, 1975.

[5] Thomas G. Robertazzi, "*Computer Networks and Systems – Queueing Theory and Perfomance Evaluation*", *Second Edition, Springer-Verlag*, 1994.

[6] ITU-T *Recommendation* I.363, "B-ISDN ATM *Adaptation Layer* (AAL) *Specification*", *Helsinky*, 1993.

[7] ITU-T *Recommendation* I.361, "B-ISDN ATM *Layer Specification*", *Helsinki*, 1993.

Capítulo 7

Implementação do **SimATM**

Neste capítulo descreveremos como o **SimATM** foi implementado.

7.1 - Introdução

O **SimATM** foi implementado utilizando-se a linguagem de programação C++ [1][2] e, portanto, usufrui das vantagens da programação orientada a objetos (OOP – *Object Oriented Programming*) [3]. Na implementação do **SimATM** foram utilizados recursos avançados da linguagem C++, tais como: encapsulação, herança múltipla, polimorfismo, *friends*, sobrecarga de operadores e *templates*. Os elementos da estrutura do simulador, apresentados no Capítulo 6, são implementados como classes da linguagem C++. A utilização de classes e do recurso de herança do C++ permitiu a implementação da estrutura do simulador de forma modular e hierárquica.

O **SimATM** foi implementado a partir da definição de um conjunto completamente novo de objetos. O desenvolvimento destes novos objetos foi inspirado nos objetos utilizados no ambiente de simulação de sistemas de comunicação **SimNT**, anteriormente apresentado no item 5.1.1, e nas estruturas de dados utilizadas no simulador de redes ATM desenvolvido pelo NIST, anteriormente apresentado no item 5.1.2. Nenhum código fonte em linguagem C do simulador de redes ATM desenvolvido pelo NIST foi utilizado na implementação do **SimATM**.

Os objetos desenvolvidos para o **SimATM** utilizam estruturas de dados, tais como: vetores, listas, dicionários[1], filas com prioridade, etc., cujo código fonte é disponibilizado na referência [4].

A seguir descreveremos a implementação de cada um dos elementos da estrutura do **SimATM** na linguagem C++. Maiores detalhes desta linguagem podem ser obtidos a partir das referências [1] e [2].

---

[1] Um dicionário é uma estrutura de dados que possui um conjunto de chaves e valores. Para cada chave, que em geral é do tipo `string`, é associado um valor que pode ser de qualquer tipo. O código fonte da estrutura de dados dicionário pode ser obtido a partir da referência [4].

7.2 - Kernel

O *kernel* do **SimATM** é implementado na classe `Kernel`. A classe `Kernel` possui como dados-membro privados [1][2]: o tempo de simulação (`double`), a fila de eventos (`EventPriorityQueue`), três ponteiros para arquivos de desempenho de simulação (`ofstream*`) e ainda duas estruturas de dados dicionário: `Networks` e `SaveFileNames`. Um dicionário possui uma chave e um valor. O dicionário `Networks` armazena as instâncias de rede ATM do *kernel* na forma de ponteiros para a classe `Network`. Para cada rede ATM é associado um nome ou chave (*key*), que consiste de uma seqüência de caracteres (`string`). O dicionário `SaveFileNames` armazena o nome dos três arquivos de desempenho de simulação para cada rede ATM simulada.

A classe `Kernel` implementa o módulo interpretador de comandos e o módulo gerenciador de eventos do *kernel*. O módulo interpretador de comandos e os comandos da linguagem do **SimATM** são implementados através de funções-membro da classe `Kernel`. O módulo gerenciador de eventos também é implementado como uma função membro da classe `Kernel` chamada `Run`. Esta função é executada pelo interpretador de comandos quando o comando ***run*** é digitado.

7.3 - Redes ATM

A estrutura das redes ATM do simulador são implementadas na classe `Network`. A classe `Network` herda as classes `Param` e `DataTable`. Desta forma são armazenados e manipulados os parâmetros e dados de tabela de uma rede ATM. A Figura 7.1 mostra um diagrama de herança [3] para a classe `Network`.

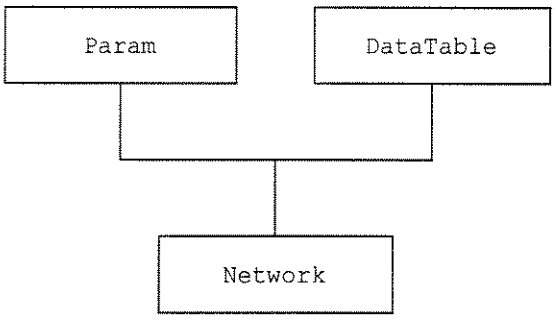


Figura 7.1 – Diagrama de herança da classe `Network`

A classe `Network` possui como dados-membros privados um ponteiro para a fila de eventos (`EventPriorityQueue*`) e duas estruturas de dados dicionário: `Blocks` e `Neighbors`. O dicionário `Blocks` armazena todas as instâncias de blocos da rede. A chave do dicionário é o nome de um bloco (`string`) e o valor contém um ponteiro para este bloco

(`Block*`). O dicionário `Neighbors` armazena informações de vizinhança dos blocos da rede. A chave do dicionário é o nome de um bloco (`string`) e o valor é um ponteiro para uma lista duplamente terminada (`doubleEndedList*`) [4][5] de ponteiros para blocos (`Block*`). O dicionário `Neighbors` contém uma lista de ponteiros para os blocos vizinhos de cada bloco da rede.

Os módulos de inicialização, terminação, gerenciamento de parâmetros, gerenciamento de tabelas de dados, gerenciamento de blocos, gerenciamento de conexões ATM, gerenciamento de conexões de dados e de gerenciamento de vizinhos da estrutura das redes ATM são implementados como funções-membro públicas da classe `Network`.

7.4 - Eventos

Os eventos do **SimATM** são implementados na classe `Event`. Essa classe utiliza o recurso de *macros* do C++, que possibilita a definição de uma mesma função para vários tipos de dados ou, no caso, para vários tipos de mensagens. Esses tipos de mensagens devem ser definidos no arquivo `Messdefs.h`. A versão atual do **SimATM** está limitada a um número máximo de 20 tipos de mensagens. Os tipos *default* de mensagens são: `CPCS_UNITDATA`, `SAR_UNITDATA`, `ATM_DATA`, `CELL` e `PHY_DATA`.

A classe `Event` possui funções-membro que permitem consultar e modificar os valores dos campos dos eventos, especialmente o campo *message*. A classe `Event` possui ainda quatro construtores, que facilitam a geração de eventos nas camadas e sistemas de fila dos blocos da rede. A Figura 7.2 ilustra o formato desses construtores.

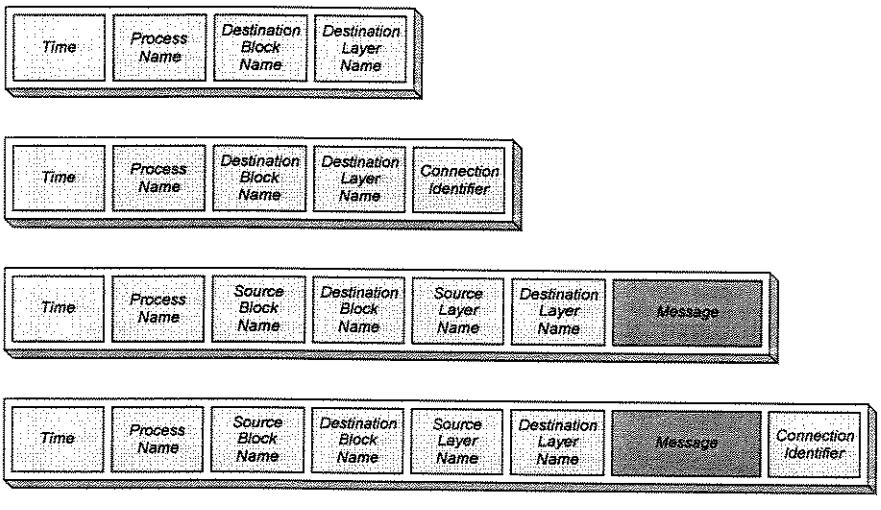


Figura 7.2 – Construtores da classe `Event`

Os dois primeiros construtores ilustrados na Figura 7.2 são utilizados para gerar eventos internos, que não carregam mensagens. Esses eventos servem para disparar processos de manipulação de células, processos estatísticos, etc. Os dois últimos construtores carregam mensagens e são utilizados para gerar tanto eventos internos, como eventos externos. Esses construtores são definidos para cada tipo de mensagem através do uso de uma *macro*.

A classe `Event` sobrecarrega o operador < de forma a permitir a comparação dos eventos a partir do campo *Time*. Se esse campo for igual nos dois eventos que estão sendo comparados, então é feita a comparação a partir do campo *Token*.

7.5 - Fila de Eventos

A fila de eventos foi implementada utilizando-se a estrutura de dados: fila com prioridade (*Priority Queue*) [5][4]. Filas com prioridade são usuais em problemas onde é importante encontrar e remover o menor valor dentre uma coleção de valores o mais rápida e eficientemente possível. Segundo *Budd* [4], duas estruturas de dados podem ser usadas para implementar eficientemente filas com prioridade, são elas: *heap* e *skew heap*. A estrutura *heap* tem como desvantagem a necessidade de declaração do tamanho máximo da fila quando ela é inicializada, o que inviabilizou esta solução para a fila de eventos, pois é muito difícil prever de antemão o número máximo de eventos que precisariam ser armazenados durante uma simulação ATM. Na estrutura *skew heap* não é necessário declarar o tamanho máximo da fila quando ela é inicializada, entretanto a solução de um problema introduz outro: a dificuldade de manter a estrutura em árvore, utilizada para implementar esta fila, balanceada. O fato desta estrutura estar desbalanceada implica, no pior caso, em um tempo de inserção e remoção de elementos relativamente lento. Entretanto, segundo *Budd* [4], este pior caso não ocorre freqüentemente, o que em média garante uma boa operação. Escolhemos a estrutura *skew heap*[2] para implementar a fila de eventos do **SimATM**.

A fila de eventos é implementada na classe `EventPriorityQueue`. Essa classe possui funções-membro que permitem o armazenamento de eventos, a consulta ou remoção do evento de maior prioridade da fila e a verificação do estado da fila (vazia ou ocupada).

A classe `EventPriorityQueue` possui também um contador dos eventos armazenados na fila. Toda vez que um evento é armazenado este contador é incrementado e o campo de informações *Token* do evento é configurado com o valor atual do contador. A

[2] O código fonte da estrutura de dados *skew heap* pode ser obtido a partir da referência [4].

prioridade de execução deste novo evento é dada a partir de uma comparação entre o tempo de execução desejado e o tempo de execução dos eventos já armazenados na fila. Se houver um empate, ou seja, se já existirem eventos agendados para o mesmo instante de tempo, a prioridade será determinada a partir da comparação do campo *Token* destes eventos. Desta forma, garante-se a execução de eventos agendados para um mesmo instante de tempo segundo a disciplina FIFO.

7.6 - Parâmetros

Os parâmetros do **SimATM** são implementados na classe `Param`, que foi originalmente desenvolvida para o ambiente **SimNT**, descrito no Capítulo 5.

Essa classe também utiliza o recurso de *macros* do C++. Os tipos de dados, que podem ser utilizados para armazenar o campo *Valor* de um parâmetro, são definidos no arquivo `Simdefs.h`, que contém os tipos básicos utilizados no simulador. A versão atual do **SimATM** está limitada a um número máximo de 20 tipos básicos de dados. Os tipos *default* são: `int`, `double`, `complex`, `long`, `string`, `unsigned char`, `short int`, `unsigned int`, e `float`.

A classe `Param` utiliza uma estrutura de dados dicionário[3] [5][4] para o armazenamento dos parâmetros. Para cada parâmetro é associado um nome ou chave (*key*), que consiste de uma seqüência de caracteres (`string`).

As principais funções-membro da classe `Param` estão relacionadas com a armazenagem, consulta e remoção de parâmetros e de seus dados privados. As funções de armazenagem e consulta de parâmetros são implementadas para cada tipo básico de dados do simulador. A classe `Param` possui ainda funções-membro para a manipulação de parâmetros a partir da interface gráfica do **SimNT**. Possivelmente essas funções também serão utilizadas na interface gráfica do **SimATM**.

7.7 - Tabela de Dados

A tabela de dados é implementada na classe `DataTable`. Esta classe possui como dados-membros privados: o nome da tabela, a estrutura `Data` e a estrutura `Type`. A estrutura

[3] O código fonte desta estrutura de dados dicionário faz parte do código do simulador **SimNT**.

`Data` é implementada através de duas estruturas de dados dicionário[4] encadeadas, enquanto a estrutura `Type` é implementada através de uma única estrutura de dados dicionário. Cada uma dessas estruturas de dados dicionário tem uma chave que permite acessar um valor. Tanto a chave como o valor podem ser de qualquer tipo, uma vez que esta estrutura dicionário é implementada utilizando-se o recurso de *templates* do C++.

O primeiro dicionário da estrutura `Data` tem como chave uma variável do tipo `string` (nome da chave na tabela) e como valor a segunda estrutura dicionário. Essa segunda estrutura dicionário também tem como chave uma variável do tipo `string` (nome do dado na tabela) e como valor um ponteiro do tipo `void` [1][2] (aponta para o valor do dado na tabela), que permite o armazenamento de valores com qualquer tipo. Entretanto, o tipo desses valores deve ser armazenado a fim de possibilitar a recuperação dos mesmos. Para isso, utiliza-se a estrutura `Type`. A chave desse dicionário é o nome do dado na tabela e o valor consiste do tipo desse dado.

A classe `DataTable` possui funções-membro para armazenar e consultar um registro na tabela. Essas funções são implementadas utilizando-se *macros* e, portanto, cada função é definida para os tipos básicos do simulador. Desta forma, quando uma nova chave é inserida na tabela, é possível determinar o tipo do valor que está sendo inserido, e armazená-lo na estrutura `Type`. Quando é feita uma consulta ao valor de um dado da tabela, o conteúdo apontado pelo ponteiro `void` é convertido (*casting*) para o tipo previamente armazenado na estrutura `Type`.

A classe `DataTable` possui ainda funções-membro que permitem remover chaves e dados da tabela, bem como realizar procuras. As funções de procura permitem determinar:

- O nome da chave que possui um determinado dado, através do fornecimento do seu nome e do seu valor.
- O nome da chave que possui um determinado par de dados, através do fornecimento dos seus nomes e dos seus valores.
- O nome de todas as chaves que possuem um mesmo dado, através do fornecimento do seu nome e do seu valor.

[4] O código fonte desta estrutura de dados dicionário pode ser obtido a partir da referência [4].

- O nome de todas as chaves que possuem um mesmo par de dados, através do fornecimento dos seus nomes e dos seus valores.
- O nome de todas as chaves que possuem um mesmo nome de dado.
- Se existe uma determinada chave na tabela.
- Se existe o nome de um determinado dado armazenado sob uma dada chave.

7.8 - Unidades de Dados

7.8.1 - CPCS-SDU

A CPCS-SDU é implementada na classe `CPCS_SDU`. A classe `CPCS_SDU` possui como dados-membro privados variáveis correspondentes aos campos da sua estrutura, apresentada no Capítulo 6. As funções-membro da classe `CPCS_SDU` permitem consultar e modificar os seus dados-membro privados.

7.8.2 - CPCS-PDU

A CPCS-PDU é implementada na classe `CPCS_PDU`. A classe `CPCS_PDU` possui como dados-membro privados variáveis correspondentes aos campos da sua estrutura, apresentada no Capítulo 6. O campo *Payload* é implementado como um ponteiro para um objeto do tipo `CPCS_SDU`. As funções-membro da classe `CPCS_PDU` permitem consultar e modificar os seus dados-membro privados. A classe `CPCS_PDU` possui ainda dois construtores: um para inicializar os seus dados privados quando a CPCS-PDU é criada, e outro para realizar a cópia a partir de uma CPCS-PDU já existente.

7.9 - Primitivas

7.9.1 - Primitiva CPCS-UNITDATA *Invoke* e *Signal*

Estas primitivas são implementadas como um único objeto que é definido na classe `CPCS_UnitData`. A estrutura destas primitivas já foi apresentada no item 2.3.5.1. A Figura 7.3 mostra o formato da primitiva implementada no **SimATM**. Os campos da primitiva da figura correspondem aos dados-membro privados da classe `CPCS_UnitData`. O campo *Interface Data* é um ponteiro para um objeto do tipo `CPCS_SDU`.

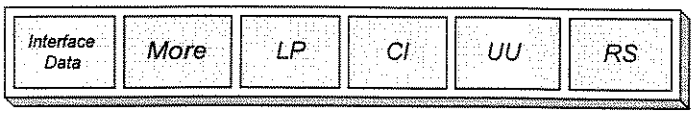


Figura 7.3 – Formato da primitiva CPCS-UNITDATA no SimATM

As funções-membro da classe `CPCS_UnitData` permitem consultar e modificar os seus dados-membro privados. A classe `CPCS_UnitData` possui ainda dois construtores: um para inicializar os seus dados privados quando a primitiva é criada, e outro para realizar a cópia a partir de uma primitiva já existente.

7.9.2 - Primitiva SAR-UNITDATA *Invoke* e *Signal*

Estas primitivas são implementadas como um único objeto que é definido na classe `SAR_UnitData`. A estrutura destas primitivas já foi apresentada no item 2.3.5.2. A Figura 7.4 mostra o formato da primitiva implementada no **SimATM**. Os campos da primitiva da figura correspondem aos dados-membro privados da classe `SAR_UnitData`. O campo *Interface Data* é um ponteiro para um objeto do tipo `CPCS_PDU`.

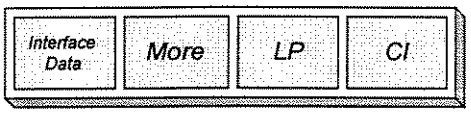


Figura 7.4 – Formato da primitiva SAR-UNITDATA no SimATM

As funções-membro da classe `SAR_UnitData` permitem consultar e modificar os seus dados-membro privados. A classe `SAR_UnitData` possui ainda dois construtores: um para inicializar os seus dados privados quando a primitiva é criada, e outro para realizar a cópia a partir de uma primitiva já existente.

7.9.3 - Primitiva ATM-DATA *Request* e *Indication*

Estas primitivas são implementadas como um único objeto que é definido na classe `ATM_Data`. A estrutura destas primitivas já foi apresentada no item 2.3.5.3. A Figura 7.5 mostra o formato da primitiva implementada no **SimATM**. Os campos da primitiva da figura correspondem aos dados-membro privados da classe `ATM_Data`. O campo *Interface Data* é um ponteiro para um objeto do tipo `CPCS_PDU`.

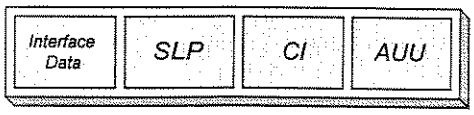


Figura 7.5 – Formato da primitiva ATM-DATA no SimATM

As funções-membro da classe `ATM_Data` permitem consultar e modificar os seus dados-membro privados. A classe `ATM_Data` possui ainda dois construtores: um para inicializar os seus dados privados quando a primitiva é criada e outro para realizar a cópia a partir de uma primitiva já existente.

7.9.4 - Primitiva PHY-DATA *Request* e *Indication*

Estas primitivas são implementadas como um único objeto que é definido na classe `PHY_Data`. A estrutura destas primitivas já foi apresentada no item 2.3.5.4. A Figura 7.6 mostra o formato da primitiva implementada no **SimATM**. Os campos da primitiva da figura correspondem aos dados-membro privados da classe `PHY_Data`. O campo *Interface Data* é um ponteiro para um objeto do tipo `Cell`.

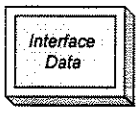


Figura 7.6 – Formato da primitiva PHY-DATA no SimATM

As funções-membro da classe `PHY_Data` permitem consultar e modificar os seus dados-membro privados. A classe `PHY_Data` possui ainda dois construtores: um para inicializar os seus dados privados quando a primitiva é criada e outro para realizar a cópia a partir de uma primitiva já existente.

7.10 - Células ATM

As células do **SimATM** são implementadas na classe `Cell`. Os campos da célula ATM são implementados como dados-membro privados dessa classe. O campo `PCPCS-PDU` é um ponteiro para um objeto do tipo `CPCS-PDU`.

A classe `Cell` possui funções-membro que permitem consultar e modificar os valores dos campos das células, bem como quatro construtores, que facilitam a criação de células. A classe `Cell` possui ainda um construtor de cópia. A Figura 7.7 ilustra o formato dos construtores disponíveis.

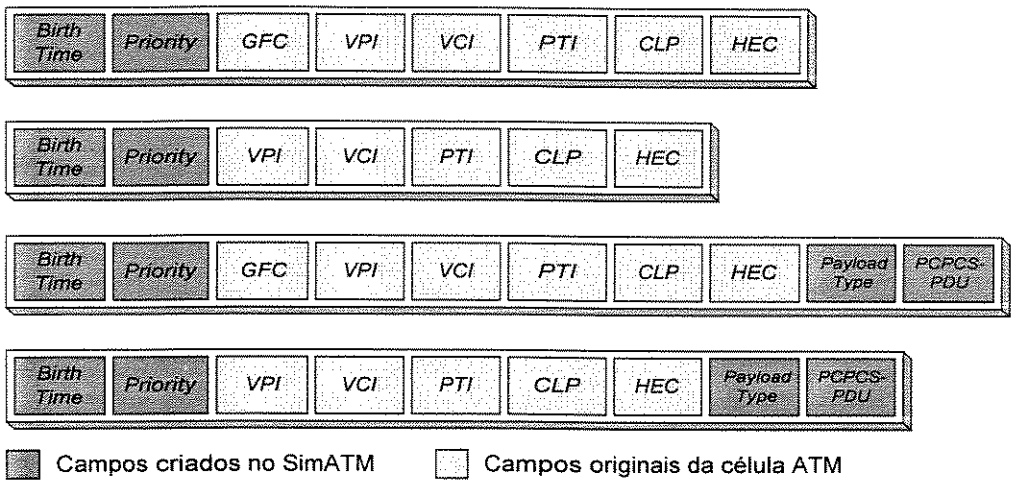


Figura 7.7 – Construtores da classe `Cell`

O primeiro construtor é usado para criar células vazias na interface UNI, enquanto o segundo construtor é usado para criar células vazias na interface NNI. O terceiro construtor é usado para criar células de usuário na interface UNI e o último construtor é usado para criar células de usuário na interface NNI.

A classe `Cell` sobrecarrega o operador < de forma a permitir a comparação das células a partir do campo *Priority*. Se esse campo for igual em ambas as células que estão sendo comparadas a comparação é feita a partir do campo *QueueArrivalTime*.

7.11 - Fila de Células

Assim como a fila de eventos, a fila de células foi implementada utilizando-se a estrutura de dados *skew heap* [5][4]. A fila de células é implementada na classe `CellPriorityQueue`. Essa classe possui funções-membro que permitem o armazenamento de células, a consulta ou remoção da célula de maior prioridade da fila, e a verificação do estado atual da fila (vazia ou ocupada).

Ao contrário da classe `EventPriorityQueue`, a classe `CellPriorityQueue` não possui um contador de células armazenadas na fila. Ao invés disto, toda vez que uma célula é armazenada na fila, o campo *QueueArrivalTime* desta célula é configurado com o instante tempo em que foi feito o armazenamento. A prioridade de atendimento dessa nova célula é dada a partir de uma comparação entre a prioridade desta célula e a prioridade das células já armazenadas na fila. Se houver um empate, ou seja, se já existirem células com a mesma prioridade, o atendimento será determinado a partir da comparação do campo *QueueArrivalTime* destas células. Desta forma, garante-se o atendimento das células de

mesma prioridade de acordo com a sua ordem de chegada na fila, ou seja, segundo a disciplina FIFO.

7.12 - Sistemas de Fila

O sistema de fila é implementado na classe `QueueingSystem`. Essa classe herda a classe `Param` e, portanto, todas as suas funções-membro. A Figura 7.8 mostra um diagrama de herança para a classe `QueueingSystem`.

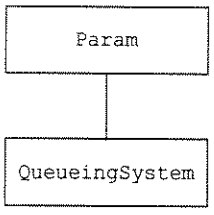


Figura 7.8 – Diagrama de herança da classe `QueueingSystem`

A classe `QueueingSystem` possui como dados-membro privados: o nome do sistema de fila, um ponteiro para o bloco (`Block*`) a que o sistema pertence, variáveis para a manipulação de parâmetros, variáveis de estado, variáveis com nomes de arquivos, arquivos de amostragem de variáveis de estado e instâncias de uma classe chamada `Statistics`, que é utilizada para implementar o cálculo de estatísticas de variáveis de estado.

A classe `QueueingSystem` possui como dados-membro protegidos: um ponteiro para a fila de eventos (`EventPriorityQueue*`), um ponteiro para o evento (`Event*`) a ser executado, um ponteiro para a célula que está sendo servida (servidor) e uma instância do objeto `CellPriorityQueue` (fila de células).

7.13 - Camadas

O elemento **camada** da estrutura do **SimATM** é implementado na classe `Layer`. A classe `Layer` herda as classes `Param` e `DataTable`. Desta forma são armazenados e manipulados os parâmetros e dados de tabela das camadas. A Figura 7.9 mostra um diagrama de herança para a classe `Layer`.

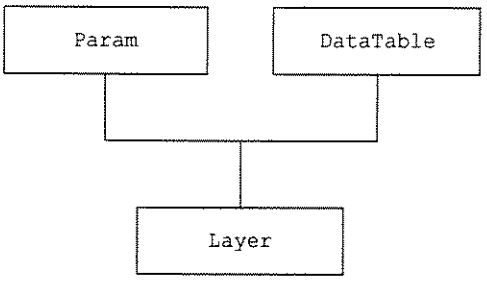


Figura 7.9 – Diagrama de herança da classe `Layer`

A classe `Layer` possui como dados-membro privados o nome e o tipo da camada (ambos do tipo `string`), e como dados-membro protegidos: um ponteiro para o bloco

(`Block*`) a que a camada pertence, um ponteiro para o evento (`Event*`) a ser executado e um ponteiro para a fila de eventos (`EventPriorityQueue*`).

A classe `Layer` possui funções-membro virtuais puras, ou seja, que só precisam ser implementadas nas classes derivadas da classe `Layer`. Devido a essa característica a classe `Layer` é dita abstrata [1][2]. Uma classe abstrata é uma classe esquemática que é utilizada para construir uma ou mais classes derivadas. Portanto, não é possível criar uma instância de uma classe abstrata.

Os módulos de inicialização, execução e terminação da estrutura das camadas do **SimATM** são implementados através de funções-membro virtuais puras. Essas funções são declaradas na classe `Layer` e implementadas nas classes derivadas da classe `Layer`. Para cada classe derivada da classe `Layer` é possível implementar de forma diferenciada cada um destes módulos. Desta forma é feita a diferenciação no comportamento dos vários modelos de camadas do simulador.

O módulo gerenciador de parâmetros da estrutura das camadas do **SimATM** é implementado através funções-membro que permitem consultar, modificar e remover os parâmetros do bloco e da rede ATM a que a camada pertence.

O módulo gerenciador de tabelas de dados é implementado através de funções-membro que permitem consultar, modificar e remover os dados de tabelas dos blocos e da rede ATM a que a camada pertence, bem como realizar procuras na tabela de dados do bloco a que a rede pertence e verificar a existência de um dado e de uma chave na tabela de dados do bloco a que a camada pertence.

A classe `Layer` possui ainda funções-membro que permitem consultar e modificar o nome da camada e o seu tipo, bem como consultar o nome, o tipo e o modelo do bloco a que a camada pertence. Existem ainda funções-membro públicas para consultar e modificar o evento a ser executado, bem como gerar novos eventos.

OBS. 1: As funções anteriores que manipulam valores de parâmetros e valores de dados de tabela são expandidas para todos os tipos básicos do simulador através do uso de *macros*.

OBS. 2: As funções anteriores que geram eventos cujas mensagens não são nulas, são expandidas para todos os tipos de mensagens do simulador também através do uso de *macros*. Os tipos de mensagens do **SimATM** encontram-se definidas no arquivo `Messdefs.h`.

7.14 - Modelos de Camadas

Os modelos de camadas do **SimATM** são implementados conforme as classes mostradas na Tabela 7.1.

Camada	Classe
Fonte de Tráfego CBR	`CBR_TS`
Fonte de Tráfego VBR Surtuoso	`VBR_BATCH_TS`
Fonte de Tráfego Genérica	`GENERIC_TS`
Receptor de Tráfego	`TrafficReceiver`
Camada de Adaptação ATM Tipo 5	`AAL5_Layer`
Camada ATM do Terminal Faixa Larga	`BTE_ATM_Layer`
Camada ATM do Chaveador	`Switch_ATM_Layer`
Camada Física para a Interface Baseada em Células	`CBPHY_Layer`

Tabela 7.1 – Classes dos modelos de camadas do SimATM

Todas as classes apresentadas na Tabela 7.1 são derivadas da classe `Layer` e, portanto, herdam os dados e funções-membro desta classe. As classes da Tabela 7.1 ainda implementam as funções-membro virtuais puras da classe `Layer`. A Figura 7.10 mostra um diagrama de herança dos modelos de camadas do simulador.

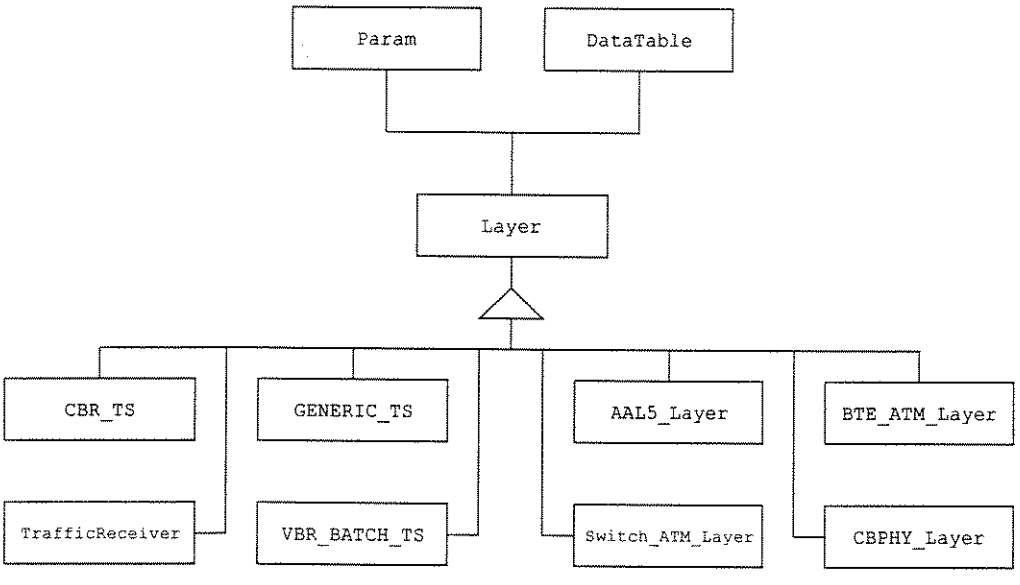


Figura 7.10 – Diagrama de herança dos modelos de camadas do SimATM

7.15 - Blocos

O elemento **bloco** da estrutura do **SimATM** é implementado na classe `Block`. Assim como a classe `Layer`, a classe `Block` herda as classes `Param` e `DataTable`. Desta forma são armazenados e manipulados os parâmetros e dados de tabela dos blocos. A Figura 7.11 mostra um diagrama de herança para a classe `Block`.

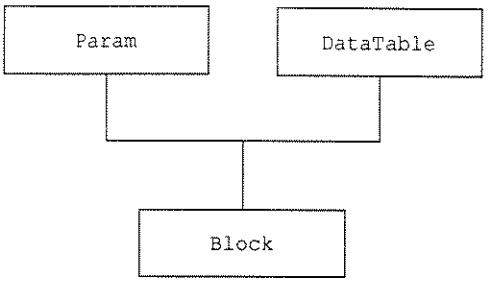


Figura 7.11 – Diagrama de herança da classe `Block`

A classe `Block` possui como dados-membro privados o nome e o tipo do bloco (`string`), e como dado membro protegido um ponteiro para a rede ATM (`Network*`) a que o bloco pertence.

Assim como a classe `Layer`, a classe `Block` também possui funções-membro virtuais puras e, portanto, também é uma classe abstrata. Essas funções virtuais puras são declaradas na classe `Block` e implementadas nas suas classes derivadas.

Os módulos de inicialização, conexão, execução, desconexão e terminação da estrutura dos blocos do **SimATM** são implementados através de funções-membro virtuais puras.

O módulo gerenciador de parâmetros da estrutura dos blocos do **SimATM** é implementado através de funções-membro e de funções-membro virtuais puras. As funções-membro permitem consultar, modificar e remover os parâmetros da rede ATM a que o bloco pertence. As funções-membro virtuais puras permitem consultar, modificar e remover os parâmetros das camadas e sistemas de fila de um bloco.

O módulo gerenciador de tabelas de dados da estrutura dos blocos do **SimATM** também é implementado através de funções-membro virtuais puras que permitem consultar, modificar e remover os dados de tabela das camadas de um bloco e ainda realizar procuras nessas tabelas.

O módulo gerenciador de camadas da estrutura dos blocos do **SimATM** também é implementado através de funções-membro virtuais puras que permitem acrescentar e remover camadas em um bloco, acionar os módulos de inicialização, execução e terminação das

camadas de um bloco, verificar a existência de uma dada camada e ainda consultar o nome de todas as camadas de um bloco.

O módulo gerenciador de sistemas de fila da estrutura dos blocos do **SimATM** também é implementado através de funções-membro virtuais puras que permitem acrescentar e remover sistemas de fila em um bloco, acionar os módulos de inicialização, execução e terminação dos sistemas de fila de um bloco, verificar a existência de um dado sistema de fila, consultar o nome de todos os sistemas de fila de um bloco, armazenar e remover células em um sistema de fila e ainda consultar o número de células total, no *buffer* e no servidor de um sistema de fila.

Cada uma das funções virtuais puras da classe `Block` é implementada de forma diferenciada nas classes derivadas da classe `Block`.

OBS.: As funções-membro anteriores que manipulam valores de parâmetros e valores de dados de tabela são expandidas para todos os tipos básicos do simulador através do uso de *macros*. Os tipos básicos do **SimATM** encontram-se no arquivo `Simdefs.h`.

7.16 - Equipamentos ATM

O elemento **equipamento ATM** da estrutura do **SimATM** é implementado na classe `ATMEquipment`, que é derivada da classe `Block`. Portanto, a classe `ATMEquipment` herda os dados e funções-membro dessa classe. A Figura 7.12 mostra um diagrama de herança para a classe `ATMEquipment`. A classe `ATMEquipment` possui como dados membro protegidos duas estruturas de dados dicionário: `Layers` e `QueueingSystems`, utilizadas para armazenar as camadas e sistemas de fila de um equipamento.

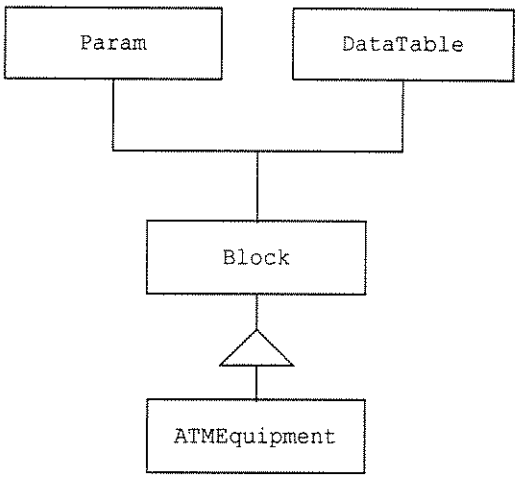


Figura 7.12 – Diagrama de herança da classe `ATMEquipment`

O dicionário `Layers` possui como chave uma `string` que contém o nome da camada armazenada, e como valor um ponteiro para objetos do tipo `Layer`. O dicionário `QueueingSystems` possui como chave uma `string` que contém o nome do sistema de fila armazenado, e como valor um ponteiro para objetos do tipo `QueueingSystem`. As instâncias das camadas e sistemas de fila de um equipamento são alocadas dinamicamente nas classes derivadas da classe `ATMEquipment`. Quando uma nova camada ou sistema de fila é criado, o ponteiro para esse objeto é armazenado na classe dicionário correspondente.

As funções-membro virtuais puras da classe `Block` cuja implementação é comum a todos os equipamentos da rede ATM são implementadas na classe `ATMEquipment`. Estas funções correspondem aos módulos gerenciador de parâmetros, gerenciador de tabelas de dados, gerenciador de camadas e gerenciador de sistemas de fila.

As funções-membro virtuais puras cuja implementação difere para cada equipamento da rede ATM são implementadas nas classes derivadas da classe `ATMEquipment`, que constituem os modelos de equipamentos da rede. Estas funções correspondem aos módulos de inicialização, conexão, execução, desconexão e de terminação. Entretanto, as funções-membro virtuais responsáveis pela criação e remoção de camadas e sistemas de fila, correspondentes aos gerenciadores de camadas e de sistemas de fila, respectivamente, são implementadas nas classes derivadas da classe `ATMEquipment`. A razão para a implementação destas funções nas classes derivadas da classe `ATMEquipament`, está na necessidade de criação e remoção de camadas e sistemas diferentes em cada modelo de equipamento ATM da rede.

7.17 - Modelos de Equipamentos ATM

Os modelos de equipamentos ATM do **SimATM** são implementados conforme as classes mostradas na Tabela 7.2.

Equipamento	**Classe**
Terminal Faixa Larga	`BTE`
Chaveador ATM	`Switch`

Tabela 7.2 – Classes dos modelos de equipamento do SimATM

As classes `BTE` e `Switch` são derivadas da classe `ATMEquipment` e, portanto, herdam os dados e funções-membro desta classe. A Figura 7.13 mostra o diagrama de herança dos modelos de equipamentos do simulador.

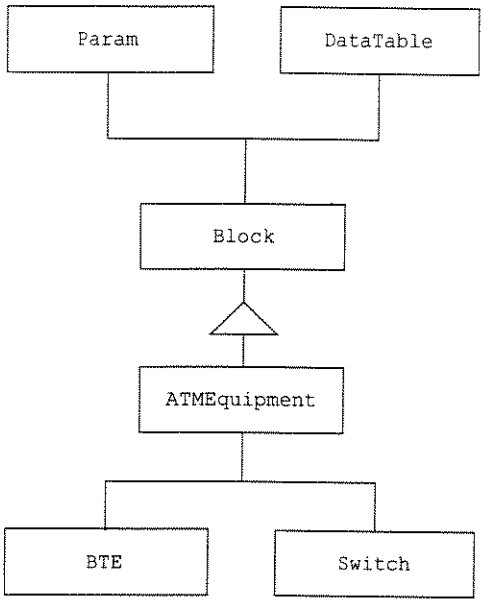


Figura 7.13 – Diagrama de herança dos modelos de equipamentos do SimATM

7.18 - Aplicativos ATM

O elemento **aplicativo ATM** da estrutura do **SimATM** é implementado na classe `ATMApplication`, que é derivada da classe `Block`. Portanto, assim como a classe `ATMEquipment`, a classe `ATMApplication` herda os dados e funções-membro da classe `Block`. A Figura 7.14 mostra um diagrama de herança para a classe `ATMApplication`.

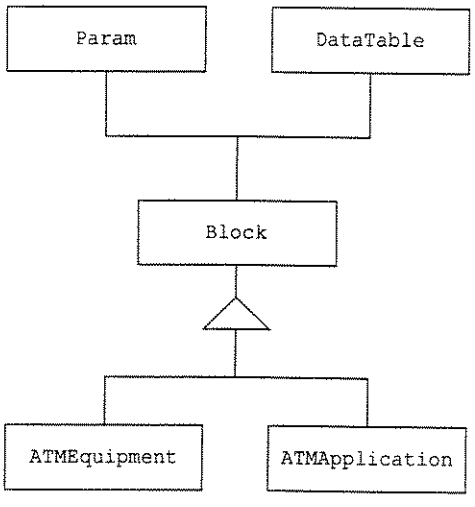


Figura 7.14 – Diagrama de herança da classe `ATMApplication`

Ao contrário da classe `ATMEquipment`, a classe `ATMApplication` possui apenas uma estrutura de dados dicionário: a estrutura `Layers`, utilizada para armazenar as camadas de um aplicativo. Assim com na classe `ATMEquipment`, o dicionário `Layers` possui como chave uma `string` que contém o nome da camada armazenada, e como valor um ponteiro para objetos do tipo `Layer`. As instâncias das camadas de um aplicativo também são alocadas dinamicamente nas classes derivadas da classe `ATMApplication`. Quando uma nova camada é criada, o ponteiro para esse objeto é armazenado na classe dicionário `Layers`. As funções-membro virtuais puras da classe `Block`, cuja implementação é comum a todos os aplicativos da rede ATM são implementadas na classe `ATMApplication`. Estas funções correspondem aos módulos gerenciador de parâmetros, gerenciador de tabelas de dados, gerenciador de camadas e gerenciador de sistemas de fila. Entretanto, as funções virtuais puras relacionadas com os sistemas de fila são implementadas como nulas, uma vez que os aplicativos ATM não possuem sistemas de fila.

As funções-membro virtuais puras cuja implementação difere para cada aplicativo da rede ATM são implementadas nas classes derivadas da classe `ATMApplication`, que constituem os modelos de aplicativos da rede. Estas funções correspondem aos módulos de inicialização, conexão, execução, desconexão e de terminação. Entretanto, as funções-membro virtuais responsáveis pela criação e remoção de camadas, correspondentes ao gerenciador de camadas são implementadas nas classes derivadas da classe `ATMApplication`. A razão para a implementação destas funções nas classes derivadas da classe `ATMApplication`, está na necessidade de criação e remoção de camadas de forma diferenciada em cada modelo de aplicativo ATM da rede.

7.19 - Modelos de Aplicativos ATM

Como já vimos no Capítulo 5, o **SimATM** possui somente um modelo de aplicativo ATM. Este modelo é implementado na classe `Application`, que é derivada da classe `ATMApplication` e, portanto, herda os seus dados e funções-membro. A Figura 7.15 mostra um diagrama de herança para o modelo de aplicativo ATM do simulador.

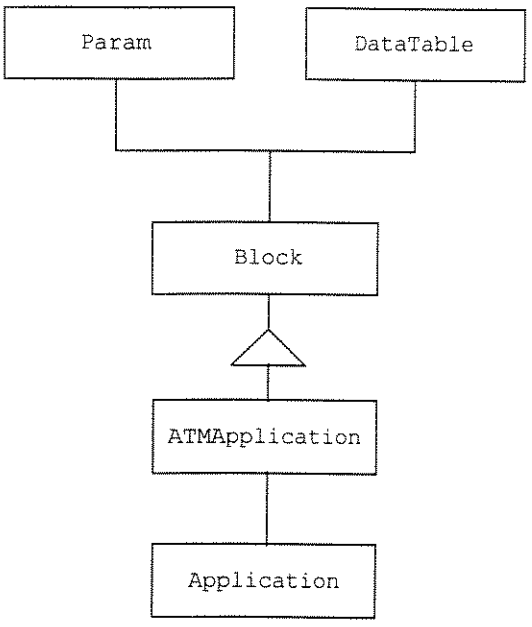


Figura 7.15 – Diagrama de herança do modelo de aplicativo do SimATM

7.20 - Referências Bibliográficas

[1] B. Stroustrup, "*The C++ Programmig Language*", *Addison-Wesley*, 1995.

[2] Tom Swan, "*Mastering Borland* C++", *Sams Publishing*, 1992.

[3] J. Rumbaugh *et al*, "*Object-Oriented Modeling and Design*", *Prentice-Hall*, 1991.

[4] Timothy A. Budd, "*Classic Data Structures in* C++", *Addison-Wesley*, 1994. O código fonte das estruturas de dados descritas neste livro podem ser solicitadas através do e-mail: almanac@cd.orst.edu.

[5] W. Ford, W. Topp, "*Data Structures with* C++", *Prentice-Hall, Inc.*, 1996.

Capítulo 8

Simulando uma Rede ATM

Neste capítulo iremos apresentar, a título de demonstração dos recursos do **SimATM**, um exemplo de simulação de uma rede ATM. Duas simulações foram realizadas: uma para observar o comportamento da rede em regime transitório e outra para observar o comportamento da rede em regime permanente. A configuração da rede simulada, os resultados obtidos nas simulações e o desempenho do simulador serão apresentados.

8.1 - Configuração da Rede ATM Simulada

A Figura 8.1 mostra a rede ATM simulada. O aplicativo App_1 possui uma fonte CBR *Traffic Source*, enquanto o aplicativo App_2 possui uma fonte VBR *Batch Traffic Source*. O aplicativo App_3 também possui uma fonte CBR, entretanto nenhum dado é transmitido a partir deste aplicativo.

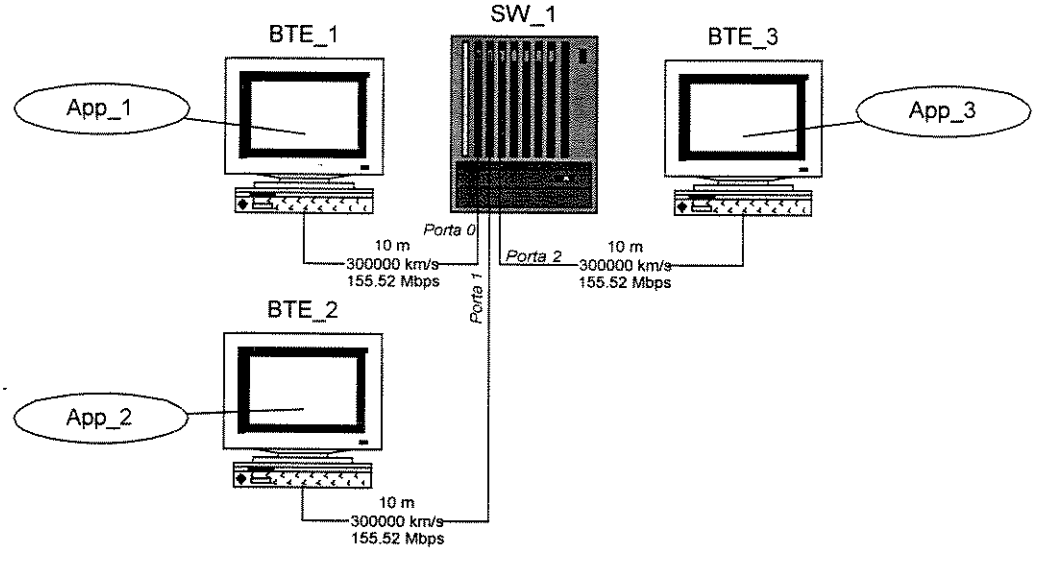


Figura 8.1 – Rede ATM simulada

Duas conexões ATM são estabelecidas: uma a partir do BTE_1 até o BTE_3, chamada de VCC_1, e outra a partir do BTE_2 até o BTE_3, chamada de VCC_2. É feita a transmissão de dados à taxa de *bits* constante (CBR) a partir do aplicativo App_1 até o aplicativo App_3 utilizando uma conexão de dados chamada DC_1, e à taxa de bits variável (VBR) a partir do

aplicativo App_2 até o aplicativo App_3 utilizando outra conexão de dados chamada DC_2. Estas conexões de dados utilizam as duas conexões ATM previamente estabelecidas. O tráfego da conexão de dados CBR tem prioridade sobre o tráfego da conexão de dados VBR. Os BTEs são conectados através de um chaveador ATM chamado SW_1. O aplicativo App_1 transmite a taxa de 18,56 *Mbps* e com pacotes de tamanho 232 *bytes*, o que dá um intervalo entre pacotes de 100 μs. O aplicativo App_2 transmite pacotes de tamanho médio igual a 0,4 células ATM e com intervalo médio entre lotes (*batches*) de 2,7263 μs, o que dá uma taxa aproximada após a AAL 5 de 56,34 *Mbps*.

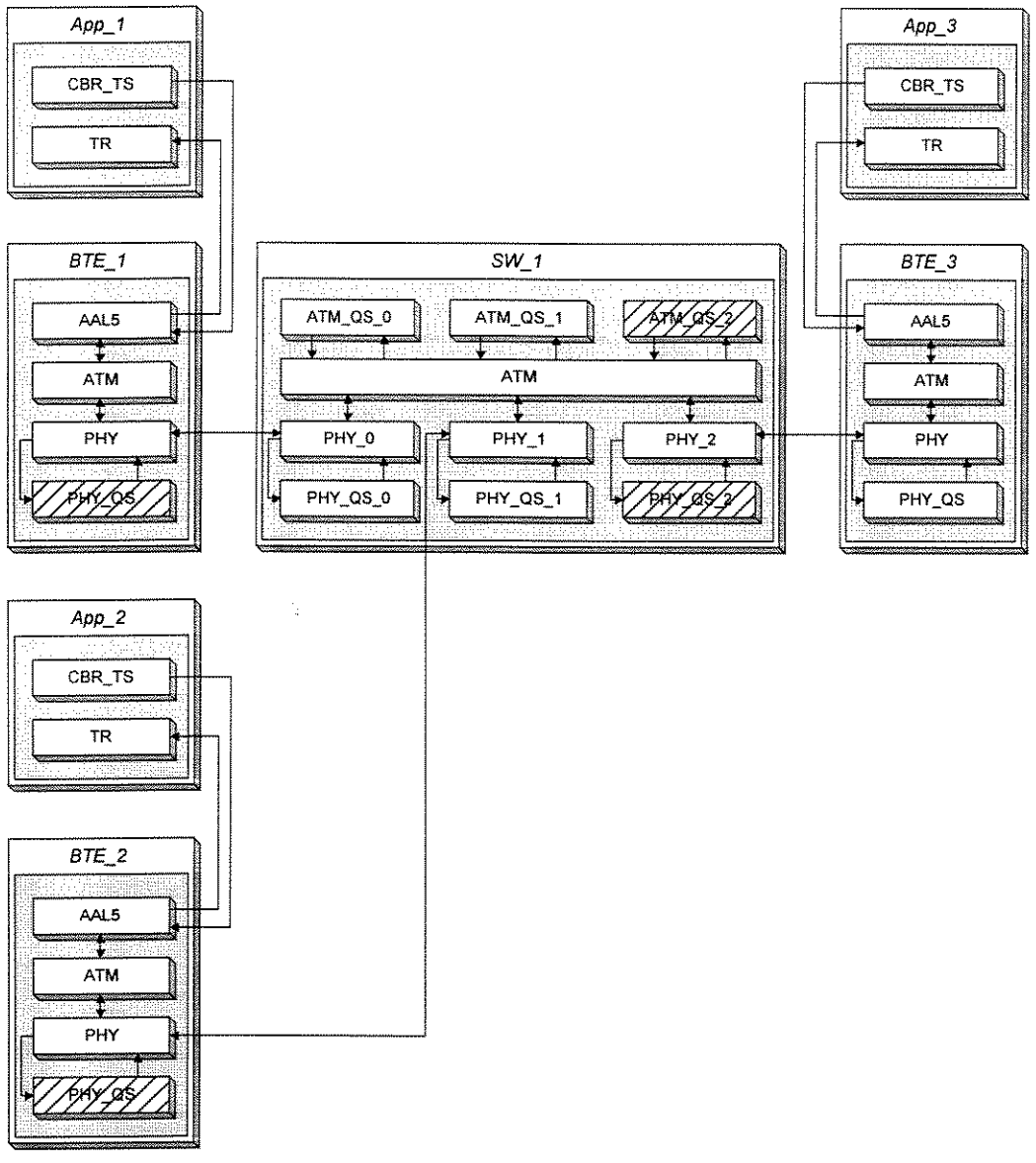


Figura 8.2 – Camadas e sistemas de filas da rede ATM simulada

Todos os *buffers* dos sistemas de fila dos BTEs e do chaveador são de tamanho infinito. A inserção de células de OAM no nível da camada física dos equipamentos foi

desabilitada. O atraso de empacotamento de pacotes na AAL 5 foi configurado para 1 ns. A distância entre os equipamentos ATM da rede é de 10 metros. A velocidade de propagação do sinal na fibra óptica foi configurada para 300000 Km/s, o que resulta em um atraso de propagação de 33,33 ns. A taxa de transmissão em todos os enlace da rede é de 155,52 *Mbps* (OC-3). O tempo de ciclo do chaveador é de 2,7263 μs.

A Figura 8.2 mostra os nomes das camadas e sistemas de fila dos blocos da rede simulada. Os sistemas de fila hachurados armazenam as células ATM das conexões VCC_1 e VCC_2 com destino ao BTE_3.

8.2 - Simulações Realizadas

Duas simulações foram realizadas: uma para observar o comportamento da rede em regime transitório e outra para observar o comportamento da rede em regime permanente. Para realizar essas simulações dois arquivos de configuração de rede foram utilizados.

8.3 - Resultados

A seguir apresentaremos através de gráficos os resultados das simulações realizadas. Na simulação em regime transitório a rede ilustrada na Figura 8.1 foi simulada por 1 ms, enquanto na simulação em regime permanente a rede foi simulada por 5,5 segundos, tempo suficiente para que exatamente 1082175 células transitassem pela rede.

Serão apresentados somente os gráficos das variáveis dos sistemas de fila que armazenam células ATM das conexões VCC_1 e VCC_2. Os outros sistemas da rede permanecem sempre vazios.

Os gráficos a seguir mostram as variáveis de estado instantâneas e médias dos sistemas de fila de interesse (Figura 8.2). A nomenclatura destas variáveis foi apresentada no item 6.11- Sistemas de Fila. A Tabela 8.1 mostra uma síntese desta nomenclatura a fim de facilitar o entendimento das figuras.

Abreviatura	**Significado**	**Tradução**
NoSRC	*Number of System Received Cells*	Número de Células Recebidas no Sistema de Fila
NoBQC	*Number of Buffer Queued Cells*	Número de Células Armazenadas no *Buffer*
NoBDC	*Number of Buffer Dropped Cells*	Número de Células Perdidas no Sistema de Fila
BS	*Buffer Status*	Estado do *Buffer*
NoBC	*Number of Buffer Cells*	Número de Células no *Buffer*
DoBC	*Delay of Buffer Cells*	Tempo de Permanência das Células no *Buffer*

SS	*Server Status*	Estado do Servidor
NoSC	*Number of Server Cells*	Número de Células no Servidor
DoSC	*Delay of Server Cells*	Tempo de Permanência das Células no Servidor
SMNoBC	*Sample Mean NoBC*	Média do Número de Células no *Buffer*
SVNoBC	*Sample Variance NoBC*	Variância do Número de Células no *Buffer*
SMDoBC	*Sample Mean DoBC*	Média do Tempo de Permanência das Células no *Buffer*
SVDoBC	*Sample Variance DoBC*	Variância do Tempo de Permanência das Células no *Buffer*
SMNoSC	*Sample Mean NoSC*	Média do Número de Células no Servidor
SVNoSC	*Sample Variance NoSC*	Variância do Número de Células no Servidor

Tabela 8.1 – Variáveis de estado instantâneas e médias dos sistemas de fila

8.3.1 - Regime Transitório

8.3.1.1 - Aplicativo App_1

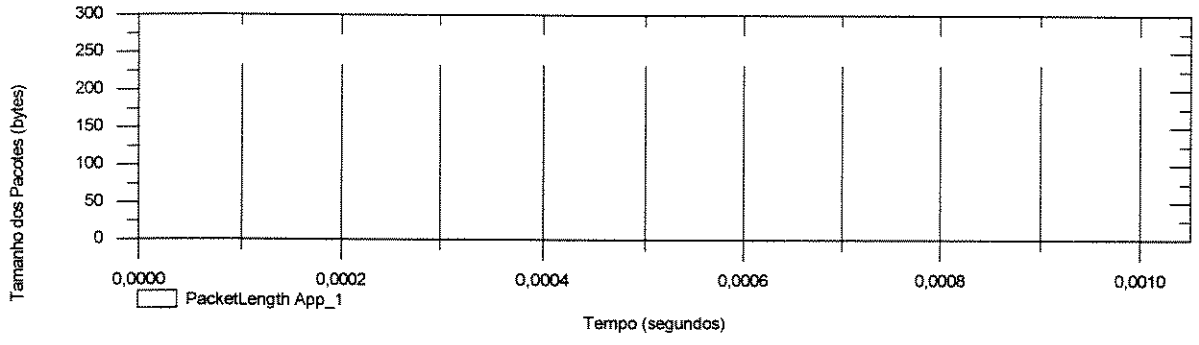


Figura 8.3 – Padrão de tráfego gerado pelo aplicativo App_1

A Figura 8.3 mostra o padrão de tráfego gerado pelo aplicativo App_1.

8.3.1.2 - Terminal BTE_1

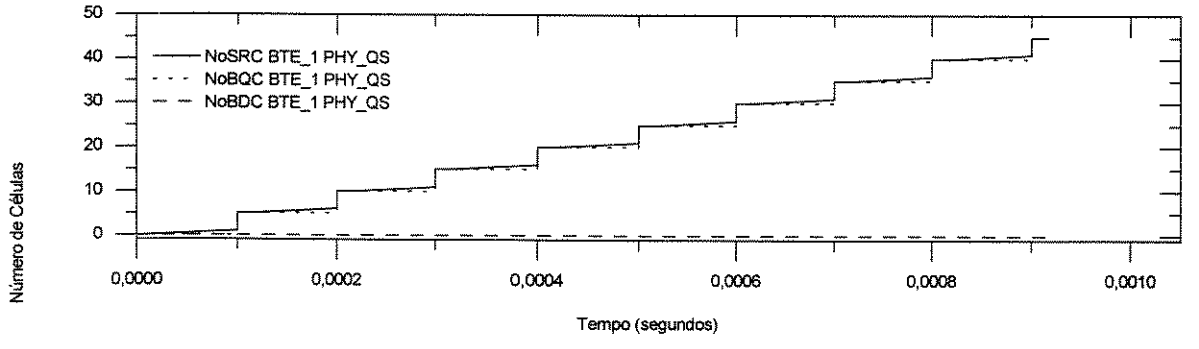


Figura 8.4 – Número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS do BTE_1

A Figura 8.4 mostra o número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS do BTE_1 após os pacotes terem sido recebidos do aplicativo App_1 e fragmentados em células ATM. A Figura 8.5 mostra o estado do *buffer* deste mesmo sistema de fila. Quando BS = 1 significa que existem células no *buffer*.

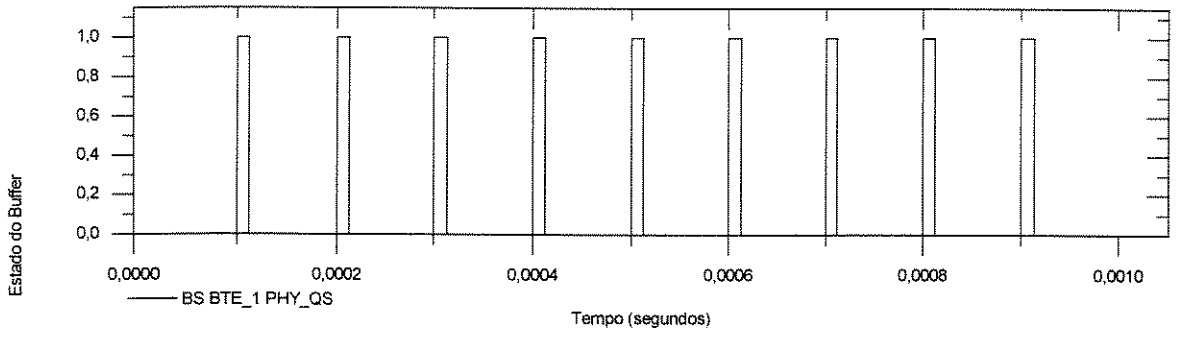


Figura 8.5 – Estado do *buffer* do sistema de fila PHY_QS do BTE_1

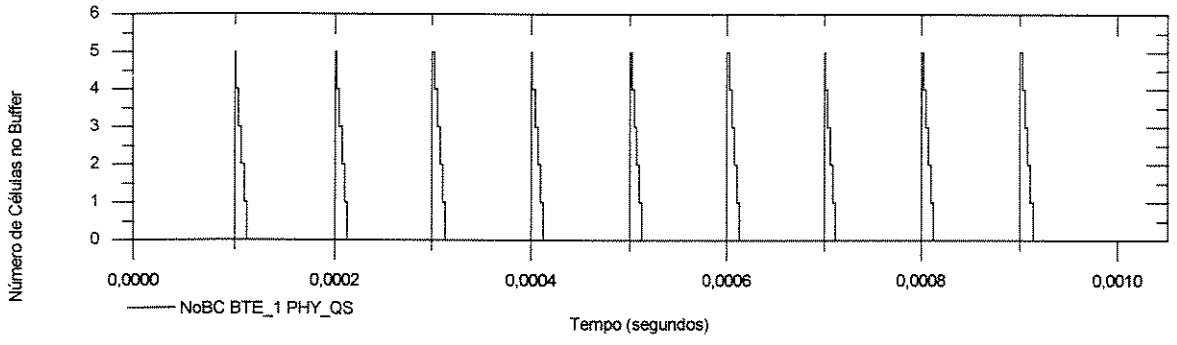


Figura 8.6 – Número de células no *buffer* do sistema de fila PHY_QS do BTE_1

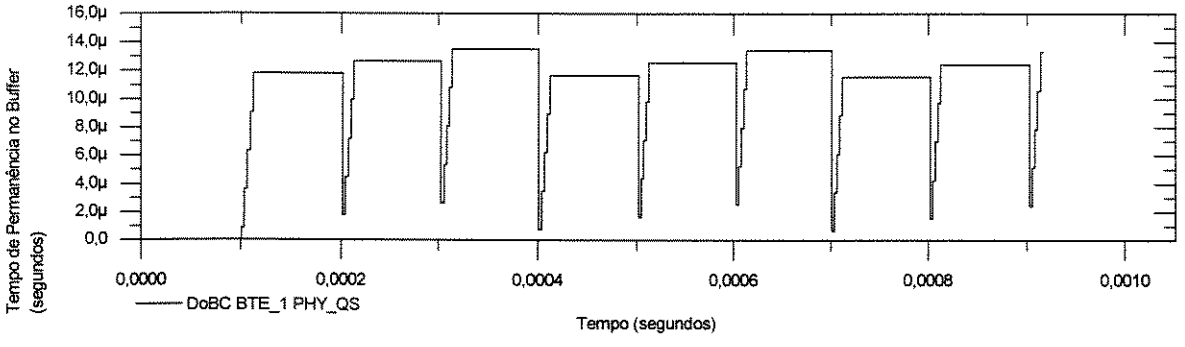


Figura 8.7 – Tempo de permanência das células ATM no *buffer* do sistema de fila PHY_QS do BTE_1

A Figura 8.6 mostra o número de células no sistema de fila PHY_QS do BTE_1. Pode-se observar que todas as células são retiradas do *buffer* antes que um novo pacote seja recebido. A Figura 8.7 mostra o tempo de permanência das células ATM neste mesmo *buffer*. Somente a primeira célula de cada pacote aguarda por serviço menos que um período de *frame*. Devido a diferença de sincronismo entre a taxa da fonte e a taxa de serviço, o tempo de

permanência das células no *buffer* apresenta um comportamento cíclico, que se repete a cada três pacotes. A Figura 8.8 mostra o estado do servidor do sistema de fila do BTE_1. A Figura 8.9 mostra o tempo de permanência das células neste servidor, que é igual ao tamanho de um *frame* (2,7263 μs).

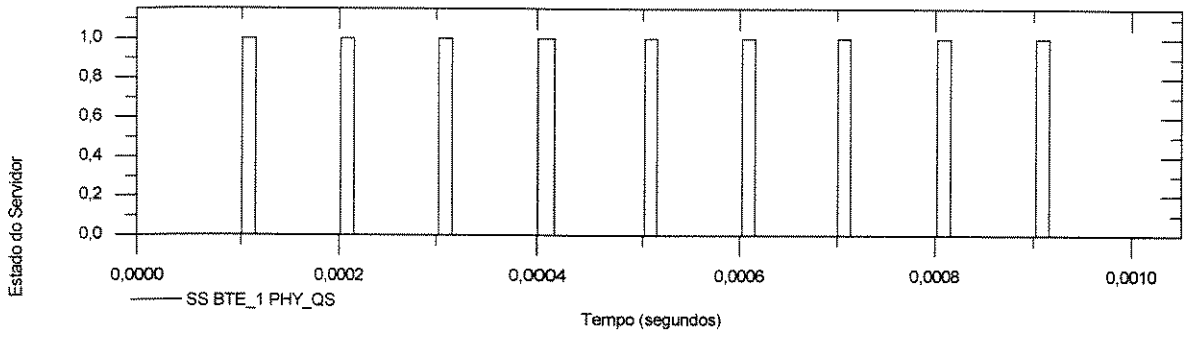


Figura 8.8 – Estado do servidor do sistema de fila PHY_QS do BTE_1

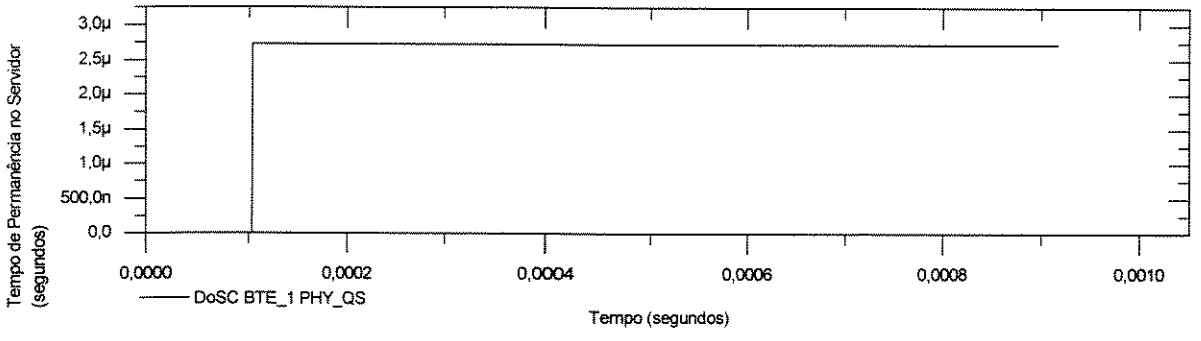


Figura 8.9 – Tempo de permanência das células no servidor do sistema de fila PHY_QS do BTE_1

8.3.1.3 - Aplicativo App_2

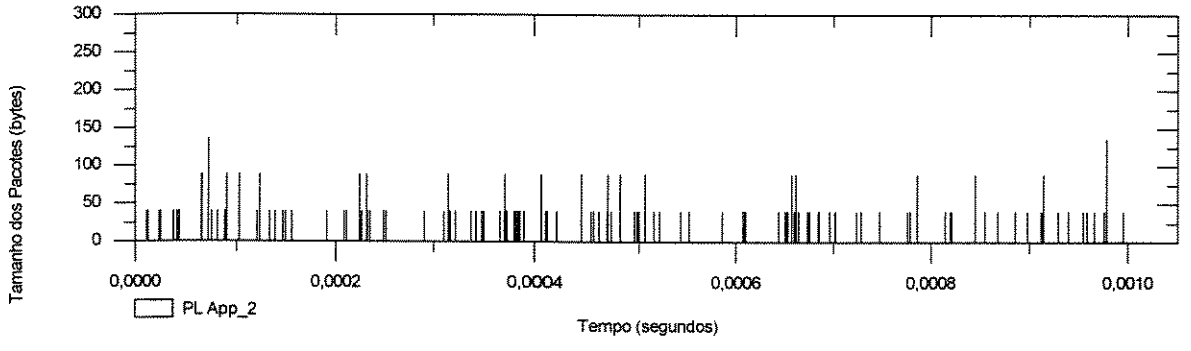


Figura 8.10 – Padrão de tráfego gerado pelo aplicativo App_2

A Figura 8.10 mostra o padrão de tráfego gerado pelo aplicativo App_2.

8.3.1.4 - Terminal BTE_2

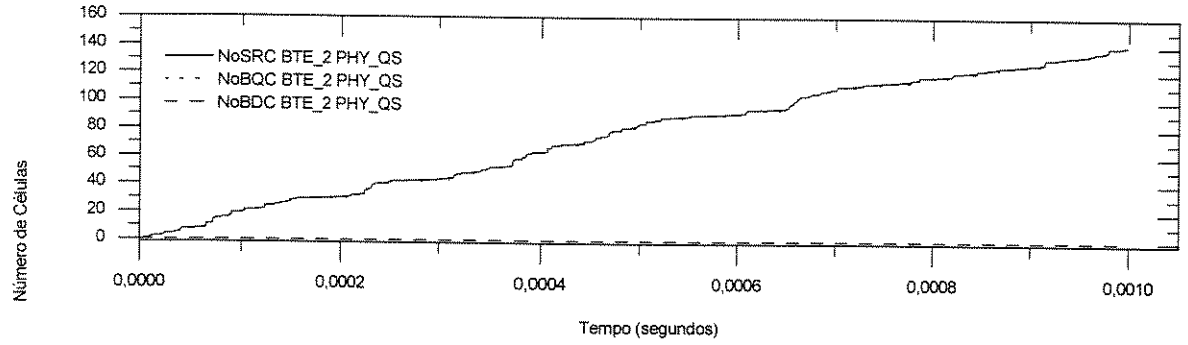


Figura 8.11 – Número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS do BTE_2

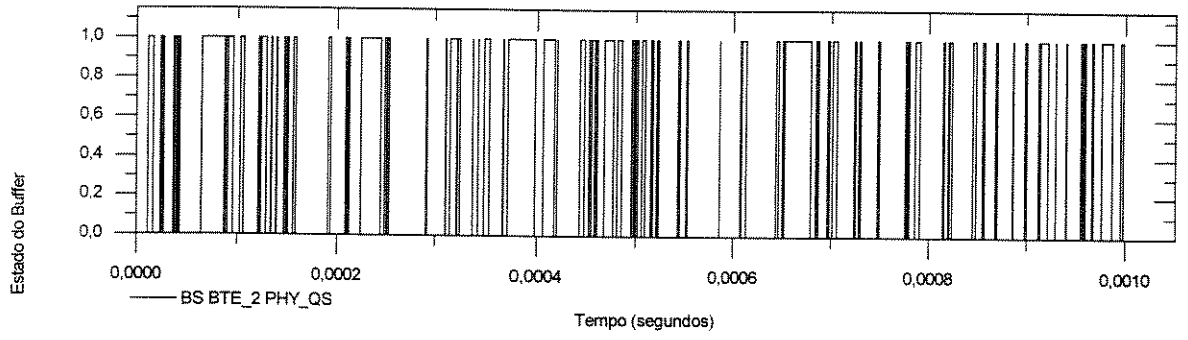


Figura 8.12 – Estado do *buffer* do sistema de fila PHY_QS do BTE_2

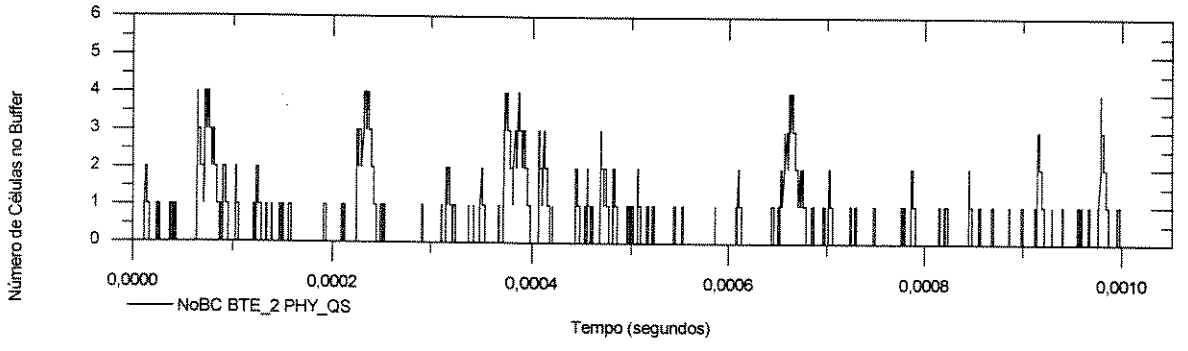


Figura 8.13 – Número de células no *buffer* do sistema de fila PHY_QS do BTE_2

A Figura 8.11 mostra o número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS do BTE_2 após os pacotes terem sido recebidos do aplicativo App_2 e fragmentados em células ATM. A Figura 8.12 mostra o estado do *buffer* deste mesmo sistema de fila. O número de células na fila deste sistema é mostrado na Figura 8.13. É possível observar que o número de células no *buffer* não ultrapassa 6 células, ou seja, a taxa do enlace que liga o BTE_2 ao SW_1 é suficiente para atender o tráfego gerado pelo aplicativo App_2. A Figura 8.14 mostra o tempo de permanência das células no mesmo

buffer. O atraso máximo fica em torno de 25 μs. A Figura 8.15 mostra o estado do servidor do sistema de fila do BTE_2. A média do estado do servidor fornece a utilização deste servidor. A Figura 8.16 mostra o tempo de permanência das células no servidor, que também é igual a 2,7263 μs.

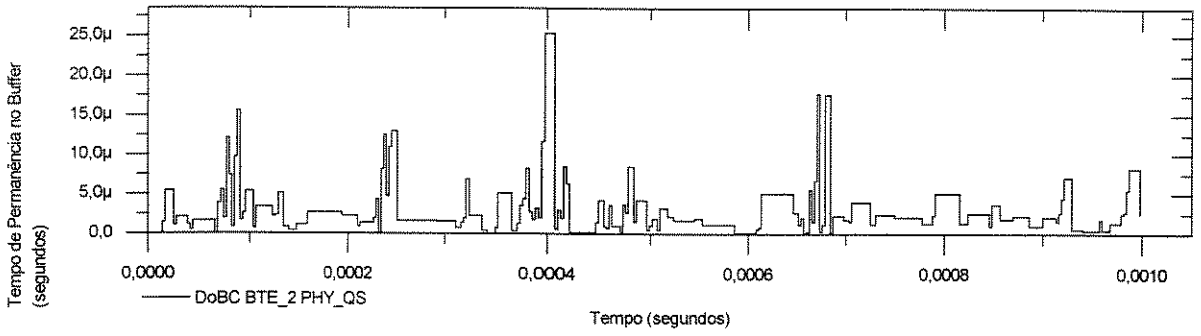


Figura 8.14 – Tempo de permanência das células ATM no *buffer* do sistema de fila PHY_QS do BTE_2

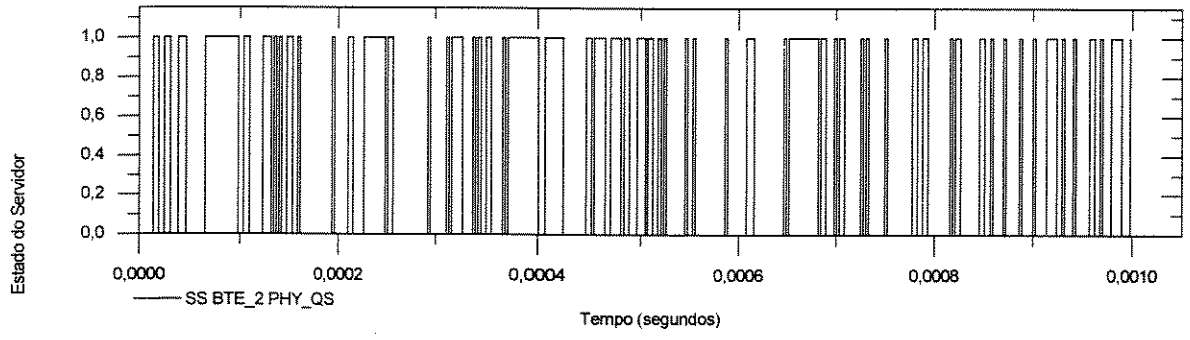


Figura 8.15 – Estado do servidor do sistema de fila PHY_QS do BTE_2

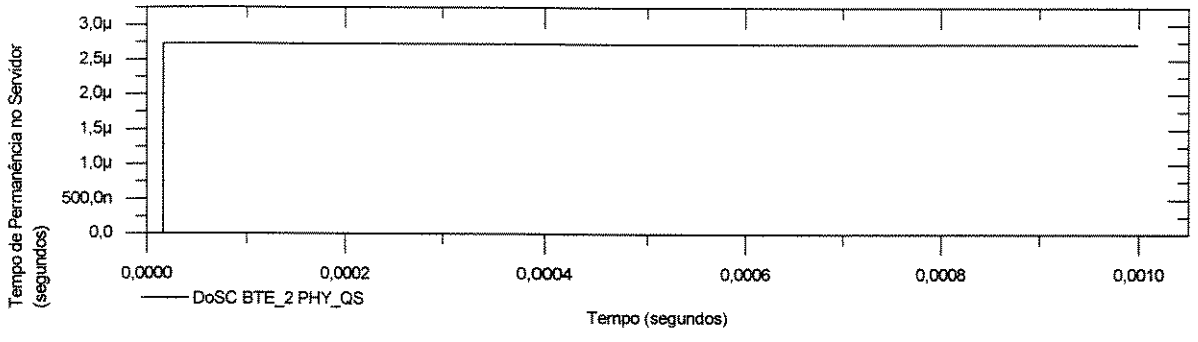


Figura 8.16 – Tempo de permanência das células no servidor do sistema de fila PHY_QS do BTE_2

8.3.1.5 - Chaveador SW_1

8.3.1.5.1 - Camada ATM

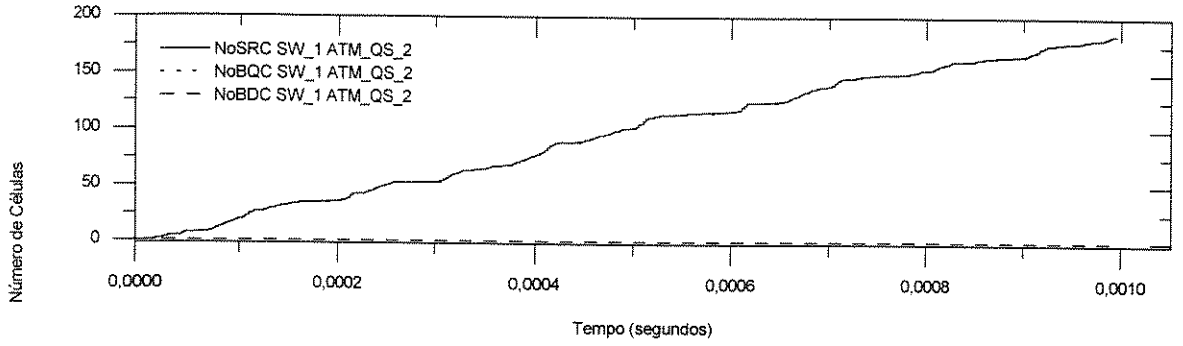


Figura 8.17 – Número de células recebidas, armazenadas e perdidas no sistema de fila ATM_QS_2 do SW_1

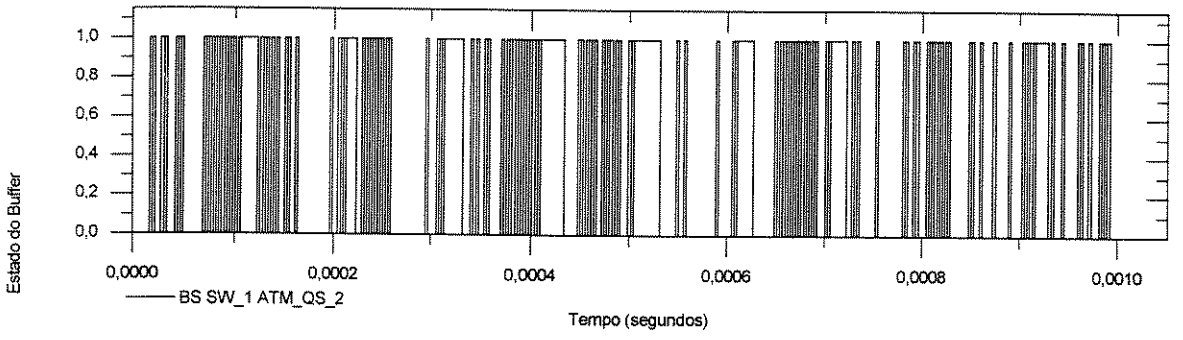


Figura 8.18 – Estado do *buffer* do sistema de fila ATM_QS_2 do SW_1

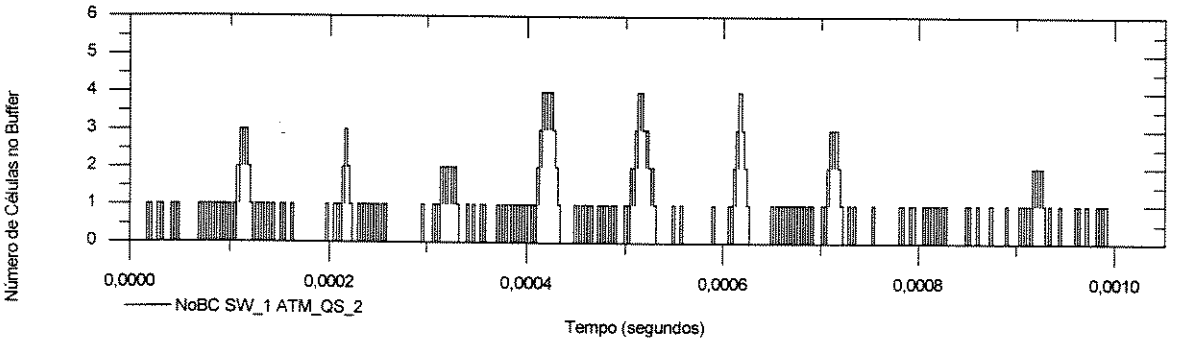


Figura 8.19 – Número de células no *buffer* do sistema de fila ATM_QS_2 do SW_1

A Figura 8.17 mostra o número de células recebidas, armazenadas e perdidas no sistema de fila ATM_QS_2 do SW_1. Este sistema de fila concentra os tráfegos provenientes do BTE_1 e do BTE_2, que tem como destino final o BTE_3. A Figura 8.18 mostra o estado do *buffer* deste sistema. O número de células armazenadas na fila é mostrado na Figura 8.19, o tempo de permanência das células na fila é mostrada na Figura 8.20, o estado do servidor é mostrado na Figura 8.21 e o tempo de permanência das células no servidor é mostrado na Figura 8.22. Pode-se observar que os picos no número de células no *buffer* e no tempo de

permanência das células no *buffer* ocorrem nos instantes de tempo em que os tráfegos gerados pelos dois aplicativos se somam, ou seja, nos instantes múltiplos de 0,1 ms.

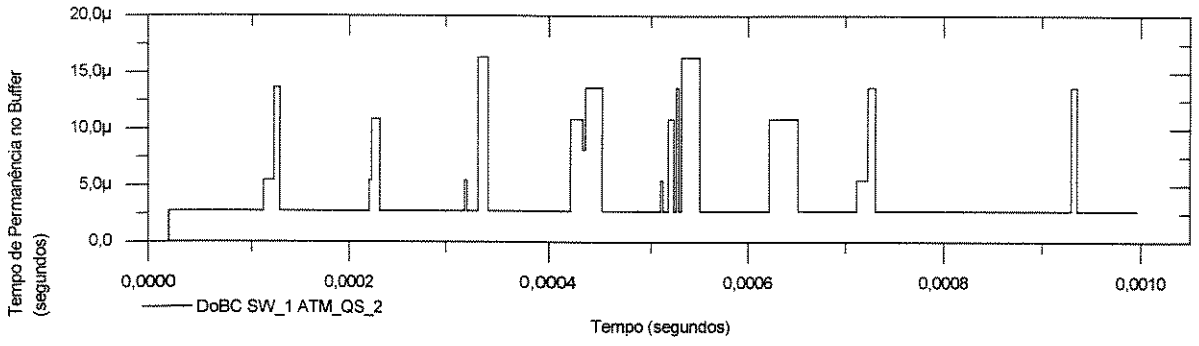


Figura 8.20 – Tempo de permanência das células ATM no *buffer* do sistema de fila ATM_QS_2 do SW_1

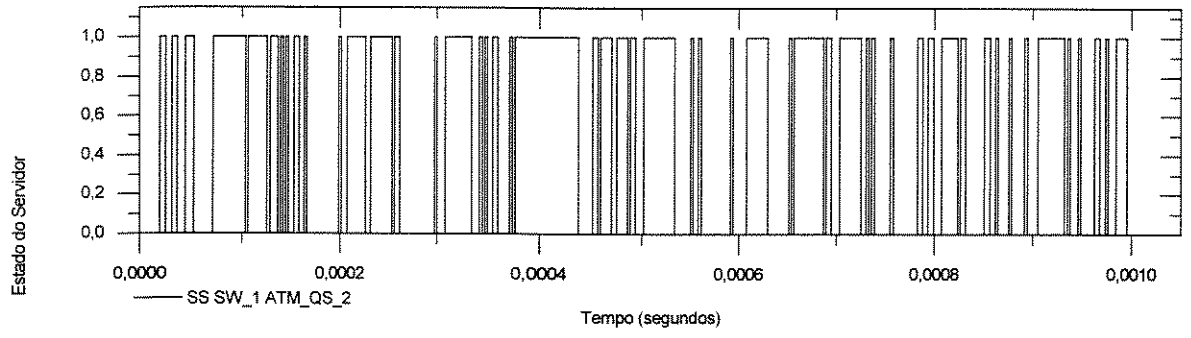


Figura 8.21 – Estado do servidor do sistema de fila ATM_QS_2 do SW_1

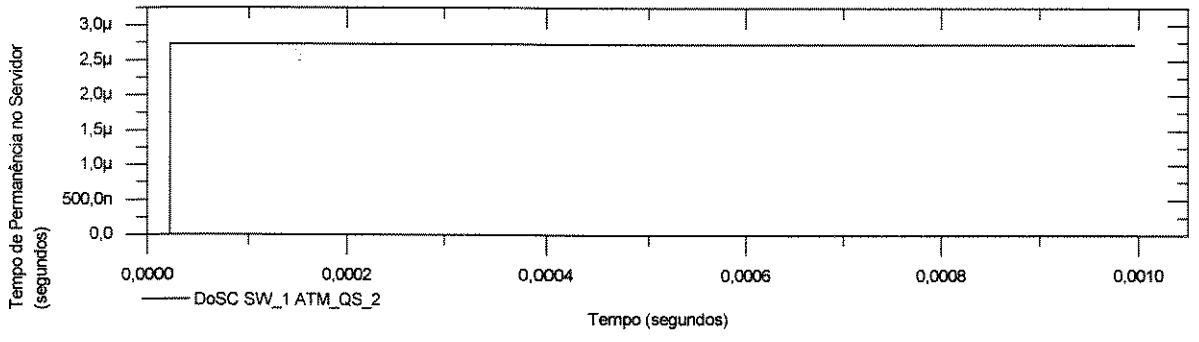


Figura 8.22 – Tempo de permanência das células no servidor do sistema de fila ATM_QS_2 do SW_1

8.3.1.5.2 - Camada Física

As figuras: Figura 8.23, Figura 8.24, Figura 8.25, Figura 8.26, Figura 8.27 e Figura 8.28 mostram as variáveis de estado instantâneas do sistema de fila PHY_QS_2 do chaveador SW_1. O número de células no *buffer* deste sistema de fila não ultrapassa 1 célula porque a multiplexação das células das conexões ATM VCC_1 e VCC_2 ocorre na camada ATM do chaveador SW_1.

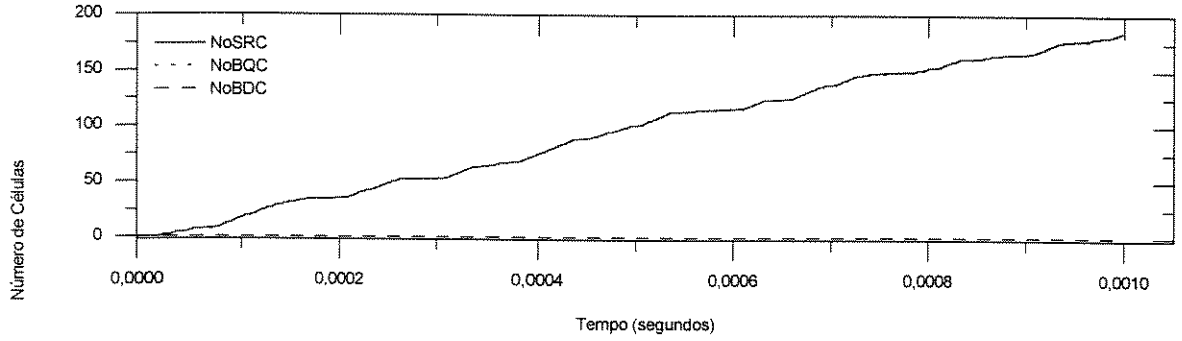


Figura 8.23 – Número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS_2 do SW_1

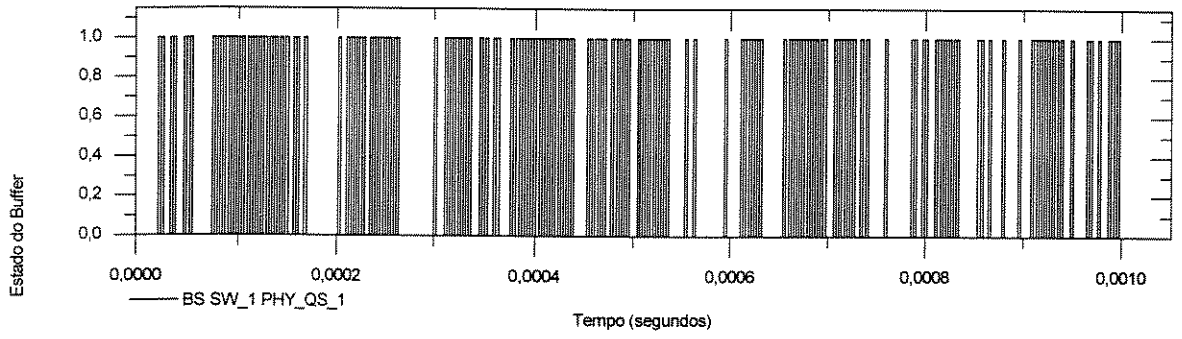


Figura 8.24 – Estado do *buffer* do sistema de fila PHY_QS_2 do SW_1

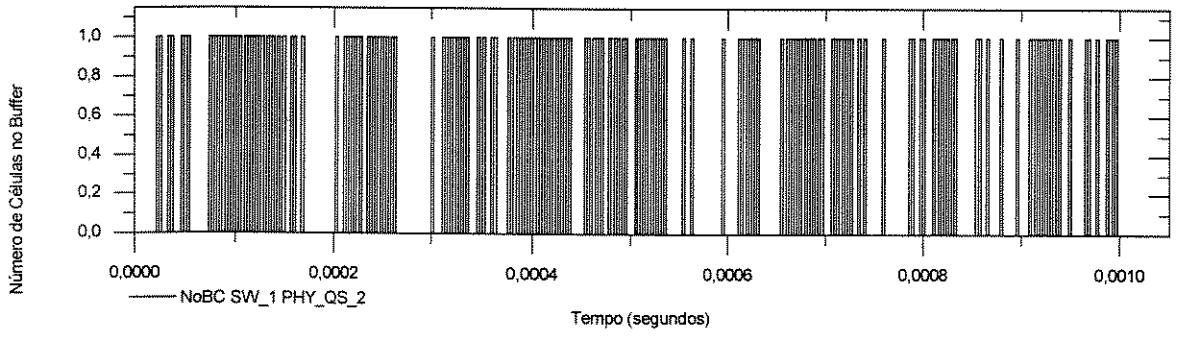


Figura 8.25 – Número de células no *buffer* do sistema de fila PHY_QS_2 do SW_1

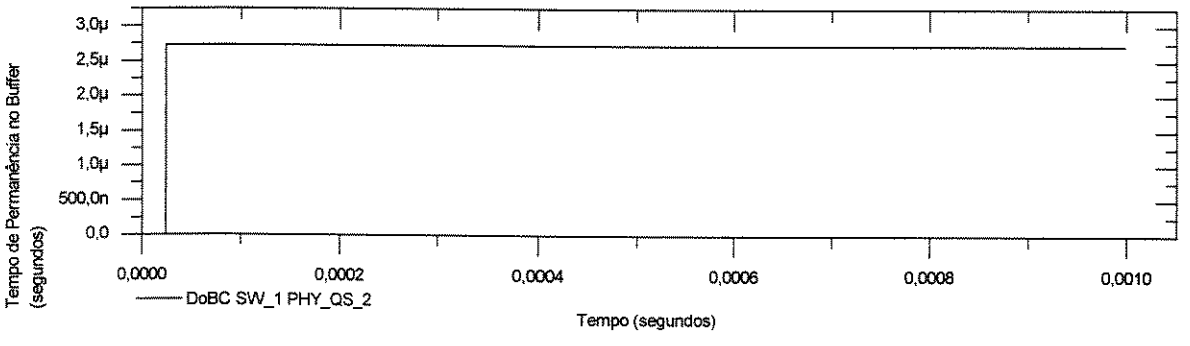


Figura 8.26 – Tempo de permanência das células ATM no *buffer* do sistema de fila PHY_QS_2 do SW_1

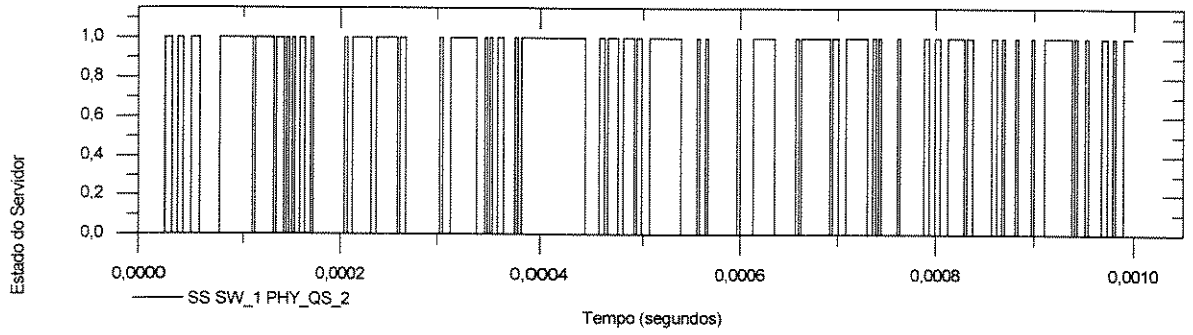


Figura 8.27 – Estado do servidor do sistema de fila PHY_QS_2 do SW_1

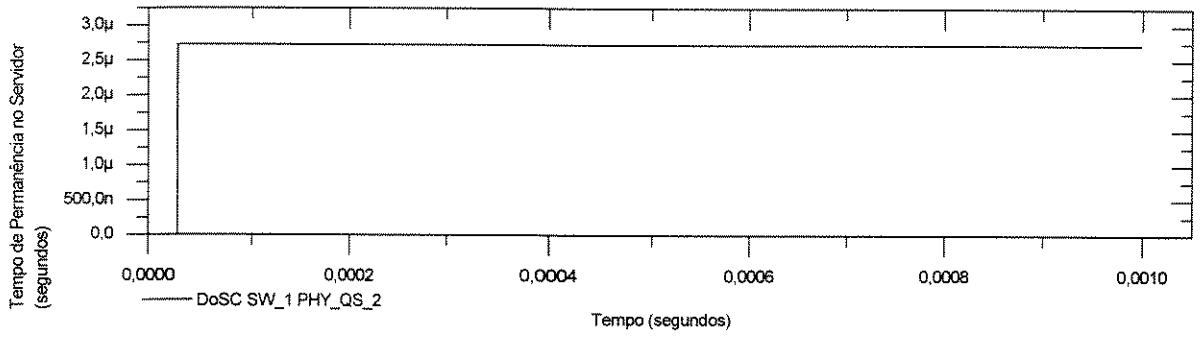


Figura 8.28 – Tempo de permanência das células no servidor do sistema de fila PHY_QS_2 do SW_1

8.3.2 - Regime Permanente

As figuras: Figura 8.29, Figura 8.30 e Figura 8.31 mostram as variáveis instantâneas NoSRC, NoBQC e NoBDC, amostradas por tempo a cada 1 ms, até o final da simulação. No sistema de fila PHY_QS do BTE_1 foram recebidas 274995 células, o que dá uma taxa aproximada de 49999,09 células/segundo. No sistema de fila PHY_QS do BTE_2 foram recebidas 807230 células, o que dá uma taxa de 146769,09 células/segundo. O número total de células que trafegaram pela rede foi de 1082175 células. Nenhuma célula foi perdida.

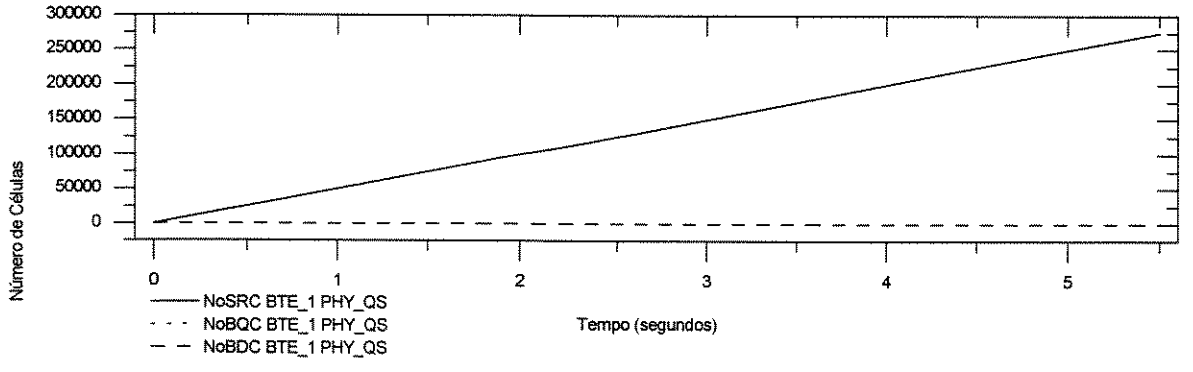


Figura 8.29 – Número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS do BTE_1

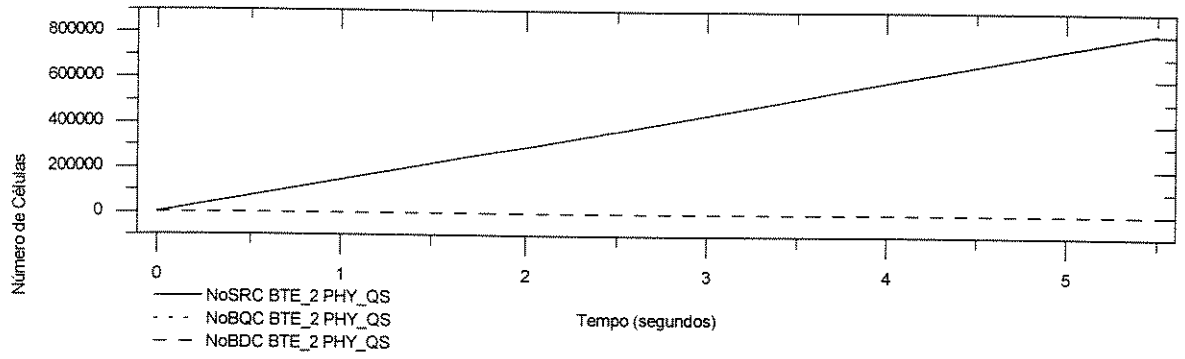


Figura 8.30 – Número de células recebidas, armazenadas e perdidas no sistema de fila PHY_QS do BTE_2

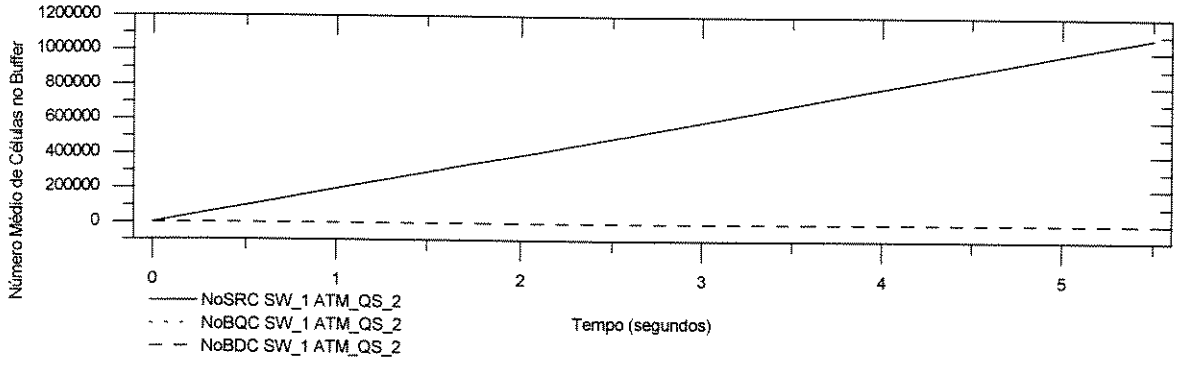


Figura 8.31 – Número de células recebidas, armazenadas e perdidas no sistema de fila ATM_QS_2 do SW_1

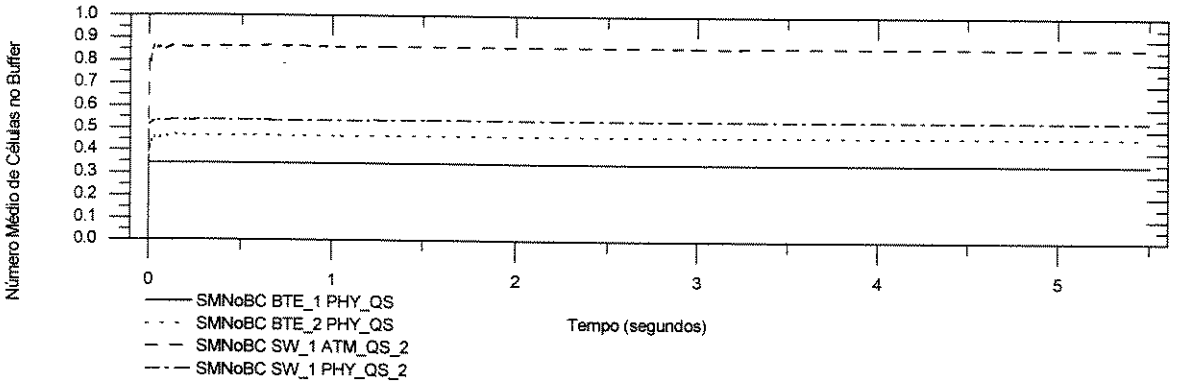


Figura 8.32 – Número médio de células ATM nos *buffers* dos sistemas de fila de interesse

A Figura 8.32 mostra o número médio de células ATM nos *buffers* dos sistemas de fila de interesse. No *buffer* do sistema de fila PHY_QS do BTE_1 o número médio de células foi de aproximadamente 0,34192 células. No *buffer* do sistema de fila PHY_QS do BTE_2 o número médio de células foi de aproximadamente 0,46555 células. No *buffer* do sistema de fila ATM_QS_2 do SW_1 o número médio de células foi de aproximadamente 0,86171

células. Finalmente, no *buffer* do sistema de fila PHY_QS_2 do SW_1 o número médio de células foi de aproximadamente 0,53652 células.

A Figura 8.33 mostra o tempo médio de permanência das células ATM nos *buffers* dos sistemas de fila de interesse. No *buffer* do sistema de fila PHY_QS do BTE_1 o tempo médio de permanência das células ATM foi de aproximadamente 6,83856 μs. No *buffer* do sistema de fila PHY_QS do BTE_2 o tempo médio de permanência das células ATM foi de aproximadamente 3,17146 μs. No *buffer* do sistema de fila ATM_QS_2 do SW_1 o tempo médio de permanência das células ATM foi de aproximadamente 4,37879 μs. Finalmente, no *buffer* do sistema de fila PHY_QS_2 do SW_1 o tempo médio de permanência das células ATM foi de aproximadamente 2,726337116 μs.

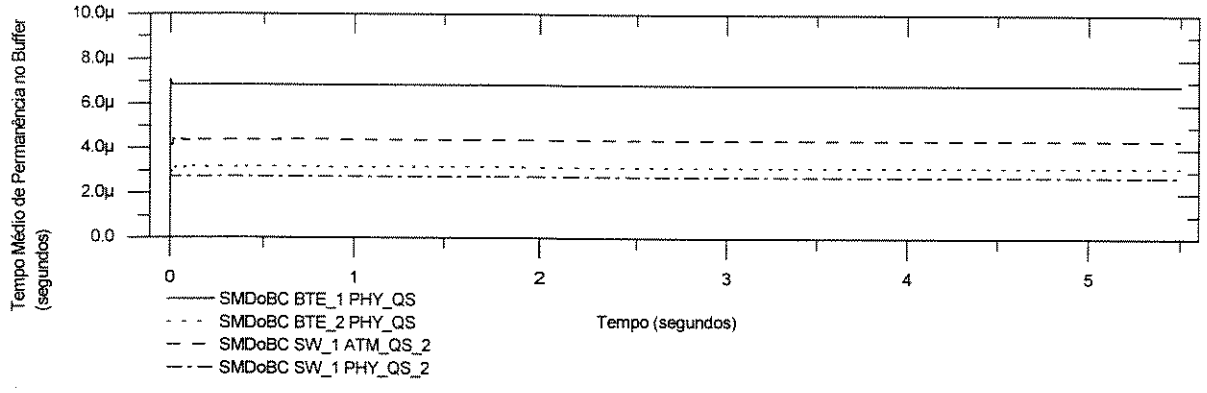


Figura 8.33 – Tempo de permanência das células ATM nos *buffers* dos sistemas de fila de interesse

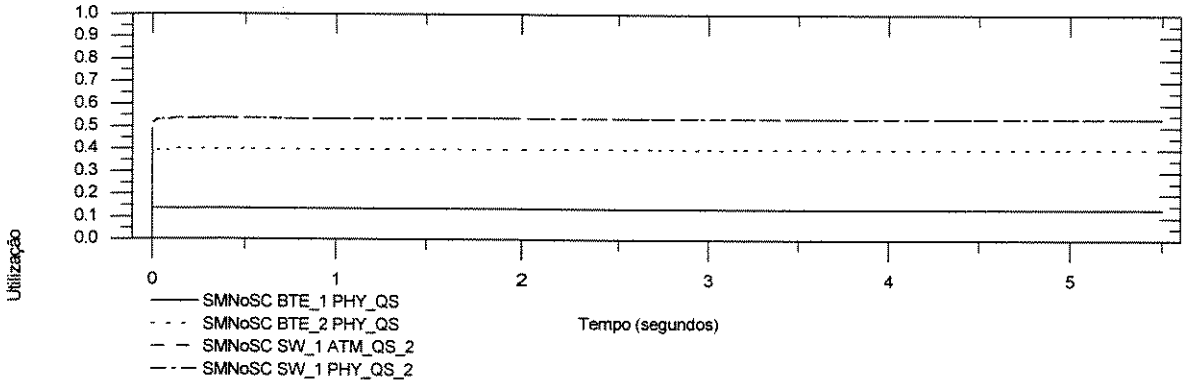


Figura 8.34 – Utilização dos sistemas de fila de interesse

A Figura 8.34 mostra a utilização média (definiremos utilização no próximo item) dos sistemas de fila de interesse. No sistema de fila PHY_QS do BTE_1 a utilização foi de aproximadamente 0,1363164. No sistema de fila PHY_QS do BTE_2 a utilização foi de aproximadamente 0,4002. No sistema de fila ATM_QS_2 do SW_1 a utilização foi de

aproximadamente 0,53652. Finalmente, no sistema de fila PHY_QS_2 do SW_1 a utilização foi de aproximadamente 0,53652 células.

A Figura 8.35 mostra o tempo médio de permanência das células ATM no servidor dos sistemas de fila de interesse. Em todos os servidores o tempo médio de permanência foi de 2,726337 μs.

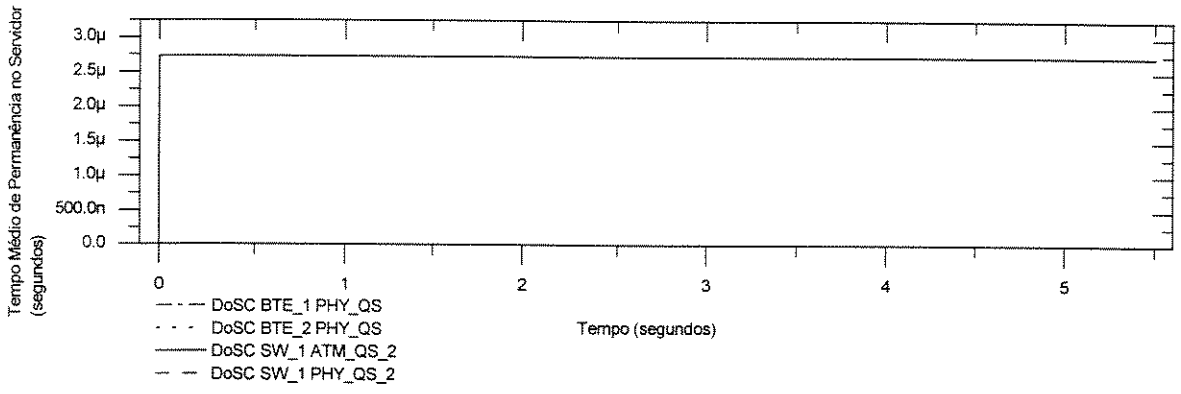


Figura 8.35 – Tempo de permanência das células ATM nos servidores dos sistemas de fila da rede

A Tabela 8.2 sumariza os resultados obtidos na simulação em regime permanente.

Bloco	Sistema de Fila	Variável	Valor
BTE_1	PHY_QS	SMNoBC	0,34192 células
		SMDoBC	6,83856 μs
		SMNoSC	0,136316 células
		DoSC	2,726337 μs
BTE_2	PHY_QS	SMNoBC	0,46555 células
		SMDoBC	3,1714 μs
		SMNoSC	0,4002 células
		DoSC	2,726337 μs
SW_1	ATM_QS_2	SMNoBC	0,8617 células
		SMDoBC	4,3787 μs
		SMNoSC	0,53652 células
		DoSC	2,726337 μs
	PHY_QS_2	SMNoBC	0,53652 células
		SMDoBC	2,726337116 μs
		SMNoSC	0,53652 células
		DoSC	2,726337 μs

Tabela 8.2 – Sumário das variáveis de estado médias

Bloco	Sistema de Fila	Equação Teórica	Equação Equivalente (1) = (2)	Valor de (1)	Valor de (2)	Erro
BTE_1	PHY_QS	$\overline{Nq} = \lambda.W$	SMNoBC = λ.SMDoBC	0,34192 células	λ * 6,83856 μs = 0,341928 células	0,000001776 células
		$\overline{Ns} = \lambda.\overline{x}$ ou $\rho = \lambda.\overline{x}$	SMNoSC = λ.SMDoSC	0,136316 células	λ * 2,726337 μs = 0,136314369 células	0,000000163 células
		$T = \overline{x} + W$	SMDoQSC[1] = SMDoBC + SMDoSC	-	9,564897 μs	-
		$\overline{N} = \overline{Nq} + \overline{Ns}$	SMNoQSC = SMNoBC + SMNoSC	-	0,478236 células	-
		$\overline{N} = \lambda.T$	SMNoQSC = λ.SMDoQSC	0,478236 células	λ * 9,564897 μs = 0,478236159 células	0,00000014 células
BTE_2	PHY_QS	$\overline{Nq} = \lambda.W$	SMNoBC = λ.SMDoBC	0,46555 células	λ * 3,1714 μs = 0, 465463492 células	0,000086 células
		$\overline{Ns} = \lambda.\overline{x}$ ou $\rho = \lambda.\overline{x}$	SMNoSC = λ.SMDoSC	0,4002 células	λ * 2,726337 μs = 0,400142 células	0,000057 células
		$T = \overline{x} + W$	SMDoQSC = SMDoBC + SMDoSC	-	5,897737 μs	-
		$\overline{N} = \overline{Nq} + \overline{Ns}$	SMNoQSC = SMNoBC + SMNoSC	-	0,86575 células	-
		$\overline{N} = \lambda.T$	SMNoQSC = λ.SMDoQSC	0,86575 células	λ * 5,897737 μs = 0,8656054 células	0,000144 células
SW_1	ATM_QS_2	$\overline{Nq} = \lambda.W$	SMNoBC = λ.SMDoBC	0,8617 células	λ * 4,3787 μs = 0,861592814 células	0,000107 células
		$\overline{Ns} = \lambda.\overline{x}$ ou $\rho = \lambda.\overline{x}$	SMNoSC = λ.SMDoSC	0,53652 células	λ * 2,726337 μs = 0,536458850 células	0,000061 células
		$T = \overline{x} + W$	SMDoQSC = SMDoBC + SMDoSC	-	7,105037 μs	-
		$\overline{N} = \overline{Nq} + \overline{Ns}$	SMNoQSC = SMNoBC + SMNoSC	-	0,86575 células	-
		$\overline{N} = \lambda.T$	SMNoQSC = λ.SMDoQSC	1,39822 células	λ * 7,105037 μs = 1,3980516 células	0,000168 células
	PHY_QS_2	$\overline{Nq} = \lambda.W$	SMNoBC = λ.SMDoBC	0,53652 células	λ * 2,72633711 μs = 0,53645887 células	0,000061 células
		$\overline{Ns} = \lambda.\overline{x}$ ou $\rho = \lambda.\overline{x}$	SMNoSC = λ.SMDoSC	0,53652 células	λ * 2,726337 μs = 0, 53645885 células	0,000061 células
		$T = \overline{x} + W$	SMDoQSC = SMDoBC + SMDoSC	-	5,452674 μs	-
		$\overline{N} = \overline{Nq} + \overline{Ns}$	SMNoQSC = SMNoBC + SMNoSC	-	1,073040 células	-
		$\overline{N} = \lambda.T$	SMNoQSC = λ.SMDoQSC	1,07304 células	λ * 5,452674 μs = 1,072917 células	0,000122 células

Tabela 8.3 – Verificação das equações apresentadas anteriormente

[1] SMDoQSC – *Sample Mean Delay of Queueing System Cells* – Esta variável não existe no SimATM. Trata-se apenas da soma das variáveis SMDoBC e SMDoSC. O mesmo é feito para a variável SMNoQSC – *Sample Mean Number of Queueing System Cells.*

8.5 - Desempenho do Simulador

O desempenho do **SimATM** foi avaliado durante a simulação em regime permanente. O microcomputador utilizado possui um processador Pentium Intel™ de 133 MHz, com 64 Mbytes de memória RAM e com sistema operacional Windows 95™. O número de eventos executados na simulação em regime permanente foi de 16.431.035 eventos. O tempo total de simulação foi de 2 horas, 28 minutos e 27 segundos, o que dá uma média de 542,172 μs por eventos. Durante a simulação foram feitas amostras para arquivo do tempo necessário para executar um lote de 10000 eventos. A Figura 8.36 mostra um histograma e um gráfico de probabilidade para estas amostras.

No gráfico inferior da Figura 8.36 é mostrada uma contagem do número de lotes de 10000 eventos que possuem um determinado valor de tempo de processamento. No gráfico superior da Figura 8.36 é mostrada uma contagem cumulativa dos lotes de eventos que possuem um determinado valor de tempo de processamento. A partir da Figura 8.36 pode-se observar que a maioria das amostras (98%) se concentram na proximidade do tempo de processamento do lote de eventos igual a 5,3795 segundos.

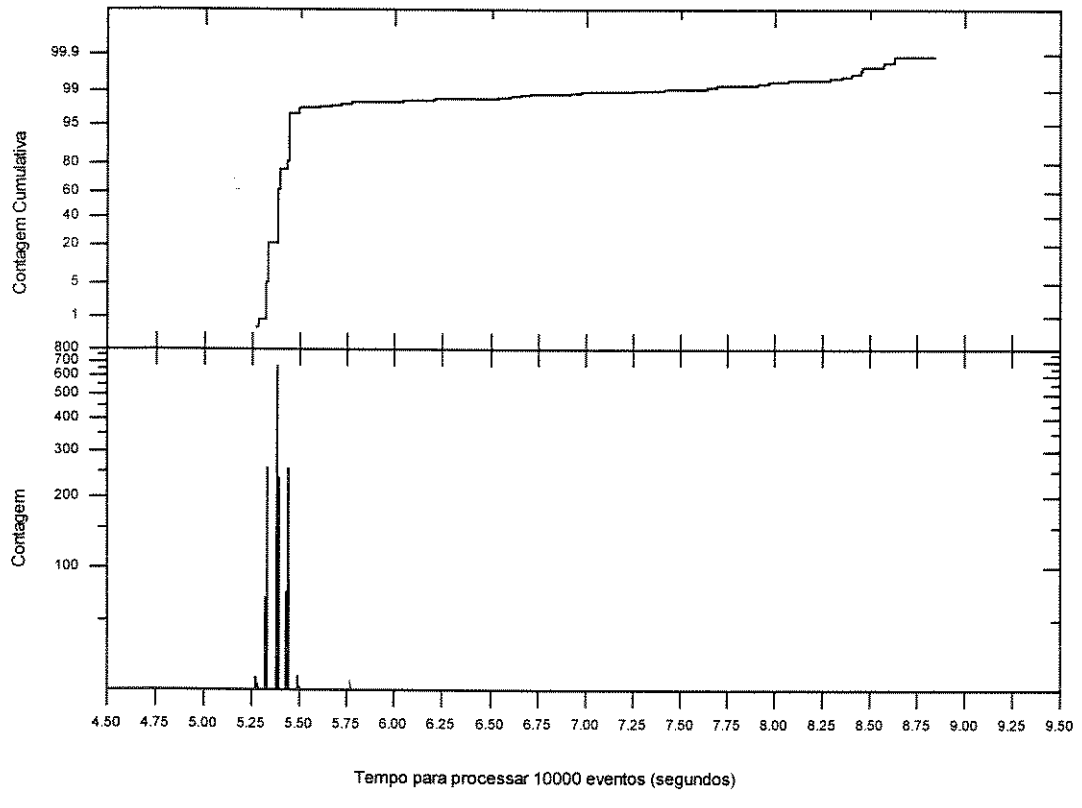


Figura 8.36 – Desempenho do SimATM

Aproximando-se o histograma por uma gaussiana obtemos uma média para o tempo de processamento do lote de eventos igual a 5,37945 segundos e desvio padrão igual a 0,000224502 segundos. Ou seja, a média do tempo de execução de um evento é igual a aproximadamente 537,945 μs.

8.6 - Referências Bibliográficas

[1] Leonard Kleinrock, "*Queueing Systems – Volume I: Theory*", *John Wiley & Sons*, 1975.

[2] Thomas G. Robertazzi, "*Computer Networks and Systems – Queueing Theory and Perfomance Evaluation*", *Second Edition*, *Springer-Verlag*, 1994.

Capítulo 9

Conclusões

Este trabalho discute o projeto e a implementação de um simulador de redes ATM, chamado **SimATM**. O **SimATM** foi desenvolvido em linguagem de programação C++ para o sistema operacional *Windows* 95/NT™. A implementação do **SimATM** em C++ permitiu usufruir das vantagens da programação orientada à objetos (OOP – *Object Oriented Programming*), tais como modularidade e reusabilidade. Recursos avançados desta linguagem foram utilizados para desenvolver o **SimATM**, tais como: herança múltipla, polimorfismo, *friends*, sobrecarga de operadores, *macros* e *templates*.

O **SimATM** emprega a técnica de simulação *event-driven*. A utilização desta técnica permitiu a definição de uma estrutura eficiente e flexível para o simulador. O gerenciador de eventos do **SimATM**, em conjunto com a sua estrutura de modelos, possui a robustez e o desempenho necessários à simulação de redes ATM. A estrutura de modelos do **SimATM** é expansível e modular. Novos modelos poderão ser desenvolvidos e integrados ao ambiente de simulação em versões futuras.

A arquitetura das redes ATM foi modelada a partir do Modelo de Referência de Protocolos da B-ISDN e da Arquitetura Funcional Geral de um Elemento de Rede ATM, ambos definidos pelo ITU-T. Desta forma é possível a incorporação ordenada de novos recursos de simulação ATM aos modelos do simulador.

A versão atual do **SimATM** conta com quatro tipos de modelos: modelos de equipamentos, modelos de aplicativos, modelos de camadas e modelos de sistemas de fila. Os modelos de equipamentos e aplicativos ATM podem agrupar várias instâncias de modelos de camadas e de sistemas de fila. O modelo de sistema de fila foi desenvolvido de acordo com a teoria de filas. As variáveis de estado instantâneas e médias de cada sistema de fila de uma rede ATM podem ser amostradas para arquivo de duas formas: amostragem por tempo e amostragem por ocorrência. Através da especificação deste tipo de amostragem, dois tipos de simulações podem ser realizadas: simulação em regime permanente e simulação em regime transitório. Os resultados das simulações em regime permanente podem ser comparados diretamente com a teoria de filas.

Na versão atual do **SimATM** não é possível trocar os modelos existentes no simulador sem a necessidade de recompilação do programa. Portanto, a atuação do usuário está restrita a configuração de parâmetros nos modelos existentes. Contudo, a inclusão do recurso de desenvolvimento de modelos como bibliotecas de ligação dinâmica (DLLs – *Dinamic Linkage Libraries*) permitirá a troca facilitada de modelos.

Neste trabalho apresentamos um exemplo de simulação de rede ATM que demonstra a potencialidade do simulador. Foram feitas simulações em regime transitório e em regime permanente. Os resultados da simulação em regime permanente foram validados através da lei de *Little*. A partir deste exemplo de simulação concluímos que o desempenho do simulador é satisfatório e que os objetivos iniciais deste trabalho foram alcançados.

9.1 - Sugestões para Trabalhos Futuros

Basicamente duas linhas de ação podem ser tomadas para expandir os recursos do simulador: a expansão dos recursos de simulação ATM e a expansão dos recursos do ambiente de simulação. A seguir deixamos como sugestão algumas propostas para a expansão destes recursos.

9.1.1 - Expansão dos Recursos de Simulação ATM

Tomando como base a arquitetura funcional geral de um elemento de rede ATM, anteriormente apresentada no Capítulo 3, deixamos como sugestão a seguinte seqüência cronológica para a expansão dos recursos de simulação ATM do simulador:

1°) Funções de Transferência – Implementação das camadas de adaptação tipo 1, 3/4 e de sinalização ATM (SAAL – *Signaling* ATM *Adaptation Layer*).

2°) Funções de Gerenciamento de Camadas – Implementação dos recursos necessários a troca de informações com a função de coordenação, função de gerenciamento de equipamento ATM (AEMF – ATM *Equipment Management Function*) (através da função de coordenação) e entre funções de gerenciamento de camadas. Implementação do processamento em tempo real dos fluxos de operação, administração e manutenção (OAM – *Operation, Administration* and *Maintenance*) e gerenciamento das camadas de adaptação tipo 1, ¾ e de sinalização (SAAL). A implementação dos fluxos de OAM permitirá a estimação de parâmetros de qualidade de serviço e de parâmetros de tráfego, fundamentais para a incorporação de algoritmos de controle de admissão de conexões (CAC – *Connection Admission Control*).

3°) Aplicativo de Sinalização – Implementação das funções necessárias ao gerenciamento sob demanda de conexões SVCs.

4°) Função de Coordenação – Implementação da coordenação entre diferentes funções de gerenciamento de camadas e do processamento de requisições de pedido por recursos. A implementação destas funções permitirá:

- A simulação de protocolos de sinalização ATM, fornecendo os recursos necessários para o estabelecimento, término e modificações de conexões virtuais chaveadas (SVCs – *Switched Virtual Connections*).
- A alocação dinâmica de recursos na rede, tais como: largura de faixa, VPI/VCI e espaço em *buffers*.
- Projeto e teste de algoritmos e funções de gerenciamento de tráfego, tais como: controle de admissão de conexões (CAC), controle de utilização de parâmetros (UPC – *Usage Parameter Control*), controle de prioridade de perda de células (*Cell Loss Priority Control*), gerenciamento de recursos da rede (*Network Resource Management*) e controle de fluxo ABR (ABR *Flow Control*).

5°) Função de Gerenciamento de Equipamento ATM (AEMF) – Implementação das funções relacionadas com o gerenciamento de falhas, desempenho, configuração, usuários e segurança.

6°) Função de Comunicação de Mensagem (MCF – *Message Communication Function*) – Implementação da troca de informações entre a AEMF e o sistema de gerenciamento de rede (NMS – *Network Management System*), o que permitirá a simulação do gerenciamento da rede como um todo. O estabelecimento de conexões de gerenciamento da rede poderá ser realizada.

9.1.2 - Expansão dos Recursos do Ambiente de Simulação

Deixamos como sugestão as seguintes propostas para a expansão do ambiente de simulação **SimATM**:

- Interface Gráfica – O desenvolvimento de uma interface gráfica permitirá a interação direta do usuário com o simulador, sem a necessidade de digitação de comandos. A interface gráfica do **SimATM** deverá ser desenvolvida para o sistema operacional *Windows* 95/NT.

- Modelos como Bibliotecas de Ligação Dinâmica – A utilização de bibliotecas de ligação dinâmica (DLLs) permitirá a substituição de modelos de equipamentos, aplicativos, camadas e sistemas de fila. Modelos de equipamentos com características específicas de um fabricante poderão ser desenvolvidos.
- Aperfeiçoamento da Estrutura de Amostragem de Resultados – Atualmente, o SimATM não permite selecionar individualmente as variáveis de saída que serão salvas em arquivo. Esta seleção é feita apenas para grupos de variáveis. A inclusão deste recurso permitirá uma maior seletividade na amostragem de resultados de simulação.
- Estrutura Genérica para as Unidades de Dados de Protocolos – Atualmente, as unidades de dados de protocolos (PDUs e SDUs) são implementadas em C++ separadamente dos modelos de camadas correspondentes a estes protocolos. O desenvolvimento de uma estrutura genérica de unidades de dados permitirá o desenvolvimento de modelos de camadas que definam internamente as unidades de dados dos seus protocolos.
- Expansão dos Sistemas de Fila para o Armazenamento de Pacotes em Geral – O modelo de sistema de fila utilizado atualmente no **SimATM** permite apenas o armazenamento de células ATM. A idéia é estender este modelo a fim de permitir o armazenamento de pacotes de forma geral.
- Novos Modelos de Equipamentos, Aplicativos, Camadas e Sistemas de Fila – O desenvolvimento de novos modelos é fundamental para a ampliação dos recursos do **SimATM**, e tornará a ferramenta muito mais flexível.
- Definição de Subredes – A implementação deste recurso permitirá que vários equipamentos e aplicativos ATM sejam agrupados em uma subrede ATM.

Apêndice A

Fluxogramas dos Processos das Camadas dos Equipamentos ATM

A.1 - Camada AAL 5

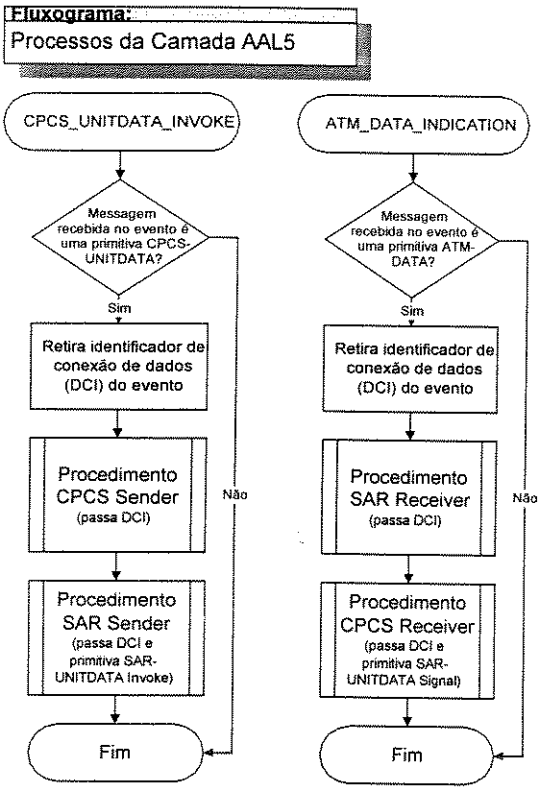


Figura A.1 – Fluxograma dos processos do modelo camada AAL 5

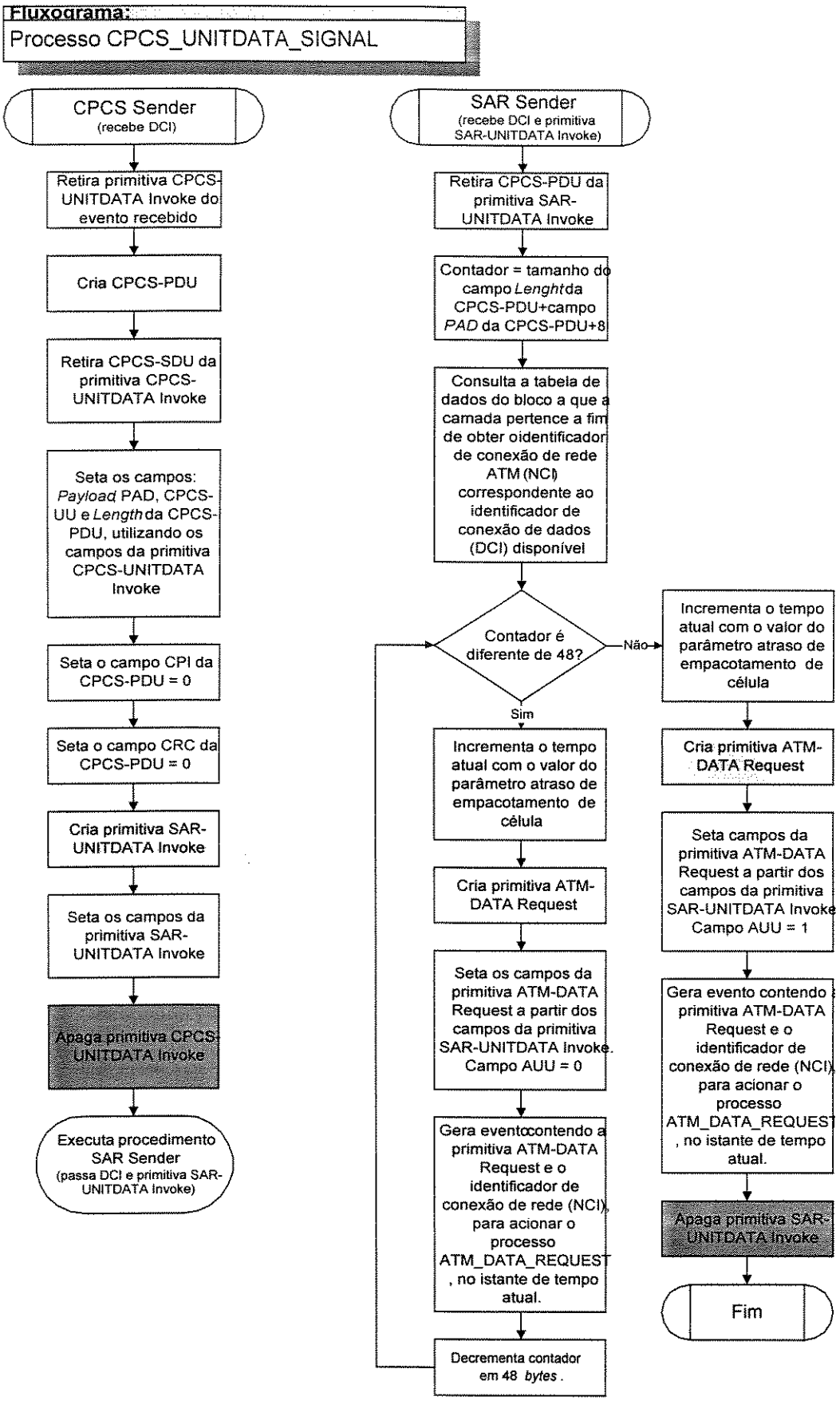


Figura A.2 – Fluxograma do processo CPCS_UNITDATA_SIGNAL

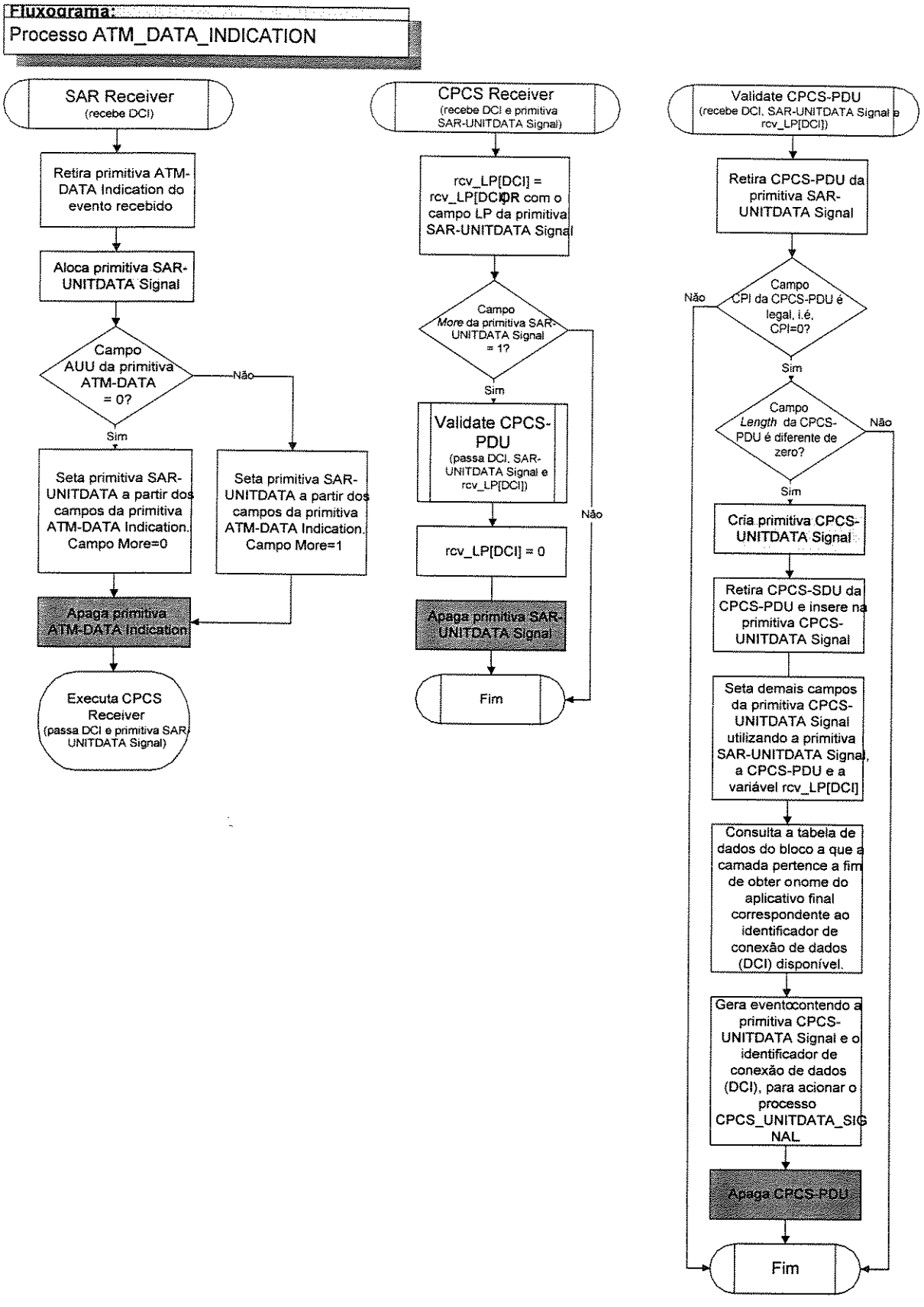


Figura A.3 – Fluxograma do processo ATM_DATA_INDICATION

A.2 - Camada ATM do BTE

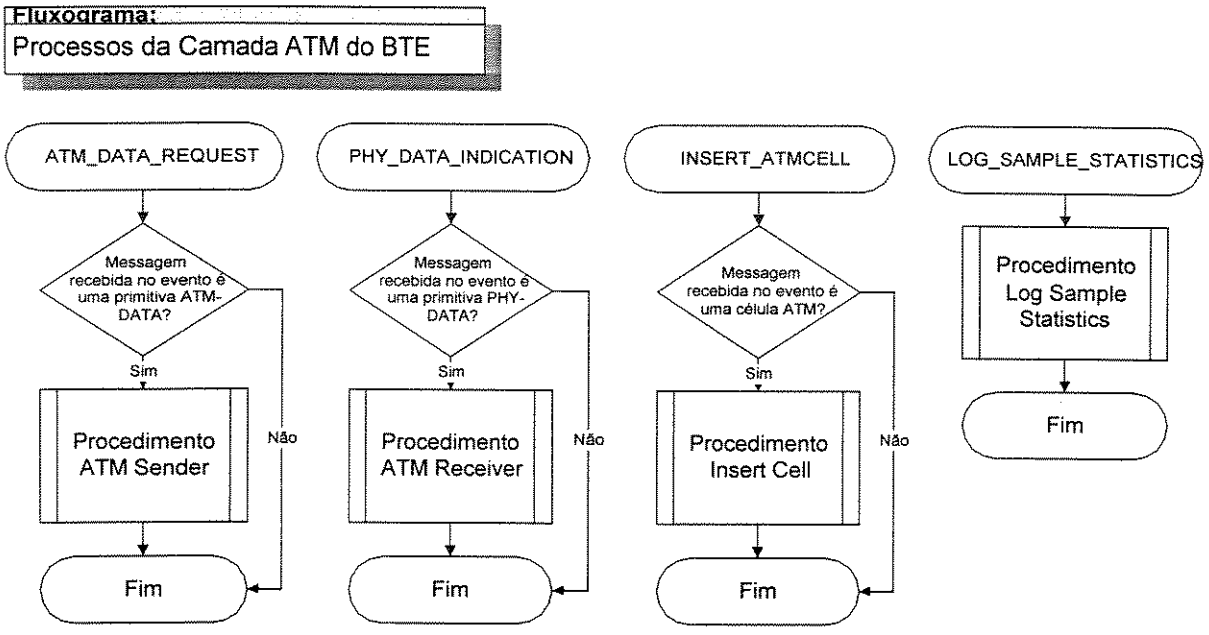


Figura A.4 – Fluxograma dos processos do modelo camada ATM do BTE

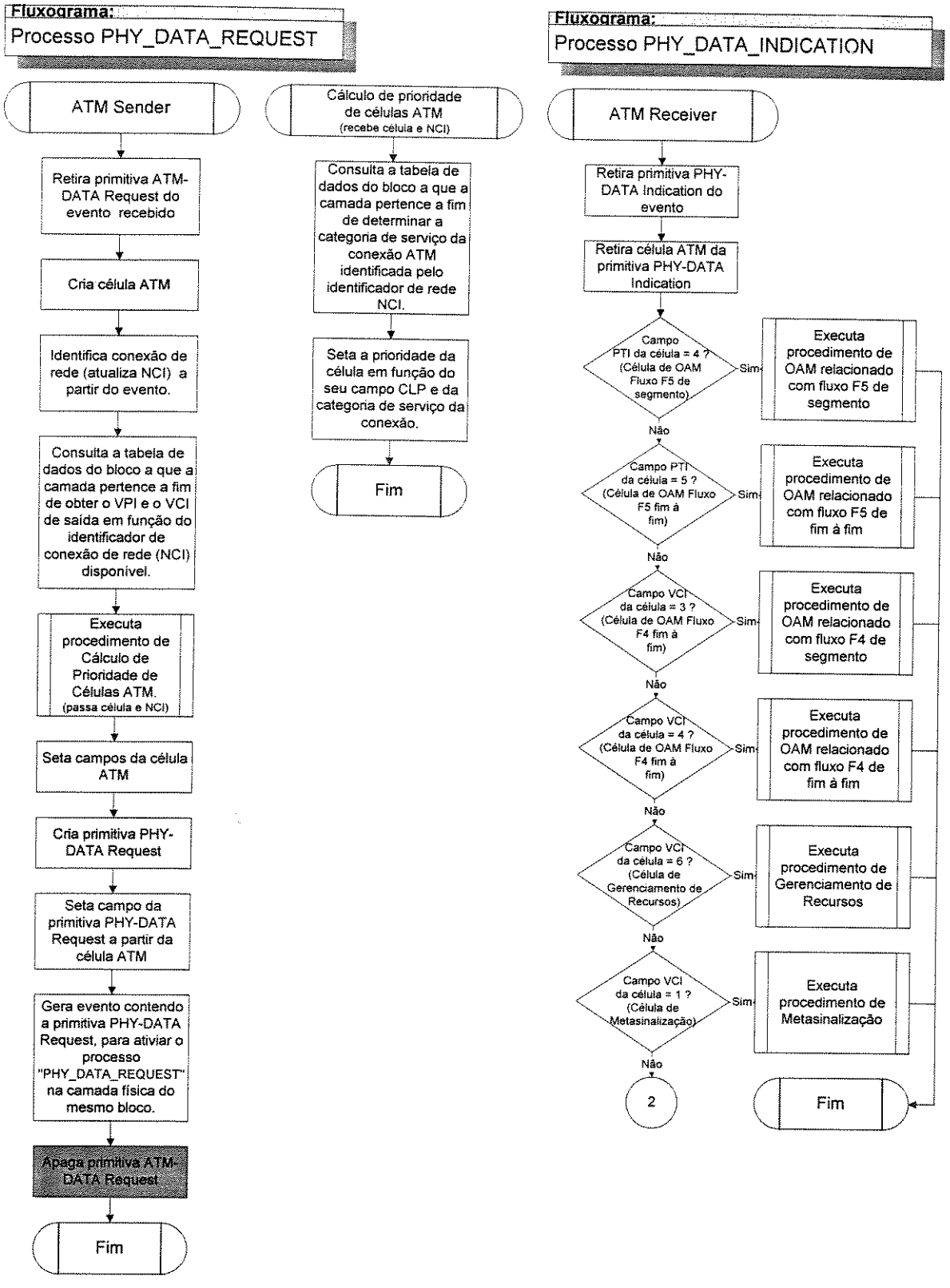


Figura A.5 – Fluxogramas dos processos PHY_DATA_REQUEST e PHY_DATA_INDICATION

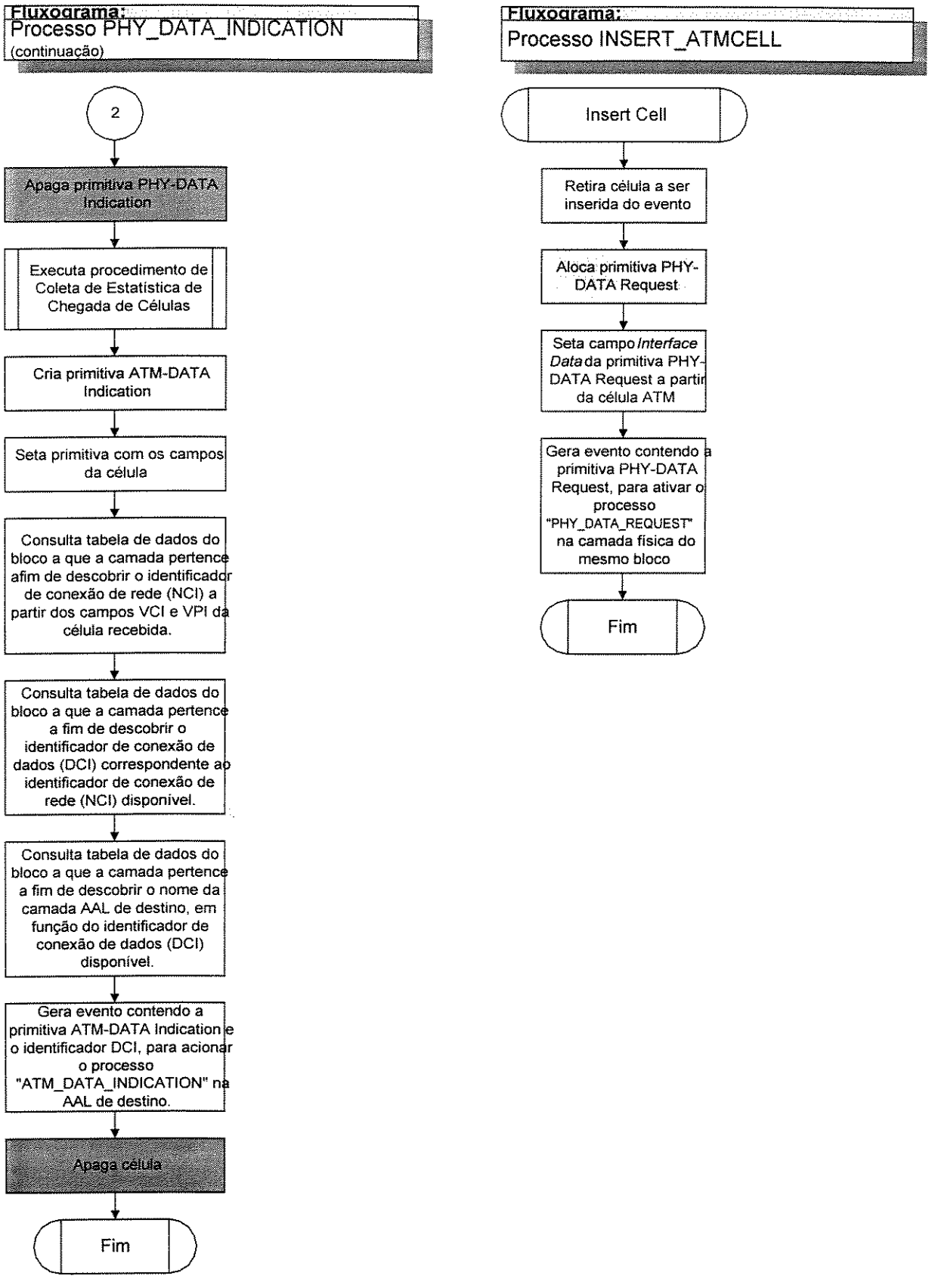


Figura A.6 – Fluxogramas dos processos PHY_DATA_INDICATION e INSERT_ATMCELL

A.3 - Camada ATM do Chaveador

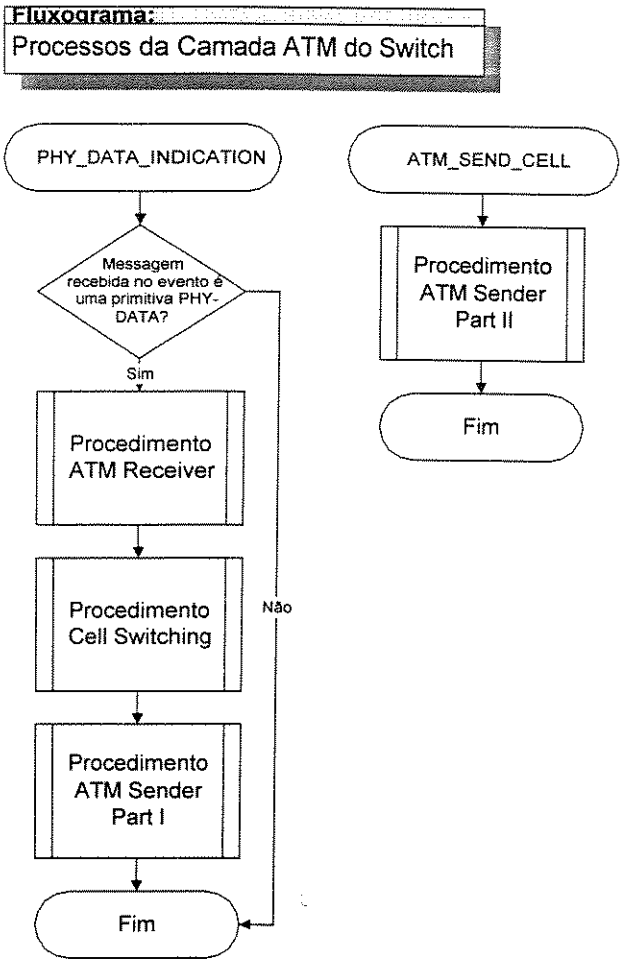


Figura A.7 – Fluxograma dos processos do modelo camada ATM do chaveador

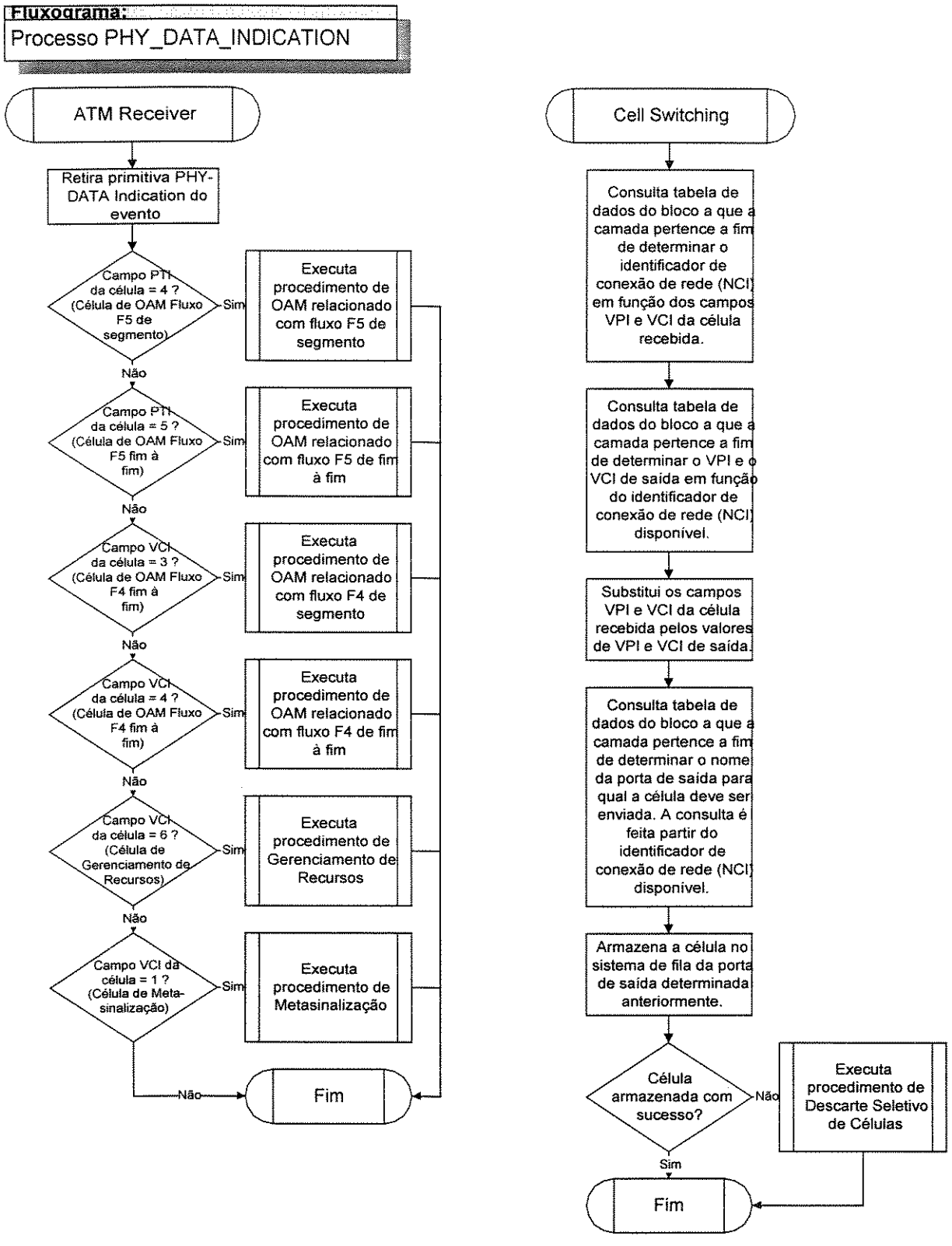


Figura A.8 – Fluxograma do processo PHY_DATA_INDICATION

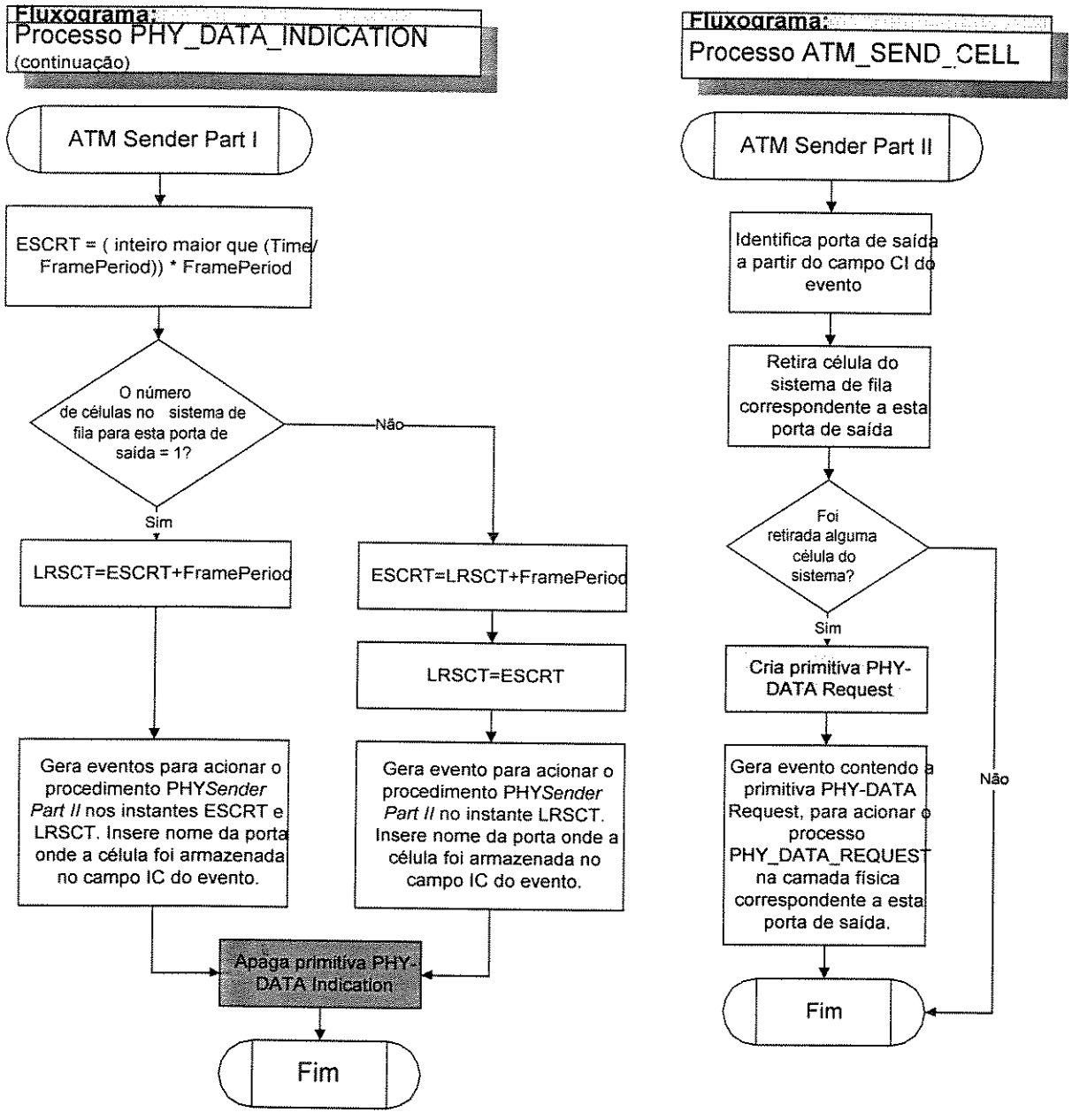


Figura A.9 – Fluxogramas dos procedimentos PHY_DATA_INDICATION e ATM_SEND_CELL

A.4 - Modelo da Camada Física

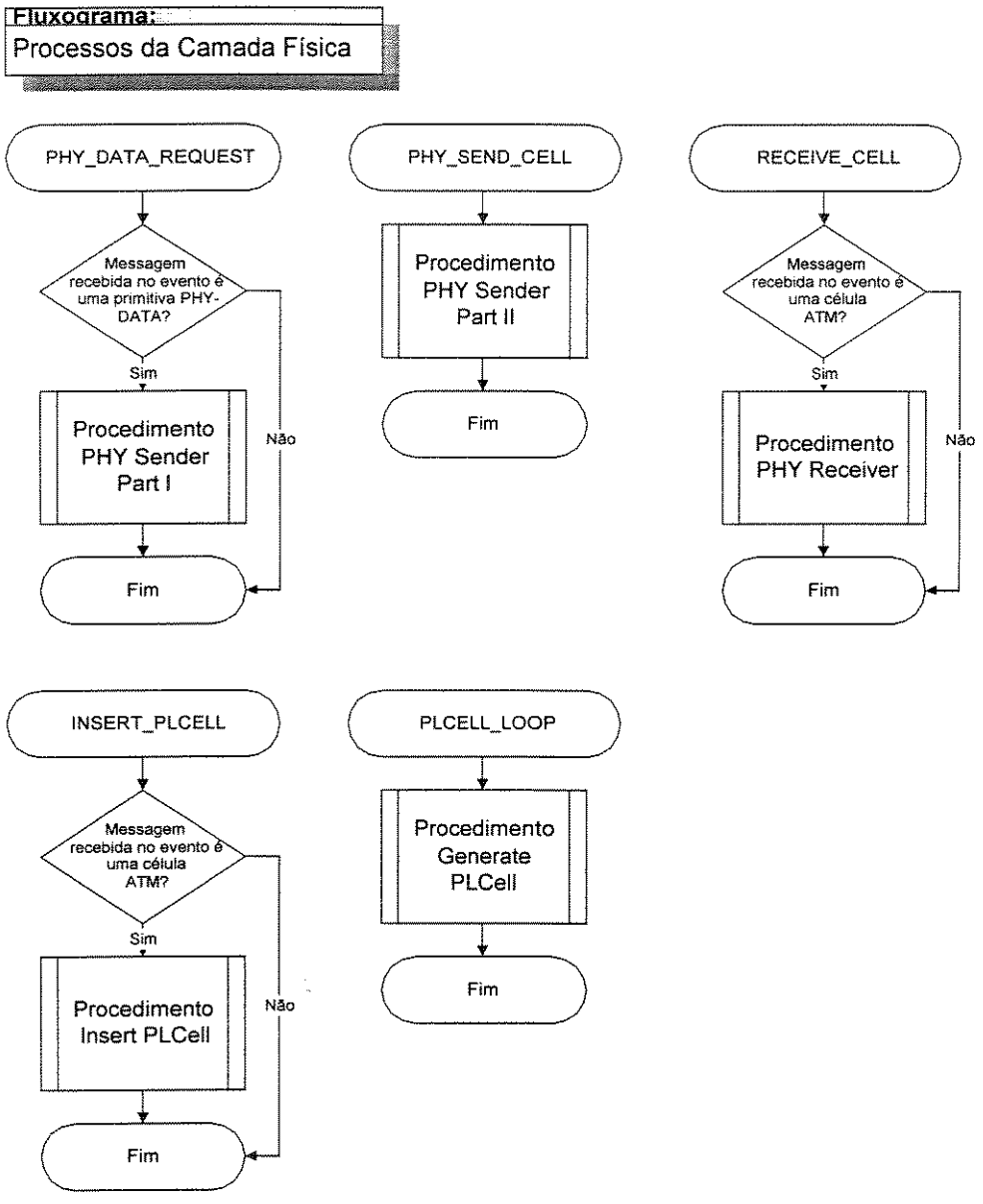


Figura A.10 – Fluxograma dos processo do modelo camada física para a interface baseada em células

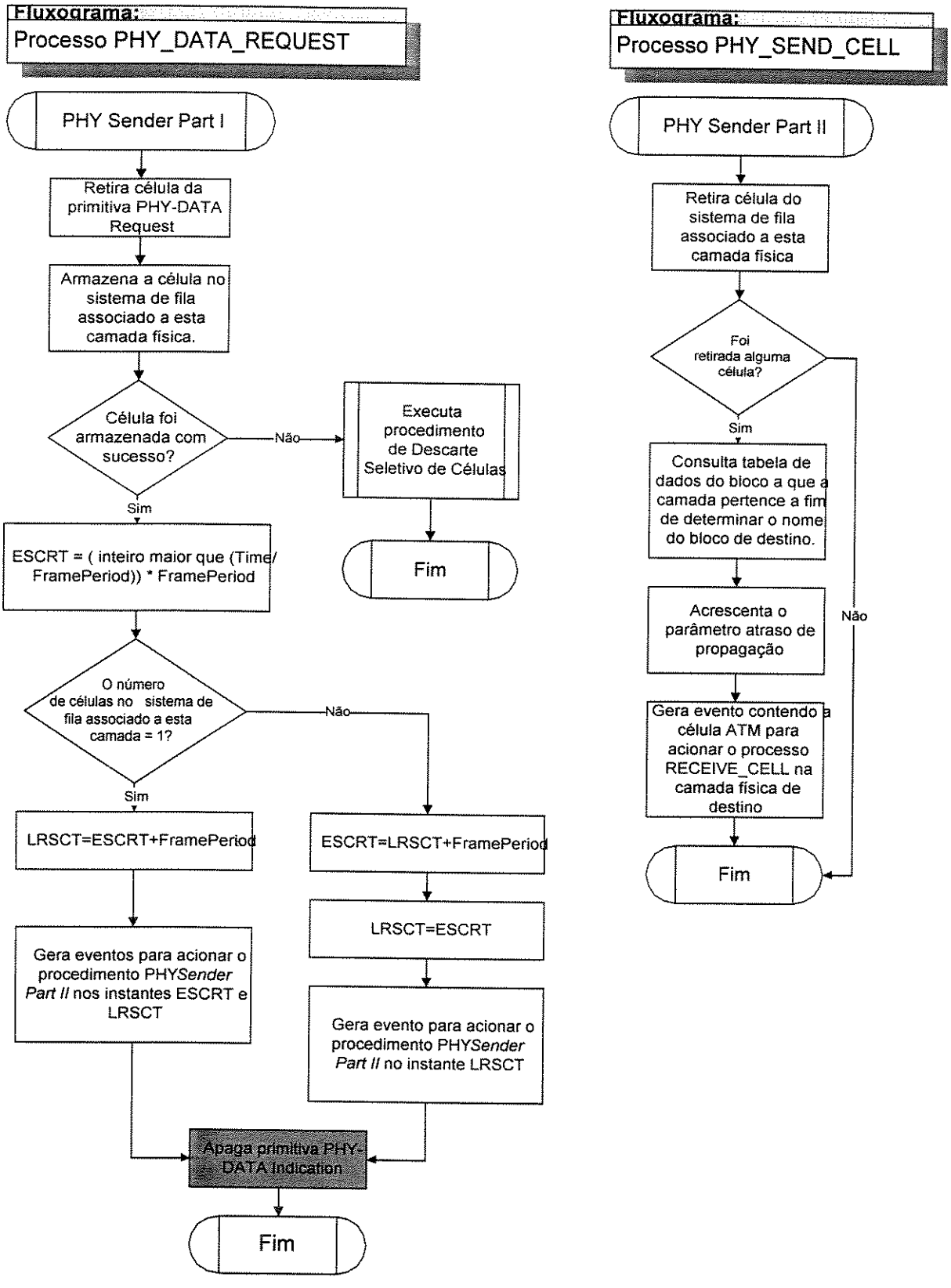


Figura A.11 – Fluxogramas dos processos PHY_DATA_REQUEST e PHY_SEND_CELL

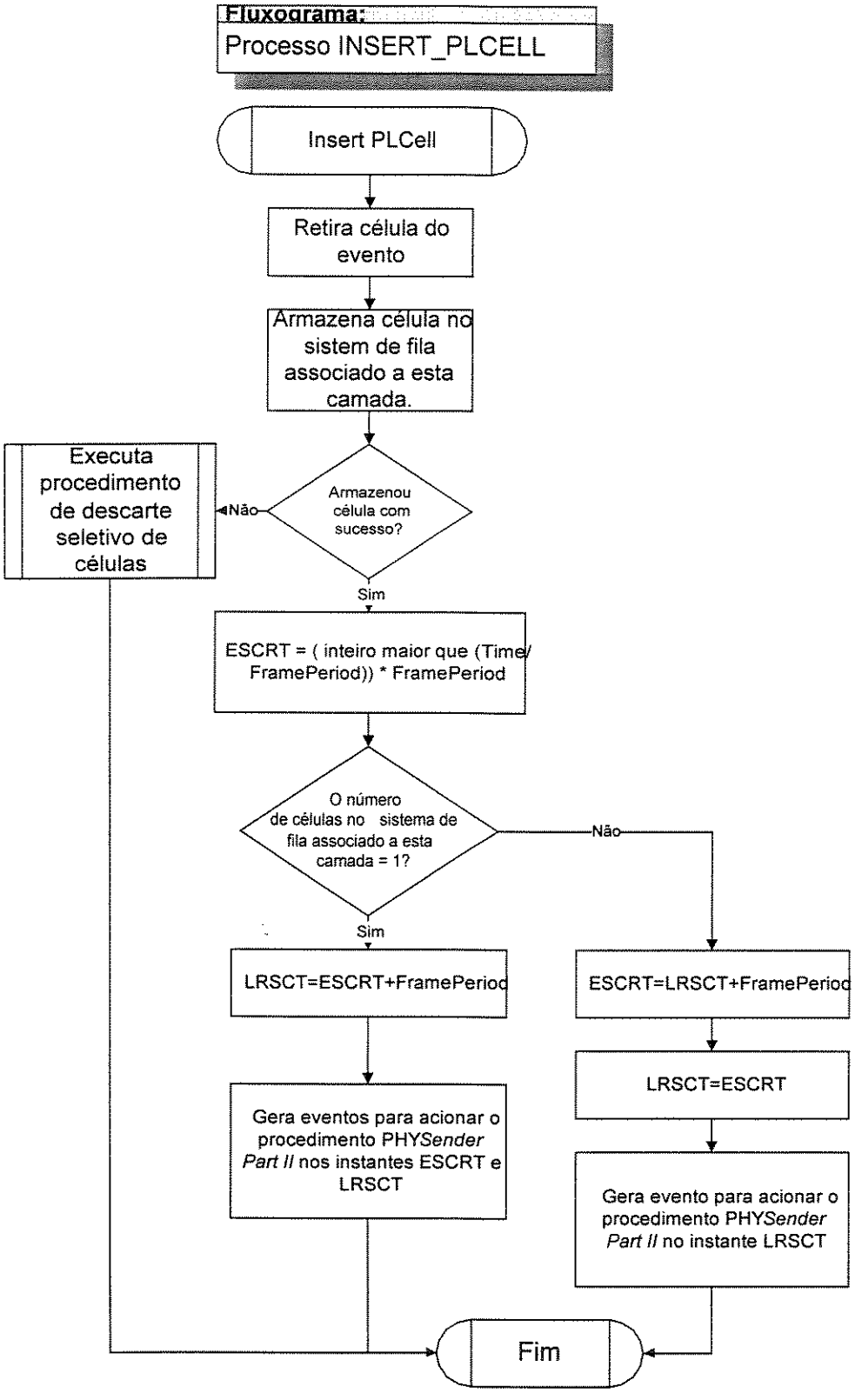


Figura A.12 – Fluxograma do processo INSERT_PLCELL

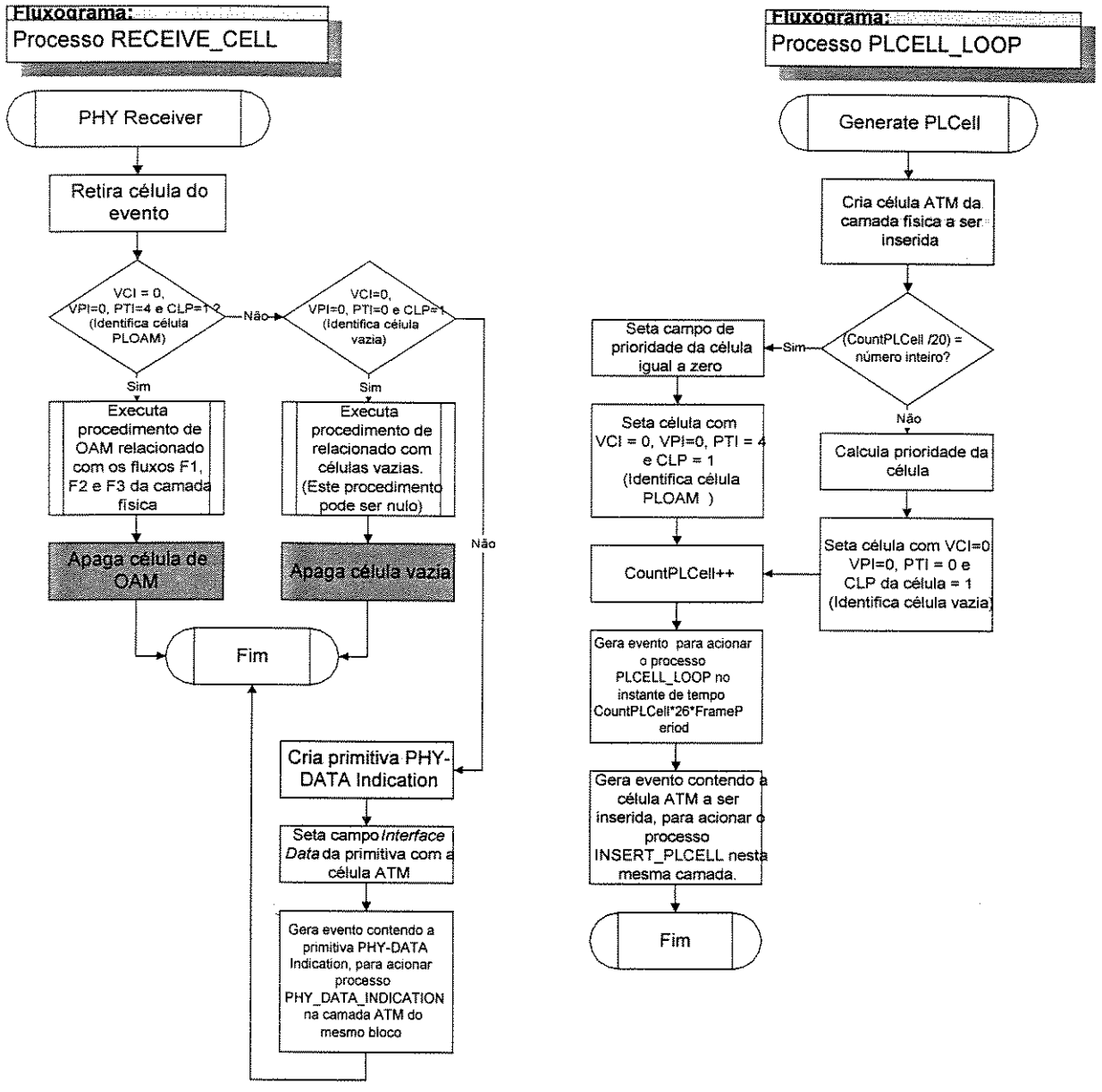


Figura A.13 – Fluxogramas dos processos RECEIVE_CELL e PLCELL_LOOP

Apêndice B

Linguagem de Comandos do **SimATM**

A seguir apresentaremos uma breve descrição dos comandos atualmente disponíveis na linguagem de comandos do **SimATM**.

create network

Cria uma rede ATM	
Sintaxe	*Create network network_name*
Descrição	Este comando cria uma rede ATM com o nome *network_name*.

create block

Cria um bloco em uma rede ATM	
Sintaxe	*Create block network_name block_name*
Descrição	Este comando cria um bloco de nome *block_name* na rede ATM de nome *network_name*.

create vcc

Cria uma conexão VCC em uma rede ATM	
Sintaxe	*Create vcc network_name NCI service_categorie bandwidth first_block_name*
Descrição	Este comando cria uma conexão VCC de nome *NCI* na rede ATM de nome *network_name*. *service_categorie*, *bandwidth* e *first_block_name* são respectivamente a categoria de serviço, a largura de faixa e o nome do primeiro bloco da nova conexão ATM.

create link

Cria um enlace em uma rede ATM	
Sintaxe	*create vcc network_name NCI name1 name2 VPI VCI*
Descrição	Este comando cria um enlace entre os equipamentos de nome *name1* e *name2* na rede ATM de nome *network_name*. Este enlace pertence a conexão ATM de nome *NCI*. Os parâmetros VPI e VCI são os identificadores virtuais deste novo enlace.

create dc

Cria uma conexão de dados em uma rede ATM	
Sintaxe	*create dc network_name DCI begin_app_name end_app_name*
Descrição	Este comando cria uma conexão de dados de nome *DCI* na rede ATM de nome *network_name*. *begin_app_name* e *end_app_name* são os nomes dos aplicativos

fonte e destino da nova conexão de dados.

remove network

Remove uma rede ATM	
Sintaxe	*remove network network_name*
Descrição	Este comando remove a rede ATM de nome *network_name.*

remove block

Remove um bloco de uma rede ATM	
Sintaxe	*remove block network_name block_name*
Descrição	Este comando remove o bloco de nome *block_name* da rede ATM de nome *network_name.*

remove vcc

Remove uma conexão VCC de uma rede ATM	
Sintaxe	*remove vcc network_name NCI*
Descrição	Este comando remove a conexão de nome *NCI* da rede ATM de nome *network_name.*

remove link

Remove um enlace de uma rede ATM	
Sintaxe	*remove link network_name NCI name1 name2*
Descrição	Este comando remove o enlace entre os blocos de nome *name1* e *name2* da conexão de nome *NCI* em uma rede ATM de nome *network_name.*

remove dc

Remove uma conexão de dados de uma rede ATM	
Sintaxe	*remove network_name DCI*
Descrição	Este comando remove a conexão de dados de nome *DCI* da rede ATM *network_name.*

connect block

Conecta um bloco de uma rede ATM	
Sintaxe	*connect block network_name name1 name2 interface*
Descrição	Este comando conecta o bloco de nome *name1* ao bloco de nome *name2* utilizando a interface de nome *interface.*

connect vcc block

Conecta bloco a uma conexão VCC em uma rede ATM	
Sintaxe	*connect vcc block network_name NCI new_block_name key_block_name*
Descrição	Este comando conecta um bloco de nome *new_block_name* ao conjunto de blocos que fazem parte da conexão de nome *NCI.* Este bloco é conectado ao bloco de nome

key_block_name.

disconnect block

Desconecta um bloco de uma rede ATM	
Sintaxe	*disconnect block network_name block_name*
Descrição	Este comando desconecta o bloco de nome *block_name* de todos os outro blocos da rede ATM de nome *network_name.*

disconnect vcc block

Desconecta bloco de uma conexão VCC em uma rede ATM	
Sintaxe	*disconnect vcc block network_name NCI old_block_name key_block_name*
Descrição	Este comando desconecta o bloco de nome *old_block_name* dos blocos que fazem parte da conexão de nome *NCI* da rede ATM de nome *network_name.*

show network

Mostra os blocos de uma rede ATM	
Sintaxe	*show network network_name*
Descrição	Este comando mostra os blocos da rede ATM de nome *network_name.*

show block

Mostra as características de um bloco de uma rede ATM	
Sintaxe	*show block network_name block_name*
Descrição	Este comando mostra o nome, o tipo e o modelo do bloco de nome *block_name* da rede ATM de nome *network_name.*

show connections

Mostra as conexões entre os blocos de uma rede ATM	
Sintaxe	*Show connections network_name*
Descrição	Este comando mostra as conexões entre os blocos da rede ATM de nome *network_name.*

show vcc

Mostra uma conexão VCC de uma rede ATM	
Sintaxe	*show vcc network network_name NCI*
Descrição	Este comando mostra as características da conexão de nome *NCI* da rede ATM de nome *network_name.*

show vcc blocks

Mostra os blocos de uma conexão VCC de uma rede ATM	
Sintaxe	*show vcc blocks network_name NCI*
Descrição	Este comando mostra os blocos que fazem parte da conexão de nome *NCI* da rede ATM de nome *network_name.*

show vcc link

Mostra um enlace de uma conexão VCC de uma rede ATM	
Sintaxe	*show vcc link network_name NCI name1 name2*
Descrição	Este comando mostra o enlace entre os blocos de nome *name1* e *name2* que faz parte da conexão ATM de nome *NCI* de uma rede ATM de nome *network_name*.

show vcc links

Mostra os enlaces de uma conexão VCC de uma rede ATM	
Sintaxe	*show vcc links network_name*
Descrição	Este comando mostra os enlaces que fazem parte de uma conexão ATM de nome *NCI* de uma rede ATM de nome *network_name*.

show vccs network

Mostra as conexões VCC de uma rede ATM	
Sintaxe	*show vccs network network_name*
Descrição	Este comando mostra as conexões ATM da rede ATM de nome *network_name*.

show vccs blocks

Mostra os blocos de todas as conexões VCC de uma rede ATM	
Sintaxe	*show vccs blocks network_name*
Descrição	Este comando mostra os blocos que fazem parte de cada conexão ATM da rede ATM de nome *network_name*.

show vccs links

Mostra os enlaces de todas as conexões VCC de uma rede ATM	
Sintaxe	*show vccs links network_name*
Descrição	Este comando mostra todos os enlaces que fazem parte de cada conexão ATM da rede ATM de nome *network_name*.

show dc

Mostra uma conexão de dados de uma rede ATM	
Sintaxe	*show dc network_name DCI*
Descrição	Este comando mostra as características da conexão de dados de nome *DCI* da rede ATM de nome *network_name*.

show dcs

Mostra as conexões de dados de uma rede ATM	
Sintaxe	*show dcs network_name*
Descrição	Este comando mostra todas as conexões de dados existentes na rede ATM de nome *network_name*.

show table network

Mostra o conteúdo da tabela de dados de uma rede ATM	
Sintaxe	*show table network network_name*
Descrição	Este comando mostra o conteúdo da tabela da rede ATM de nome *network_name.*

show table block

Mostra o conteúdo da tabela de dados de um bloco de uma rede ATM	
Sintaxe	*show table block network_name block_name*
Descrição	Este comando mostra o conteúdo da tabela de dados do bloco de nome *block_name* da rede ATM de nome *network_name.*

show table layer

Mostra o conteúdo da tabela de dados de uma camada de um bloco de uma rede ATM	
Sintaxe	*show table block network_name block_name layer_name*
Descrição	Este comando mostra o conteúdo da tabela de dados da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name.*

show parameters network

Mostra os parâmetros de uma rede ATM	
Sintaxe	*show parameters network network_name*
Descrição	Este comando mostra os parâmetros da rede ATM de nome *network_name.*

show parameters block

Mostra os parâmetros de um bloco de uma rede ATM	
Sintaxe	*show parameters network network_name block_name*
Descrição	Este comando mostra os parâmetros do bloco de nome *block_name*, pertencente a rede ATM de nome *network_name.*

show parameters layer

Mostra os parâmetros de uma camada de um bloco de uma rede ATM	
Sintaxe	*show parameters network network_name block_name layer_name*
Descrição	Este comando mostra os parâmetros da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name.*

show parameters QS

Mostra os parâmetros de um QS de um bloco de uma rede ATM	
Sintaxe	*show parameters network network_name block_name queueing_system_name*
Descrição	Este comando mostra os parâmetros do sistema de fila de nome *queueing_system_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name.*

show parameters blocks

Mostra os parâmetros de todos os blocos de uma rede ATM	
Sintaxe	*show parameters blocks network_name*
Descrição	Este comando mostra os parâmetros de todos os blocos da rede ATM de nome *network_name.*

show parameters layers

Mostra os parâmetros de todas as camadas de todos os blocos de uma rede ATM	
Sintaxe	*show parameters layers network_name*
Descrição	Este comando mostra os parâmetros de todas as camadas, de todos os blocos, da rede ATM de nome *network_name.*

show parameters QSs

Mostra os parâmetros de todas os QSs de todos os blocos de uma rede ATM	
Sintaxe	*show parameters QSs network_name*
Descrição	Este comando mostra os parâmetros de todos os sistemas de filas, de todos os blocos, da rede ATM de nome *network_name.*

show parameter network

Mostra um parâmetro de uma rede ATM	
Sintaxe	*show parameter network network_name param_name*
Descrição	Este comando mostra o parâmetro de nome *param_name* da rede, ATM de nome *network_name.*

show parameter block

Mostra um parâmetro de um bloco de uma rede ATM	
Sintaxe	*show parameter block network_name block_name param_name*
Descrição	Este comando mostra o parâmetro de nome *param_name* do bloco de nome *block_name*, pertencente a rede ATM de nome *network_name.*

show parameter layer

Mostra um parâmetro de uma camada de um bloco de uma rede ATM	
Sintaxe	*show parameter layer network_name block_name layer_name param_name*
Descrição	Este comando mostra o parâmetro de nome *param_name*, da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name.*

show parameter QS

Mostra um parâmetro de um QS de um bloco de uma rede ATM	
Sintaxe	*Show parameter QS network_name block_name queueing_system_name param_name*

Descrição	Este comando mostra o parâmetro de nome *param_name* do sistema de fila de nome *queueing_system_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

put parameter network

Coloca um parâmetro em uma rede ATM

Sintaxe	*put parameter network network_name param_name param_type rows columns value description option*
Descrição	Este comando coloca o parâmetro de nome *param_name*, de tipo *param_type*, com *rows* linhas, *columns* colunas, de valor *value*, descrito por *description* e com a opção *option*, na rede ATM de nome *network_name*.

put parameter block

Coloca um parâmetro em um bloco de uma rede ATM

Sintaxe	*put parameter network network_name block_name param_name param_type rows columns value description option*
Descrição	Este comando coloca o parâmetro de nome *param_name*, de tipo *param_type*, com *rows* linhas, *columns* colunas, de valor *value*, descrito por *description* e com a opção *option*, no bloco de nome *block_name*, pertencente a rede ATM de nome *network_name*.

put parameter layer

Coloca um parâmetro em uma camada de um bloco de uma rede ATM

Sintaxe	*put parameter network network_name block_name layer_name param_name param_type rows columns value description option*
Descrição	Este comando coloca o parâmetro de nome *param_name*, de tipo *param_type*, com *rows* linhas, *columns* colunas, de valor *value*, descrito por *description* e com a opção *option*, na camada de nome *layer_name*, pertence ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

put parameter QS

Coloca um parâmetro em um sistema de fila de um bloco de uma rede ATM

Sintaxe	*put parameter network network_name block_name queueing_system_name param_name param_type rows columns value description option*
Descrição	Este comando coloca o parâmetro de nome *param_name*, de tipo *param_type*, com *rows* linhas, *columns* colunas, de valor *value*, descrito por *description* e com a opção *option*, no sistema de fila de nome *queueing_system_name*, pertence ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

del parameter network

Remove um parâmetro de uma rede ATM

Sintaxe	*del paramenter network network_name param_name*
Descrição	Este comando remove o parâmetro de nome *param_name* da rede ATM de nome *network_name*.

del parameter block

Remove um parâmetro de um bloco de uma rede ATM

Sintaxe	*del parameter block network_name block_name param_name*
Descrição	Este comando remove o parâmetro de nome *param_name* do bloco de nome *block_name*, pertencente a rede ATM de nome *network_name*.

del parameter layer

Remove um parâmetro de uma camada de um bloco de uma rede ATM

Sintaxe	*del parameter layer network_name block_name layer_name param_name*
Descrição	Este comando remove o parâmetro de nome *param_name*, da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

del parameter QS

Remove um parâmetro de um QS de um bloco de uma rede ATM

Sintaxe	*del parameter QS network_name block_name queueing_system_name param_name*
Descrição	Este comando mostra o parâmetro de nome *param_name* do sistema de fila de nome *queueing_system_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

put data network

Coloca um dado na tabela de uma rede ATM

Sintaxe	*put data network network_name key name type value*
Descrição	Este comando coloca o dado de nome *name*, de tipo *type*, com valor *value*, na chave *key* da tabela de dados da rede ATM de nome *network_name*.

put data block

Coloca um dado na tabela de um bloco de uma rede ATM

Sintaxe	*put data network network_name block_name key name type value*
Descrição	Este comando coloca o dado de nome *name*, de tipo *type*, com valor *value*, na chave *key* da tabela de dados do bloco de nome *block_name*, pertencente a rede ATM de nome *network_name*.

put data layer

Coloca um dado na tabela de uma camada de um bloco de uma rede ATM

Sintaxe	*put data network network_name block_name layer_name key name type value*
Descrição	Este comando coloca o dado de nome *name*, de tipo *type*, com valor *value*, na chave

key da tabela de dados da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

del key network

Remove uma chave da tabela de dados de uma rede ATM	
Sintaxe	*del key network network_name key*
Descrição	Este comando remove a chave *key* da tabela de dados da rede ATM de nome *network_name*.

del key block

Remove uma chave da tabela de dados de um bloco de uma rede ATM	
Sintaxe	*del key network network_name block_name key*
Descrição	Este comando remove a chave *key* da tabela de dados do bloco de nome *block_name*, pertencente a rede ATM de nome *network_name*.

del key layer

Remove uma chave da tabela de dados de uma camada de um bloco de uma rede ATM	
Sintaxe	*del key network network_name block_name layer_name key*
Descrição	Este comando remove a chave *key* da tabela de dados da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

del data network

Remove um dado da tabela de dados de uma rede ATM	
Sintaxe	*del data network network_name key data*
Descrição	Este comando remove o dado *data* da chave *key* da tabela de dados da rede ATM de nome *network_name*.

del data block

Remove um dado da tabela de dados de um bloco de uma rede ATM	
Sintaxe	*del data network network_name block_name key data*
Descrição	Este comando remove o dado *data* da chave *key* da tabela de dados do bloco de nome *block_name*, pertencente a rede ATM de nome *network_name*.

del data layer

Remove um dado da tabela de dados de uma camada de um bloco de uma rede ATM	
Sintaxe	*del data network network_name layer_name block_name key data*
Descrição	Este comando remove o dado *data* da chave *key* da tabela de dados da camada de nome *layer_name*, pertencente ao bloco de nome *block_name* e a rede ATM de nome *network_name*.

run

Executa a simulação de uma rede ATM	
Sintaxe	*run network_name simulation_time*
Descrição	Este comando executa a simulação da rede ATM de nome *network_name* até que o tempo de simulação (*simulation_time*) seja atingido.

file

Especifica o caminho e o nome do arquivo de rede de uma rede ATM	
Sintaxe	*file network_name file_name*
Descrição	Este comando especifica o nome *file_name* para o arquivo de rede da rede ATM de nome *network_name*. O parâmetro *file_name* deve conter o caminho completo até este arquivo.

save

Salva para um arquivo de rede a configuração de uma rede ATM	
Sintaxe	*save network_name*
Descrição	Este comando salva para um arquivo de rede a configuração da rede ATM de nome *network_name*.

load

Carrega a partir de um arquivo de rede a configuração de uma rede ATM	
Sintaxe	*load network_name file_name*
Descrição	Este comando carrega a partir do arquivo de nome *file_name* a configuração da rede ATM de nome *network_name*.