Modelagem e Análise de Mecanismos para Acesso de Banda Larga à Internet

por

Guilherme Dutra Gonzaga Jaime

UFRJ

Tese submetida para a obtenção do título de

Mestre em Engenharia de Sistemas e Computação

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

Coordenação de Pesquisas e Pós-graduação em Engenharia da UFRJ

por

Guilherme Dutra Gonzaga Jaime

Setembro 2003

MODELAGEM E ANÁLISE DE MECANISMOS PARA ACESSO DE BANDA LARGA À INTERNET

Guilherme Dutra Gonzaga Jaime

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

Aprovada por:

Prof. Rosa Maria Meri Leão, Dr.

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

Prof. José Ferreira de Rezende, Dr.

Prof. José Roberto Boisson de Marca, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2003

JAIME, GUILHERME DUTRA GONZAGA

Modelagem e Análise de Mecanismos para Acesso de Banda Larga à Internet [Rio de Janeiro] 2003

XVII, 127 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2003)

Tese - Universidade Federal do Rio de Janeiro, Pesc/COPPE

1. Acesso ao Meio 2. Redes sem Fio 3. Banda larga 4. Ferramenta Tangram-II 5. Modelagem e Análise de Desempenho 6. Modelos de Fonte de Tráfego 7. Simulação I. COPPE/UFRJ II. Título (Série)

"...nada disso é tudo e tudo isso é fundamental..."

(Engenheiros do Hawaii).

xxx

Agradecimentos

Gostaria de agradecer, em primeiro lugar, a Deus pelo fato de ter aberto as portas que tornaram possível minha jornada por mais essa importante etapa, e por ter me protegido e fortificado em todos os momentos de dificuldades. Gostaria de agradecer a meus pais, Lineu e Kátia, que apesar da distância sempre me mostraram o caminho certo nos momentos de dúvidas, sem esquecer em nenhum momento de me confortar com a demonstração do amor que têm por mim. Agradeço também a meus irmãos Caroline, Buiú e Boi pela força que sempre me passaram para me adaptar aos longos tempos longe de casa. Não posso esquecer também de agradecer especialmente, a meus avós Newton e Alice por terem me acolhido em sua casa durante todo este tempo e por terem, através de seu carinho, colaborado em muito para que eu seguisse nessa jornada.

Ao Beto e Lurdinha, eu deixo também meus agradecimentos pelo carinho com que me receberam. À Vanessa meus agradecimentos pela paciência nos momentos de ausência, pelo apoio e auxílio nos momentos de pressão e pelo carinho e atenção nos momentos de tristeza.

Aos professores que aqui conheci, por tudo de importante que me ensinaram, eu também gostaria de deixar meus sinceros agradecimentos. Especialmente os professores Edmundo e Rosa, que acreditaram em minha capacidade e com toda a sua dedicação, esforço e enorme capacidade de enxergar sempre à frente, me mostraram o caminho certo, tornando a ciência e a verdade uma realidade muito mais próxima do meu mundo.

Agradeço também a meus colegas de trabalho do LAND que de uma maneira ou outra me ajudaram no desenvolvimento deste trabalho, muito deles acabaram tornando-se meus amigos: Ana Paula, Allyson, Bruno, Bernardo, Drika, Fernando, Flávio, Guto, Kelvin, Magnos, Melba, Isabela, Sadoc e Sidney.

Não poderia deixar de agradecer à Carolina, que além de facilitar a solução de vários problemas, sempre teve boa vontade e paciência de nos ouvir com atenção, tendo sempre em mãos, os melhores conselhos.

Agradeço finalmente ao Governo Federal, em especial à CAPES, por ter me apoiado durante estes anos de estudo com a bolsa de mestrado. Meus agradecimentos também à Lucent e ao CPqD pela oportunidade de trabalhar em um importante projeto, resultantes de bolsa complementar que facilitou muito minha jornada.

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

MODELAGEM E ANÁLISE DE MECANISMOS PARA ACESSO DE BANDA LARGA À INTERNET

Guilherme Dutra Gonzaga Jaime

Setembro/2003

Orientadora: Rosa Maria Meri Leão

Departamento: Engenharia de Sistemas e Computação

Com a popularização do acesso à Internet, a demanda por serviços do tipo banda larga tem crescido de forma significativa. Serviços de alta velocidade são necessários para prover acesso a *web*, por exemplo de casa ou a partir de uma rede celular, com taxas de dados comparáveis as de uma rede local e, ainda, fornecer suporte para aplicações multimídia. Sob este prisma, tem sido feito um grande esforço para que diferentes padrões de redes, passem a atender serviços de dados que requerem banda larga. O CDMA2000 1X foi desenvolvido para que redes celulares, capazes de oferecer serviços de voz e outras aplicações de comutação de circuitos suportem aplicações de comutação de pacotes, oferecendo serviço de alta velocidade. O DOCSIS é outro padrão desenvolvido com objetivo similar, mas com foco nas redes de distribuição de TV à cabo. O objetivo deste trabalho é desenvolver modelos customizáveis para prever a vazão e o retardo alcançados por usuários que utilizam aplicações como navegação *web* e voz sobre IP, através das redes de acesso citadas, à medida que a população de usuários é aumentada. Uma característica importante dos modelos propostos é a representação detalhada do comportamento do usuário *web* e dos mecanismos estudados. Diversos experimentos foram realizados visando avaliar a influência do tráfego de voz no tempo de acesso dos usuários *web* e obter a carga suportada pelo sistema sem que haja degradação da QoS das aplicações.

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

MODELING AND ANALISYS OF MEDIUM ACCESS MECHANISMS FOR WIDE BAND INTERNET ACCESS

Guilherme Dutra Gonzaga Jaime

September/2003

Advisor: Rosa Maria Meri Leão

Department: Computer and System Engineering

With the widespread use of the Internet, the demand for broadband services is constantly increasing. High speed services, from home or using the cellular infrastructure, are necessary to provide web access and multimedia transmission such as voice and video with data rates equivalent to that of Local Area Networks.

These requirements led top development of protocols for different wireless standards of cellular networks and cable TV networks, to provide packet switched data services as well. Two important examples of these protocols are CDMA2000 1X and DOCSIS. The purpose of these protocols is support high-speed packet switched services using the available technology.

The goal of this work is to develop and implement simulation models to predict the access delay and throughput reached under different system loads as the number of users increases. Some important characteristics of the model are: accurate web user behavior model; voice over IP traffic are included on the model to study the delay of web users and the effect of call admission control on the system capacity.

Palavras-chave

1. Acesso ao Meio
2. Redes sem Fio
3. Banda larga
4. Ferramenta Tangram-II
5. Modelagem e Análise de Desempenho
6. Modelos de Fonte de Tráfego
7. Simulação

Glossário

ATM : Modo de Transferência Assíncrono (*Asynchronous Transfer Mode*);

CDMA: acesso múltiplo por divisão de código (*Code Division Multiple Access*);

CMTS: sistema terminal modem-cabo (*Cable Modem Termination System*);

CSMA: acesso múltiplo com detecção de portadora (*Carrier Sense Multiple Access*);

FCH: canal fundamental (*Fundamental Channel*);

FDMA: acesso múltiplo por divisão de freqüência (*Frequency Division Multiple Access*);

HFC: sistema híbrido de fibra óptica e cabo coaxial (*Hybrid Fiber Coax*);

HTTP: protocolo de transferência de hipertexto(*Hypertext Transfer Protocol*);

IP: protocolo inter-rede (*Internet Protocol*);

LAN: rede local (*Local Area Network*);

LLC: conexão de camada de enlace (*Link Layer Connection*);

MAC: controle de acesso ao meio (*Medium Access Control*);

PPP: protocolo ponto-a-ponto (*Point to Point Protocol*);

RLP: protocolo de enlace de rádio (*Radio Link Protocol*);

RTT: tempo de ida e volta (*Round Trip Time*);

SCH: canal suplementar (*Supplemental Channel*);

SDB: rajada curta de dados (*Short Data Burst*);

SMS: serviço de mensagens curtas (*Short Message Service*);

SRBP: protocolo de sinalização em rajadas de rádio (*Signaling Radio Burst Protocol*);

TCP: protocolo de controle de transmissão (*Transmission Control Protocol*);

TDMA: acesso múltiplo por divisão de tempo (*Time Division Multiple Access*);

TGIF: dispositivo de interface gráfica Tangram (*Tangram Graphic Interface Facility*);

UDP: protocolo de datagramas do usuário (*User Datagram Protocol*);

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

COM o desenvolvimento e o avanço das tecnologias utilizadas para acessar a Internet, novos padrões que possibilitam acesso de banda larga à rede têm sido desenvolvidos. Assim, é cada vez mais comum a utilização desses padrões para aplicações, como navegação *web* e transferência de arquivos, incluindo aquelas de tempo real, como teleconferência, ensino a distância e voz sobre IP.

Sob este prisma, neste trabalho foram desenvolvidos de modelos de simulação para avaliação de desempenho de mecanismos de acesso múltiplo, com foco em padrões para acesso de banda larga à Internet. Dois modelos detalhados, reproduzindo o protocolo DOCSIS (*Data-Over-Cable Service Interface Specification*) [1, 2, 3] e o sistema CDMA2000 1X [4, 5, 6] fazem parte das contribuições trazidas por esta dissertação.

Para a criação dos modelos propostos, foi necessário realizar algumas melhorias na ferramenta de simulação TANGRAM-II [7, 8], utilizada neste trabalho. Estas foram integradas à ferramenta e fazem parte da mais nova versão do ambiente de modelagem e análise TANGRAM-II.

Neste capítulo serão apresentados: (1.1) a motivação e os objetivos do trabalho e, também, uma breve descrição das técnicas de modelagem usadas; (1.2) a contribuição, onde os modelos e a ferramenta de simulação são introduzidos; e (1.3) o roteiro desta dissertação.

1.1 Motivação e Objetivos

Como acontece há vários anos, as redes de computadores e a Internet continuam se desenvolvendo rapidamente, tendo aumentado seu número de usuários, seu tamanho e, também, sua complexidade. Entre as novas tecnologias que estão surgindo, como resultado deste desenvolvimento acelerado, se destacam os protocolos de acesso com altas taxas de transmissão. Essas tecnologias têm como característica comum a disponibilização de acesso de banda larga à Internet, a partir de equipamentos que fornecem capacidade de transmissão superior à convencionalmente utilizada. Como exemplos de aplicações para as novas tecnologias de acesso, podem ser citados: o acesso de banda larga à Internet de casas ou edifícios em condomínios; a disponibilização de acesso à rede em ambientes como aviões, aeroportos e hotéis; e a capacidade de levar a Internet até usuários móveis que utilizam aparelhos celulares. Todo este desenvolvimento sugere estudos de desempenho para avaliar a eficiência dos protocolos.

Diversas técnicas, analíticas ou de simulação, de solução de modelos têm sido desenvolvidas nos últimos anos, visando a resolver, de forma eficiente, modelos que são cada vez mais complexos [9, 10, 11, 12, 13, 14, 15]. Estas técnicas tornam possível a análise detalhada de características e mecanismos relacionados ao comportamento de um sistema de computação, estando este sistema em fase de projeto ou em fase de implementação.

Métodos analíticos podem ter como principais vantagens o baixo custo computacional e a alta precisão. No entanto, exigem, muitas vezes, que simplificações sejam feitas no sistema a ser modelado. A simulação, por sua vez, permite a inclusão de detalhes do sistema estudado, sendo possível construir modelos mais próximos do sistema real. Devido a esta representação mais detalhada, o custo computacional pode ser bastante elevado em alguns casos.

O objetivo principal deste trabalho foi o desenvolvimento de modelos detalhados de simulação para mecanismos de acesso de banda larga à Internet. As principais medidas de interesse obtidas foram o tamanho médio e a utilização das filas do sistema, a vazão e o atraso de acesso dos usuários.

Parte do desenvolvimento deste trabalho é resultado de um projeto de cooperação entre a UFRJ, a Fundação CPqD/Centro de Pesquisa e Desenvolvimento em Telecomunicações e a Lucent, uma empresa de tecnologia em telecomunicações. O objetivo deste projeto foi avaliar o desempenho dos protocolos DOCSIS e CDMA2000 1X, quando submetidos ao tráfego gerados por usuários de voz e da *web*.

1.2 Contribuição

Há alguns anos, o LAND [16] (Laboratório de Modelagem/Análise e Desenvolvimento de Redes e Sistemas de Computação) da UFRJ desenvolve trabalhos de pesquisa, que têm resultado em diversas ferramentas da área de ensino a distância, medições, engenharia de tráfego e modelagem e análise de sistemas. Entre elas, destaca-se o TANGRAM-II, um conjunto de ferramentas que inclui um ambiente para análise e modelagem de sistemas.

Um dos atrativos do TANGRAM-II é disponibilizar um ambiente de modelagem onde é possível, através de programação de alto nível e utilizando uma linguagem baseada em C, criar e resolver modelos complexos. Estes modelos podem ser resolvidos usando simulação ou técnicas analíticas. Após a solução do modelo, a ferramenta oferece um módulo para o cálculo e visualização de medidas de interesse.

Optou-se por utilizar o ambiente de modelagem do TANGRAM-II no estudo dos protocolos DOCSIS e CDMA2000 1X, pois a ferramenta foi desenvolvida no LAND, sendo possível adicionar, facilmente, novas funcionalidades de acordo com as necessidades dos modelos.

A elaboração de modelos de fontes de tráfego que buscam reproduzir corretamente as características do tráfego gerado pelos usuários, ao utilizarem determinados tipos de aplicações, foi o primeiro passo visando à análise dos protocolos DOCSIS e CDMA2000 1X. Neste trabalho, são modeladas aplicações de voz sobre IP e de navegação *web*. A primeira por ser uma aplicação que possui requisitos estreitos de

QoS e, além disso, o seu uso na Internet vem crescendo bastante nos últimos anos. A segunda por ser, atualmente, uma das aplicações mais comuns da Internet.

Uma das contribuições deste trabalho consistiu na elaboração de três modelos integrados, no TANGRAM-II: um, para representar o comportamento do usuário *web* (baseado em [17, 18]) e dois outros, para representar a transferência dos dados entre o cliente e o servidor *web* através das redes de acesso dos padrões DOCSIS e CDMA2000 1X (baseados em [19, 20, 21]). Nos três modelos são usadas distribuições de cauda longa. A ferramenta TANGRAM-II não possuía este tipo de distribuição implementada. Portanto, foi necessário adicioná-las na ferramenta para o trabalho proposto.

Na intenção de representar o comportamento do usuário de maneira precisa e, também, construir os mecanismos de acesso dos protocolos DOCSIS e CDMA2000 1X em modelos detalhados, optou-se por utilizar a técnica de simulação como método de solução para o estudo desses protocolos.

Quando se trata de simulação de sistemas de computação, uma das abordagens mais utilizadas é a simulação orientada a eventos. Nesta técnica, o simulador reage toda vez que ocorre um evento, o tempo de simulação é medido em intervalos discretos e é incrementado a cada evento. Geração de pacotes, mudança de estado da fonte e o término do serviço de pacotes são exemplos de eventos de uma simulação de redes de computadores. Porém, quando se modela uma rede de alta velocidade, a simulação orientada a eventos pode ter seu custo computacional aumentado, devido a diferenças na taxa de ocorrência dos eventos.

Para contornar este problema, uma possível abordagem é utilizar simulação baseada em modelos de estado contínuo, também chamados de modelo de fluido. Este método substitui o fluxo de unidades discretas, que viajam através dos canais, por fluidos que se movem de um recipiente para o outro. Esta técnica considera como eventos apenas as mudanças de taxa na transferência dos fluidos. Como em geral as mudanças de taxa ocorrem com freqüência bem menor do que a geração de eventos, pode-se perceber com facilidade o ganho de custo computacional desta abordagem, quando comparada à simulação de eventos discretos.

Foram desenvolvidos dois modelos detalhados de redes de acesso à Internet, além dos modelos de fonte de tráfego. O primeiro modelo reproduz o comportamento do protocolo de acesso ao meio DOCSIS, utilizado tanto em redes HFC (*Hybrid Fiber Coax*), quanto em redes híbridas de acesso de banda larga à Internet, onde, por exemplo, uma antena disponibiliza o enlace com o provedor de acesso e o distribui a vários usuários através de meios físicos convencionais, como cabo coaxial. O segundo modelo representa o sistema CDMA2000 1X, que é um dos primeiros padrões para acesso de banda larga à Internet, através de redes celulares de terceira geração.

Devido à alta complexidade do modelo do padrão CDMA2000 1X, foi utilizada a abordagem de simulação de fluido para a modelagem dos usuários de voz sobre IP. Já o modelo dos usuários *web* foi desenvolvido com o paradigma de simulação de eventos discretos. Assim, o modelo global do CDMA2000 1X envolve a utilização mista de ambos os métodos de simulação: fluido e eventos.

Como os modelos dos usuários e da rede de acesso foram implementados de forma modular, os modelos de fonte de tráfego podem facilmente ser alterados para qualquer fonte de tráfego desejada. Os modelos das redes de acesso foram projetados para serem customizáveis, ou seja, um usuário, do ambiente de modelagem que deseja utilizar tais modelos, pode modificar um amplo número de parâmetros para que seja possível adaptá-lo ao equipamento específico que se deseja modelar. Por exemplo, no modelo do canal de acesso do DOCSIS, além de configurar a população de usuários a qual o sistema será testado, pode-se modificar parâmetros como o tamanho do *buffer*, os tamanhos dos *slots* de disputa e de dados e o número máximo de *slots* de dados por quadro, entre outros.

A contribuição deste trabalho inclui, também, a criação e a implementação de novas funcionalidades para a ferramenta TANGRAM-II, tornando possível o desenvolvimento dos modelos propostos, que possuem um alto grau de complexidade. As principais funcionalidades adicionadas foram: três funções de distribuição de probabilidade e quatro funções matemáticas, além da substituição da metodologia de geração de variáveis aleatórias.

1.3 Roteiro

O capítulo 2 apresenta a camada de enlace do modelo de referência TCP/IP e os padrões MAC para acesso à Internet abordados por este trabalho. O capítulo 3 descreve resumidamente a ferramenta de modelagem TANGRAM-II e introduz as novas funcionalidades implementadas. Os modelos de fontes de tráfego e das redes de acesso criados são apresentados no capítulo 4, onde são comentados também, alguns estudos relacionados encontrados na literatura. Os experimentos realizados e os seus resultados são apresentados no capítulo 5. Por fim, o capítulo 6 traz as conclusões e sugestões para trabalhos futuros.

Capítulo 2

A Camada de Enlace e Redes de Acesso à Internet

NESTE capítulo são discutidas as principais funções da camada de enlace e os principais métodos para acesso ao meio de transmissão. Estes métodos estão incluídos em um dos três grandes grupos: (1) protocolos de divisão de canal, (2) protocolos de acesso aleatório e (3) protocolos de revezamento. Os padrões para acesso de banda larga residencial e de banda larga sem fio modelados neste trabalho serão vistos com maior detalhamento e são, na verdade, baseados em protocolos dos três grupos citados acima.

2.1 Camada de Enlace

2.1.1 Introdução

A arquitetura de protocolos da Internet [22] consiste em cinco camadas: física, enlace, rede, transporte e aplicação.

Na Figura 2.1, o modelo em questão é apresentado. Na arquitetura de camadas, cada uma trata de uma parcela específica e bem definida dos problemas a serem resolvidos para possibilitar a comunicação entre duas aplicações. Esta abordagem

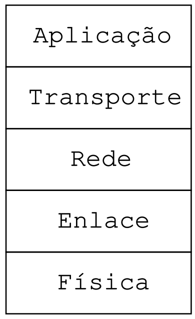


Figura 2.1: Modelo de referência TCP/IP

tanto diminui a complexidade do projeto da rede, quanto facilita a implementação dos serviços oferecidos, já que desde que uma camada forneça sempre os mesmos serviços à camada acima e utilize os mesmos serviços da inferior, as modificações feitas na maneira como esta camada implementa seus serviços não afetará o restante do sistema.

Neste modelo, a camada física possui a tarefa de movimentar *bits* entre pontos adjacentes de uma rede. Os protocolos da camada física definem que tipo de meio físico será utilizado para movimentar os sinais que representam os *bits* e também a maneira como os *bits* serão codificados nesses meios.

A segunda camada, chamada de camada de enlace, é responsável pela transferência de unidades de informação entre dois nós conectados em um mesmo enlace. Os serviços prestados por esta camada dependem de como os equipamentos estão interligados. Se é usado um meio compartilhado, então, a camada implementa mecanismos para regular o acesso ao meio. Além disso, algumas camadas de enlace oferecem transferência confiável.

Já a camada de rede é responsável pelo roteamento de unidades de informação de um nó origem até o nó destino. Outras funções da camada de rede, implementadas pelo IP *(Internet Protocol)* [22], são: o endereçamento, que permite identificar cada nó conectado à rede; o formato dos datagramas e a segmentação e reconstrução destes datagramas.

Como cada máquina conectada à Internet normalmente possui várias aplicações que utilizam simultaneamente a rede, é necessário que haja um mecanismo que permita a transferência de informação entre diferentes processos pertencentes a cada uma das máquinas onde se hospedam cada uma das aplicações. Esta é a tarefa da camada de transporte, que disponibiliza dois tipos de serviço de entrega de dados fim-a-fim [22]: o serviço de entregas confiável orientado à conexão disponibilizadas através do protocolo TCP *(Transmission Control Protocol)*, e o serviço de entregas sem conexão não confiável fornecidos pelo protocolo UDP *(User Datagram Protocol)*.

Finalmente, a camada de aplicação provê o suporte às aplicações de rede, definindo vários protocolos como o HTTP *(Hypertext Transfer Protocol)*, que dá suporte à navegação *web*, e o SMTP (*Simple Mail Transfer Protocol*) utilizado para transferências de correio eletrônico.

2.1.2 Serviços da Camada de Enlace

Apesar de a camada de enlace possuir, em primeira análise, a tarefa de transportar um datagrama entre dois nós adjacentes de um mesmo meio de comunicação, as técnicas utilizadas para implementar este serviço dependem de como os nós estão interconectados, seja através de um meio de difusão ou através de um meio ponto-a-ponto. Os principais serviços que podem ser oferecidos por um protocolo de camada de enlace serão discutidos nesta seção.

Primeiramente, para que o nó receptor trate separadamente todos os quadros enviados em seqüência pelo nó transmissor, é necessário que haja um método de delimitação ou enquadramento, onde são colocados marcadores nos limites dos quadros. Métodos como contagem de caracteres, preenchimento com caracteres, preenchimento com *bits* (*bit stuffing*) e violação da codificação da camada física ou alguma variação destes métodos geralmente são utilizados pelos protocolos da camada de enlace para o enquadramento.

Um quadro, então, consiste em um datagrama da camada de rede encapsulado em um quadro da camada de enlace com um cabeçalho que contém uma série de campos de controle a serem utilizados pelo par interessado na troca de dados.

Alguns enlaces de comunicação podem sofrer, devido às suas características físicas, interferências ou atenuações durante a transmissão de sinais, o que acaba gerando erros de transmissão. É desejável, nestes casos, que a camada de enlace ofereça garantia de que os quadros transmitidos serão corretamente entregues ao destinatário. Este serviço confiável normalmente é implementado através do envio de confirmações positivas ou negativas do receptor ao transmissor. Através do recebimento de confirmações, o transmissor consegue detectar que quadros não foram entregues corretamente e os retransmite ao receptor.

Outro fator que pode causar perdas em uma transmissão de dados entre dois pontos adjacentes é a limitação, que cada nó da rede possui, de armazenamento e processamento de quadros. Um nó pode receber, em um dado intervalo de tempo, mais quadros do que sua capacidade de processá-los. Neste caso o *buffer* do receptor pode acabar transbordando e os dados serão perdidos. Para evitar a ocorrência deste problema, a camada de enlace pode oferecer controle de fluxo entre os nós adjacentes. Exemplos de protocolos que implementam mecanismos de controle perdas/erros e controle de fluxo são o *go-back-n* e o *selective repeat* [22].

O serviço de entrega confiável geralmente é utilizado em enlaces que possuem alta taxa de erros. Entretanto, o *overhead* dos mecanismos usados para implementar este serviço não justifica o seu uso para enlaces que possuam baixa taxa de erros. Por isso, muitos dos mais difundidos protocolos de camada de enlace não fornecem o serviço de entrega confiável. Nestes casos, o encapsulamento de pacotes em quadros e o endereçamento são as principais funções da camada de enlace desses protocolos, além do controle de acesso ao meio, que é o foco dos estudos deste trabalho.

Existem basicamente duas maneiras de se interligar diferentes nós em uma rede de computadores [22]. A primeira técnica utiliza um enlace ponto-a-ponto. Vários protocolos como o PPP (*Point to Point Protocol*) e o HDLC (*High Level Data Link Control*) foram projetados para enlaces deste tipo. A segunda emprega um enlace de difusão (*broadcast*). Neste caso, vários nós são conectados a um único enlace de comunicação e todos eles devem compartilhar este meio de transmissão.

Para coordenar o acesso de vários nós a um mesmo enlace, essas redes, também chamadas de redes de difusão ou redes de acesso múltiplo, implementam um mecanismo de controle de acesso ao meio (*MAC - Medium Access Control*).

Vários protocolos para coordenação de acesso ao meio estão disponíveis na literatura e dois padrões específicos o DOCSIS e o CDMA2000 1X fazem parte deste trabalho. Por isso, os mecanismos de acesso ao meio serão mais cuidadosamente abordados na seção 2.2.

2.2 Mecanismos de Acesso ao Meio

Os mecanismos de acesso ao meio podem ser classificados em uma das seguintes categorias [22]: protocolos de acesso aleatório, protocolos de divisão de canal e protocolos de revezamento. Na seção 2.2.1, serão descritas cada uma dessas três categorias e na seção 2.3 serão apresentados os padrões MAC para acesso residencial de banda larga e sem fio estudados neste trabalho.

2.2.1 Protocolos de Acesso Aleatório

O primeiro protocolo de acesso ao meio criado para uma rede de comunicação de rádio, via satélite, na década de 70, foi o *Aloha* [23, 24]. Apesar deste protocolo praticamente não ser utilizado hoje em dia, ele é normalmente citado na literatura já que grande parte dos protocolos de acesso aleatório são baseados no *Aloha*.

Neste protocolo, quando um nó da rede deseja transmitir um quadro de dados, ele simplesmente o transmite. Caso seja detectada uma colisão, é iniciado um temporizador aleatório que, ao expirar, resulta em uma retransmissão.

Posteriormente, foi apresentada uma proposta para aumentar a eficiência do *Aloha*, o *Slotted-Aloha* [25, 24], onde o tempo é dividido em intervalos discretos chamados *slots*. Cada *slot* corresponde ao tempo necessário para a transmissão de um quadro de dados e uma transmissão só é efetuada no início de cada *slot*. Este protocolo exige que os nós estejam sincronizados. Para isso, uma estação poderia ser designada como líder e transmitiria um quadro de controle a cada ciclo de *clock* para permitir sincronização.

Os protocolos CSMA (*Carrier Sense Multiple Access*) [26], acesso múltiplo com detecção de portadora, constituem uma evolução dos protocolos *Aloha*. Estes protocolos definem que os nós que desejam transmitir devem, primeiramente, escutar o canal de comunicação para verificar se já existe algum nó transmitindo. Caso o canal esteja ocupado, existem duas possibilidades: (1) a estação continua a escutar o meio até que este fique ocioso e, então, transmite com probabilidade p (estes são os mecanismos chamados p-persistentes) ou (2) a estação aguarda um tempo aleatório e escuta o canal novamente (estes são denominados CSMA não persistentes). Caso o nó transmita o quadro e haja uma colisão, o nó espera um tempo aleatório e inicia uma nova tentativa de transmissão de acordo com um algoritmo (*binary exponential*) *backoff* [22], que busca diminuir a probabilidade de haver colisões à medida que vão acontecendo colisões consecutivas.

O protocolo CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) [22] é uma variação do CSMA onde, durante a transmissão do quadro, o nó transmissor permanece ouvindo o canal. Caso ele detecte a ocorrência de uma colisão, a transmissão corrente é abortada. Esta abordagem procura minimizar o tempo perdido quando ocorre uma colisão, já que a transmissão é suspensa assim que a colisão é detectada.

2.2.2 Protocolos de Divisão de Canal

Outra categoria de protocolos de acesso ao meio é aquela em que a capacidade de transmissão é dividida entre os nós. Exemplos de técnicas usadas são: o TDMA (*Time Division Multiple Access*), o FDMA (*Frequency Division Multiple Access*) e o CDMA (*Code Division Multiple Access*) [22].

No protocolo TDMA, a largura de banda do canal é dividida, entre todos os nós que compartilham o enlace em questão, através da alocação de *slots* para cada os nós. Assim, seja N o número de nós que acessam o enlace e R a taxa de transmissão em bits por segundo (bps) suportada pelo enlace, o protocolo TDMA divide o domínio do tempo em quadros e subdivide cada um desses quadros em $M \leq N$ intervalos de tempo chamados *slots*. Cada *slot* é, então, atribuído a um dos nós que requer acesso ao canal. Quando um nó tem um quadro para enviar, ele só tem permissão de transmitir no *slot* de tempo a ele designado. A Figura 2.2 ilustra o funcionamento do esquema acima descrito para o caso de 9 nós.

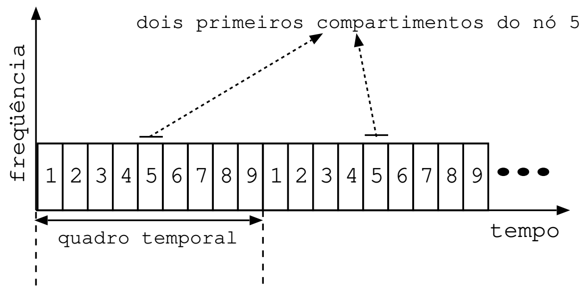


Figura 2.2: Multiplexação por divisão de tempo

Diferentemente do protocolo TDMA, o FDMA divide o canal no domínio da freqüência. Assim, os N nós que acessam o enlace teriam um canal com capacidade igual a $\frac{R}{N}$. Neste caso, como cada nó utiliza uma faixa de freqüência exclusiva, os nós podem transmitir ao mesmo tempo, sem que haja colisão. Este segundo esquema é ilustrado na Figura 2.3.

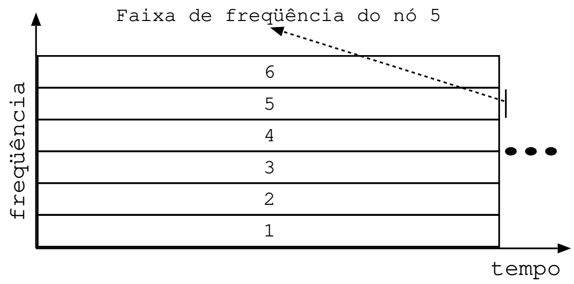


Figura 2.3: Multiplexação por divisão de freqüência

Ambos os protocolos acima descritos apresentam a vantagem de não haver possibilidade de colisão, já que o acesso ao meio é pré-definido. Além disso, cada nó que deseja utilizar a rede, vai encontrar disponibilidade de uma fração de $\frac{R}{N}$ do canal para efetuar suas transmissões.

Em contrapartida, como cada nó possui um canal de comunicação dedicado, sempre que um nó ficar ocioso, aquela fração de canal reservada a ele não poderá ser utilizada por nenhum outro nó e, por isso, será desperdiçada. Assim, um nó tem direito apenas à fração de $\frac{R}{N}$ do canal para transmitir, mesmo que ele seja o único nó ativo.

Outro protocolo de acesso ao meio por divisão de canal é o CDMA (Acesso Múltiplo por Divisão de Códigos) [27]. Nesta abordagem, é atribuído um código único para cada nó conectado ao enlace.

Conceitualmente, cada nó, quando desejar transmitir seus bits, deverá codificá-los utilizando seu código CDMA exclusivo. Cada bit enviado é codificado multiplicando-se o bit pelo código que muda com uma taxa (conhecida como *chip rate*) mais rápida que a taxa de seqüência de dados original.

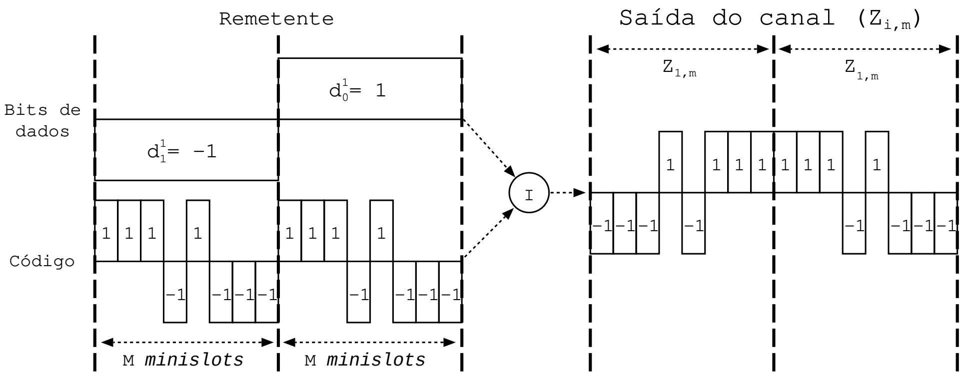


Figura 2.4: Codificação CDMA, onde M=8

Considere que cada bit de dados requer um *slot* de tempo para ser transmitido no i-ésimo *slot*. Cada *slot* é subdividido em M *minislots* conforme ilustra a Figura 2.4, onde M é a quantidade de valores usados no código. Na Figura 2.4, M é igual a 8, contudo, na prática, o valor de M pode ser bem maior. A seqüência de valores c_m, $m = 1, 2, ..., M$ é o código usado pelo transmissor onde $c_m = -1$ ou $+1$, por motivos de simplificação de cálculos, o bit de dados 0 é representado por -1.

Como mostrado na Figura 2.4, em I, o i-ésimo bit codificado $Z_{i,m}$ a ser transmitido será, então, o resultado da multiplicação do bit d_i^s pelo m-ésimo valor do código c_m através da equação: $Z_{i,m} = d_i^s c_m$, onde s indica o remetente em questão.

No caso de uma única transmissão, o receptor pode recuperar os dados transmitidos, como mostrado na Figura 2.5, em II, através da seguinte operação: $d_i^s =$

$\frac{1}{M}\sum_{m=1}^{M} Z_{i,m}c_m$.

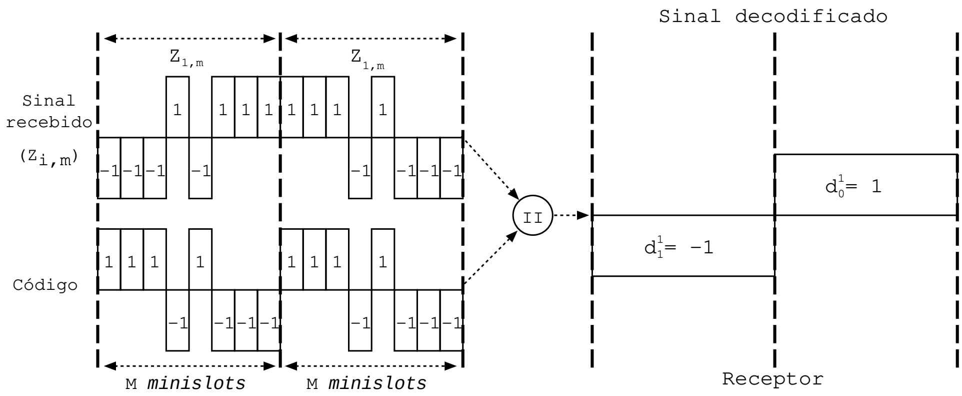


Figura 2.5: Decodificação CDMA

Utilizando este método para a codificação dos dados a serem transmitidos e escolhendo os códigos CDMA cuidadosamente [27], de forma que todos os códigos dos nós sejam ortogonais, ou seja, o produto interno destes código seja nulo, mesmo que vários usuários transmitam simultaneamente, será possível para um receptor que conheça o código de um determinado transmissor, decodificar o sinal recebido. Na presença de múltiplos nós, cada remetente calcula suas transmissões codificadas Z^s como visto anteriormente, mas o valor recebido pelo receptor no m-ésimo intervalo de tempo é, na verdade, a soma dos bits transmitidos por todos os N remetentes durante o intervalo de tempo:

$$Z^*_{i,m} = \sum_{s=1}^{N} Z^s_{i,m} \tag{2.1}$$

Assim, é possível que este receptor decodifique o i-ésimo bit de dados transmitido pelo remetente s da seguinte forma:

$$d^s_i = \frac{1}{M}\sum_{m=1}^{M} Z^*_{i,m}c^s_m \tag{2.2}$$

A utilização desta técnica permite, claramente, que diferentes nós transmitam simultaneamente, sem haver necessidade de divisão de tempo ou divisão de freqüência, e ainda sim é possível que um receptor decodifique corretamente os bits codificados pelo transmissor.

2.2.3 Protocolos de Revezamento

Os protocolos de revezamento têm como objetivo agregar as vantagens dos protocolos de acesso aleatório e dos protocolos de divisão de canal. Quando somente um nó tem dados para transmitir, ele usa toda a capacidade de transmissão. Quando N nós desejam transmitir, então, cada um deles obtém uma vazão de aproximadamente $\frac{R}{N}$, onde R é a capacidade total do canal e N é o número de usuários.

Esta classe de protocolos possui inúmeras variantes, mas as principais são: (1) o protocolo de passagem de permissões, (2) o protocolo de *polling* e (3) o protocolo de *polling* distribuído.

No protocolo de passagem de permissões, um quadro especial que indica a permissão para transmitir (*token*) é passado de nó a nó obedecendo uma lista, por exemplo, circular. Se um nó tiver quadros para transmitir, ele deve esperar o recebimento do quadro de permissão e, só então, pode transmitir. Caso um nó receba a permissão e não tenha quadros para enviar, ele simplesmente passa a permissão ao próximo nó da lista.

Já o protocolo de *polling* define um nó mestre que controlará todo o processo de acesso ao meio. Para isso, o nó mestre possui uma lista circular de todos os nós conectados à rede e envia seqüencialmente, a cada um deles, um quadro de controle que atribui a permissão para transmitir. Após enviar a permissão de transmissão ao primeiro nó, informando quantos quadros este nó pode transmitir, o nó mestre esperará até que a transmissão termine e, então, enviará uma nova permissão de transmissão ao próximo nó da lista.

Finalmente, o protocolo de *polling* distribuído funciona através de reservas e, para isso, o tempo é divido em $M \leq N slots$.

Um período, chamado de período de disputa, é formado por N *slots*, onde N é o número de nós conectados à rede. Como mostrado na Figura 2.6, se o nó i deseja transmitir, ele transmitirá um bit 1 durante o i-ésimo *slot* de disputa. Como somente o nó i tem permissão de transmitir no i-ésimo *slot* de disputa, não haverá colisão. Ao término do tempo de N *slots*, todos os nós terão um mapeamento de quais nós fizeram reserva durante o período de disputa e podem, a partir deste momento, iniciar suas transmissões de acordo com as reservas previamente feitas. Este período de transmissão de quadros de dados termina assim que todos os nós terminarem suas transmissões e, então, o processo será reiniciado. Assim, se um nó gera um quadro durante o período de transmissões, ele deve esperar o próximo período de disputa para fazer a sua reserva e só poderá transmitir efetivamente o quadro no próximo período de transmissões.

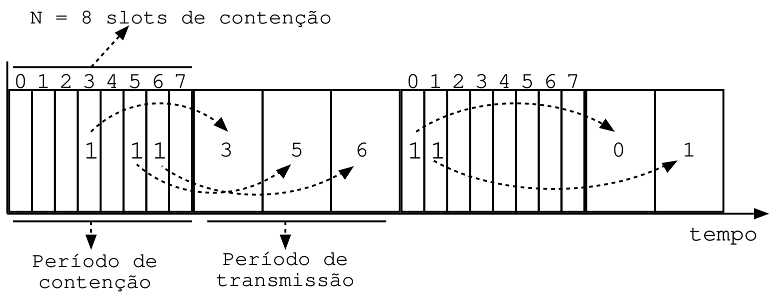


Figura 2.6: *Polling* distribuído (protocolo com reservas)

2.3 Redes de Acesso à Internet

Quando a Internet é analisada do ponto de vista da tecnologia utilizada para interligar os usuários finais à infra-estrutura da rede, ou seja, a um roteador conectado à Internet, podem ser identificadas, basicamente, três categorias: (1) redes de acesso institucionais, (2) redes de acesso residenciais (3) e redes de acesso móveis.

Embora essas categorias não sejam rígidas, já que existem redes que empregam várias tecnologias, essa classificação é interessante para facilitar o entendimento da estrutura global da Internet.

Nas redes de acesso institucionais, normalmente uma rede local é usada como meio de ligar os usuários pertencentes a empresas ou universidades ao roteador que provê acesso à rede mundial. Outras tecnologias menos comuns como redes ATM (*Assinchronous Transfer Mode*), FDDI (*Fiber Distributed Data Interface*) ou enlaces de satélite também podem ser utilizadas.

Redes de acesso residenciais utilizam, principalmente, modems que aproveitam o sistema telefônico como meio de se conectar à Internet. Estes modems tipicamente utilizam o protocolo PPP para se conectarem ao servidor *dial-up* pertencente ao provedor de serviços. As principais desvantagens do acesso discado são: a limitação da taxa de transmissão nominal de dados das linhas analógicas, que pode chegar, tipicamente, a até 56kbps; e a indisponibilização da linha telefônica enquanto a conexão estiver ativa.

Devido a essas limitações, tecnologias como: (a) o ISDN (*Integrated Services Digital Network*) que introduziu a capacidade de transmissão digital de dados, através de linhas telefônicas digitais, à taxas da ordem de 128kbps e (b) o ADSL (*Assimetric Digital Subscriber Line*) vem sendo introduzidas para permitir o acesso residencial de alta velocidade. No ADSL o enlace de comunicação é dividido em três faixas de freqüência: um canal de até 8Mbps na direção do usuário, um canal de até 1Mbps na direção da rede e um canal telefônico comum. Assim, além de oferecer conexão de banda larga à Internet, o usuário não fica com a linha telefônica ocupada. Uma outra tecnologia, recentemente disponibilizada, através do DOCSIS, será discutida em detalhes na seção 2.3.1.

Por fim, as redes de acesso móvel aproveitam a infra-estrutura já disponível de algumas redes de telefonia celular, para conectar à Internet, usuários que utilizam equipamentos como um laptop, uma agenda digital ou um celular de segunda ou terceira geração. Os principais padrões que estão sendo desenvolvidos para disponibilizar este tipo de serviço são o GPRS (*General Packet Radio Service*) [28], que implementa suporte a aplicações IP sobre redes celulares GSM (*Global System for Mobile*) e o CDMA2000, que também disponibiliza suporte a aplicações IP, mas sobre redes celulares IS-95. Como o CDMA2000 é foco deste trabalho, uma apresentação mais detalhada será feita na seção 2.3.2.

2.3.1 O DOCSIS

O DOCSIS é um padrão que foi desenvolvido para ser implementado sobre a tecnologia HFC (*Hybrid Fiber Coax*), e para tornar possível que usuários de uma rede convencional de TV à cabo (CATV) tenham um canal de acesso para enviar dados ao provedor de serviço. Várias aplicações podem ser implementadas sobre essa tecnologia, como *pay-per-view*, telefonia e acesso de banda larga à Internet.

Sendo o primeiro padrão desenvolvido para disponibilizar acesso de alta velocidade à Internet, através das redes de comunicação já implantadas pelas empresas de TV à cabo, o padrão DOCSIS foi iniciado como um projeto da organização MCNS (*Multimedia Cable Network System*). Em março de 1998, o DOCSIS versão 1.0 foi aprovado como padrão ITU-T e a versão 1.1 deste padrão, aprovada em 1999, trouxe funcionalidades adicionais para prover QoS.

Arquitetura do Sistema

A principal limitação da tecnologia HFC é a de que a rede foi inicialmente projetada para difusão de sinais de TV suportando, para isso, apenas a comunicação unidirecional (*simplex*) no sentido operadora-usuário. Para suportar o padrão DOCSIS, a arquitetura da rede HFC, mostrada na Figura 2.7, foi modificada através da introdução de dispositivos especiais chamados CMTS (*Cable Modem Termination System*) nos *hubs*, para que estes sejam utilizados como ponto de terminação dos protocolos das camadas 1 e 2 executados entre os usuários finais e o *hub*.

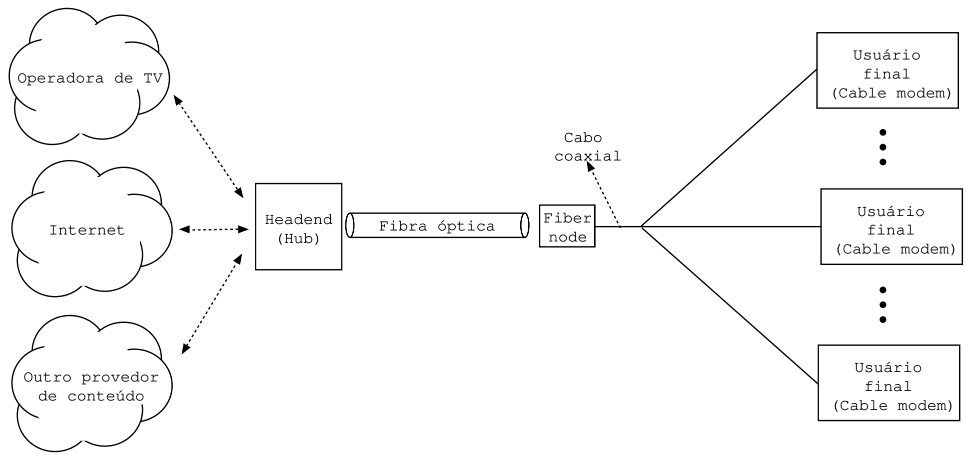


Figura 2.7: Visão geral da arquitetura do HFC

As redes HFC possuem várias características que as tornam similares, do ponto de vista da arquitetura, a algumas redes de acesso residencial, com enlace sem fio, à

Internet. Como exemplo, pode-se considerar uma rede tipicamente disponibilizada em condomínios constituídos por grandes edifícios. Nestes ambientes, uma antena (CMTS), é instalada no topo do edifício para funcionar como ponto de acesso ao provedor de Internet. Para isso, esta antena possui um enlace sem fio dedicado, de banda larga, que a conecta diretamente ao provedor de acesso e, como mostrado na Figura 2.8, distribui esta conexão a todos os nós do edifício, através de conexões convencionais à cabo.

As estruturas das redes das Figuras 2.7 e 2.8 possuem várias características em comum [2] que fazem com que utilizem o mesmo protocolo de acesso ao meio, o DOCSIS.

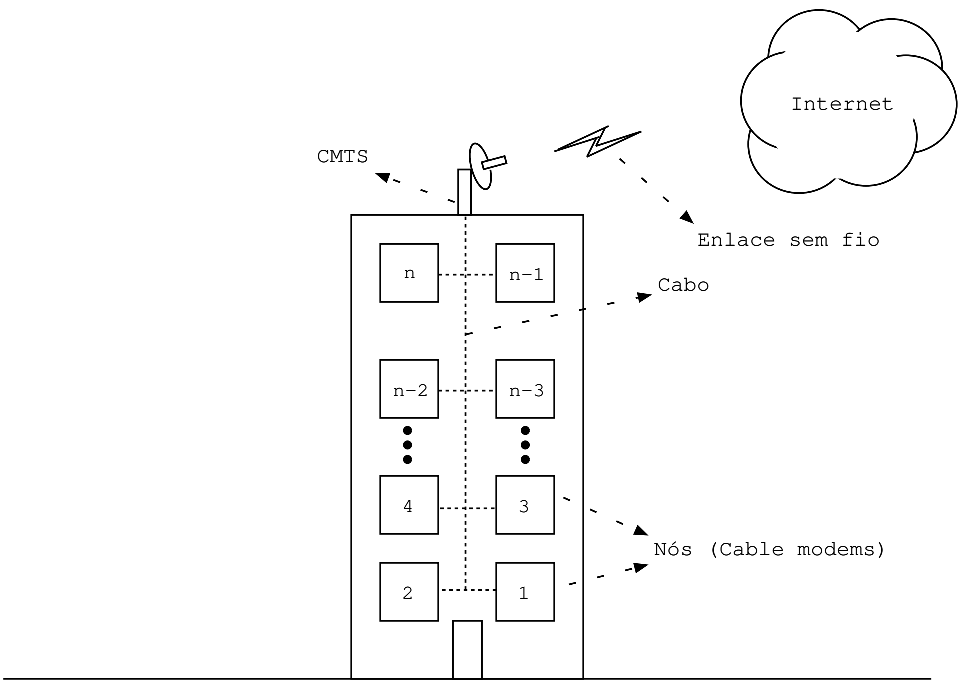


Figura 2.8: Rede de acesso sem fio à Internet

Um exemplo das similaridades entre essas duas arquiteturas diz respeito aos tipos de enlaces utilizados em cada canal de comunicação. O canal de descida (*downstream*), que representa a conexão no sentido antena-usuários, possui um enlace ponto-multiponto, e o canal de subida (*upstream*), correspondente à conexão no sentido usuários-antena, é um enlace multiponto-ponto, equivalente a um barramento. Por ser compartilhado, o canal de subida está sujeito a colisões e necessita de um esquema de acesso ao meio. Além disso, as capacidades dos canais de comunicação de subida e descida são assimétricas, onde o canal de descida possui capacidade bem superior à capacidade do canal de subida. Finalmente, ambas as arquiteturas podem ser implementadas em distâncias bem maiores do que as definidas para as redes locais, o que torna ineficiente o uso de protocolos de acesso ao meio como Aloha ou CSMA/CD.

Camada de Enlace e o Acesso ao Meio

O DOCSIS utiliza FDMA para dividir o enlace em dois canais de comunicação: (1) o canal de descida utiliza o intervalo de freqüência entre 550 e 750 MHz e é, então, dividido em subcanais de 6MHz (FDMA). As transmissões, nestes canais, são realizadas através de multiplexação por divisão de tempo. Todos os *cable modems* recebem os dados enviados pelo CMTS, mas os descartam, a menos que o endereço MAC coincida com o endereço do mesmo e (2) o canal de subida utiliza a faixa de freqüência de 5 a 42MHz e um esquema combinado de acesso aleatório e acesso com reserva. Para isso, este canal é dividido em subcanais de até 3200kHz que, enfim, são divididos em *slots* de tempo (TDMA).

Existem vários modos de operação do mecanismo MAC, no canal de subida, do DOCSIS: o modo de operação normal, o acesso isócrono, acesso de varredura de solicitação periódica (*periodic request polling*) e acesso imediato.

No modo de operação normal, o canal de subida corresponde a um fluxo contínuo de *minislots*. Um *minislot* é uma unidade básica de oportunidade de transmissão no canal de subida. No DOCSIS, o tamanho do *minislot* depende do esquema de modulação adotado e deve ser múltiplo de 6,25 μs. O mecanismo de *minislots* aumenta a eficiência de transmissão, uma vez que números apropriados de *minislots* de dados podem ser alocados para diferentes solicitações, tornando a alocação de banda bastante flexível. Existe um número máximo de *minislots* de dados por quadro. Caso a demanda por estes *minislots* seja maior do que o número disponível, transmissões serão agendadas para o próximo quadro.

O CMTS coordena o acesso ao canal de subida e atribui o uso do mesmo em um mapa de alocação de banda. Alguns *minislots* do quadro de subida são denominados *minislots* de disputa e são utilizados para envio de mensagens de solicitação de banda. Outros *minislots* denominados *minislots* de dados são utilizados para transmissão de dados de usuários e podem ocupar múltiplos *minislots* contígüos.

Para reduzir o desperdício do uso da banda, devido às colisões, os modems inicialmente enviam solicitações de banda, sujeitas a colisões, ao CMTS. O CMTS, então, agenda as solicitações e, através do canal de descida, passa o mapa de alocação de banda aos modems, como mostrado na Figura 2.9. No período agendado, os dados podem ser enviados, no canal de subida, livres de colisão.

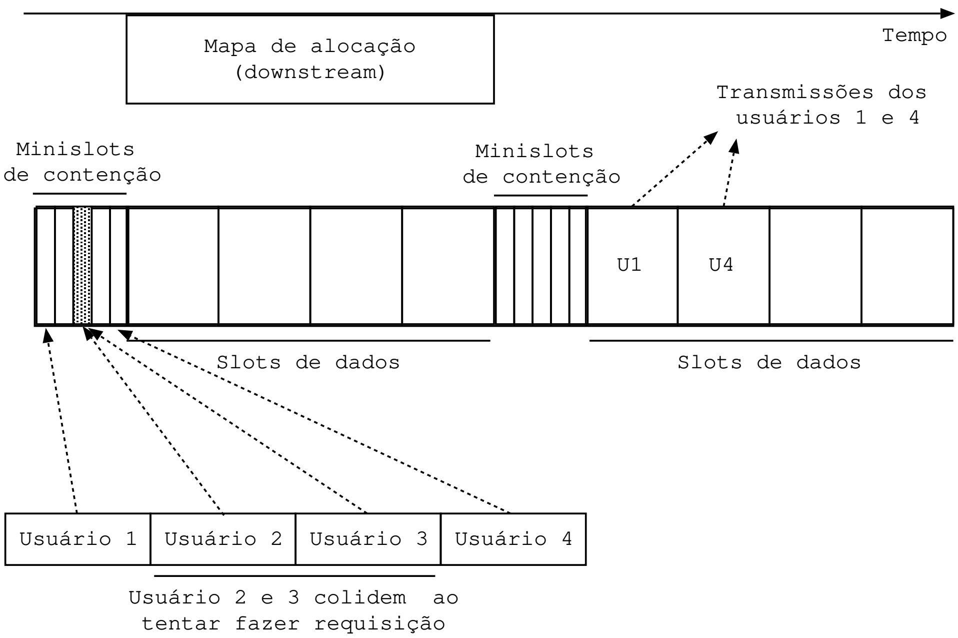


Figura 2.9: DOCSIS: modo de acesso normal

No cabeçalho das unidades de dados, há um campo opcional denominado *Extended Header Field*, que pode ser utilizado para solicitações *piggybacking*. Através destas solicitações, os modems podem requerer banda adicional sem passar pelo processo de disputa. Com isto, o atraso de acesso pode ser minimizado, uma vez que as solicitações não precisam ser efetuadas através dos *minislots* de disputa. Além disso, o número destes *minislots* de solicitações diminui.

No modo de acesso isócrono, que é ideal para o transporte de dados com taxa de bits constante, apenas um pedido por *minislots* de dados é enviado no início de uma sessão e a atribuição será válida até que esta sessão termine.

Já no modo de acesso de varredura de solicitação periódica, as solicitações de banda podem ser feitas sob demanda, e não em cada quadro, como é feito no modo normal de operação. São oferecidas varreduras de solicitações periódicas, onde o tempo entre varreduras é bem maior do que o tempo de um quadro. Este modo de operação é adequado para dados com taxa de *bit* variável.

Por fim, no modo de acesso imediato, o modem pode enviar os dados diretamente, sem fazer solicitação de banda. Neste caso, os dados serão transmitidos pelo *cable modem*, se ainda houver banda disponível após o atendimento de todas solicitações. O usuário sabe previamente quantas transmissões serão feitas no próximo quadro através do mapa de alocação de banda que antecede cada quadro. Este modo de acesso possui atraso reduzido, no entanto podem ocorrer colisões.

Com relação a QoS, o DOCSIS define seis categorias de serviço: UGS (*Unsolicited Grant Service*), UGS-AD (*Unsolicited Grant Service with Activity Detection*), rtPS (*real time Polling Service*), nrtPS (*non-real-time Polling Service*), BE (*Best Effort*) e CIR (*Committed Information Rate*).

Na categoria UGS, o CMTS provê uma quantidade fixa de dados a intervalos periódicos para o fluxo UGS. Esta categoria é adequada para aplicações de videoconferência e vídeo sob demanda (acesso isócrono). No entanto, nesta categoria pode haver desperdício de banda quando o fluxo de dados está inativo.

Na categoria UGS-AD, ao inicio de uma sessão, o serviço é similar ao da categoria UGS. Contudo, o CMTS utiliza um detector de atividade para verificar se o fluxo de dados está inativo. Quando o fluxo está inativo, o CMTS reverte o modo de acesso para varredura de solicitações periódicas. As categorias rtPS e nrtPS utilizam o modo de acesso de varredura de solicitações periódicas e são adequadas para aplicações VoIP e FTP de banda larga, respectivamente. No entanto, a categoria nrtPS recebe poucas oportunidades de solicitações quando a rede está no estado de congestionamento.

A categoria de serviço BE pode utilizar os modo de acesso normal, *piggybacking* ou imediato. É adequada para aplicações *telnet*, FTP ou WWW.

Por fim, a categoria CIR é proprietária, pode ser definida de diversas formas, dependendo do fabricante. Uma configuração bastante utilizada para CIR é semelhante ao serviço rtPS mas com uma taxa mínima garantida.

Para atender aos requisitos de QoS destas diferentes categorias de serviço, o CMTS deve possuir um método de controle de admissão e um algoritmo de agendamento (*scheduling*) para cada uma delas.

O DOCSIS adota o algoritmo *Bynary Exponential Backoff* no processo de resolução de colisões durante a disputa. Unidades de dados denominadas *Data Backoff Start* (DBS) e *Data Backoff End* (DBE) são utilizadas para indicar o tamanho inicial e máximo da janela de *backoff*. O algoritmo tem início com um tamanho de janela de *backoff* de 2^{DBS}. A seguir, é escolhido, de forma aleatória, um número x entre 0 e 15 e o tamanho da janela de *backoff* é modificado para 2^x. O modem não faz nenhuma solicitação até chegar a janela selecionada por este número. Se a disputa não é bem sucedida, a janela de *backoff* é incrementada por um fator de 2, de forma sucessiva até o tamanho máximo da janela (2^{DBE}) ou até que o número de colisões chegue a 16, onde a solicitação é abortada.

2.3.2 O CDMA2000-1X

Tecnologias como a do *CDMA2000 1X* tornaram possível que usuários de uma rede celular convencional tivessem canais de acesso, de até 153kbps, para enviar ou receber dados do provedor de serviço, através da conexão de seu equipamento local (*laptops* ou aparelhos celulares) à rede 2,5G.

O padrão CDMA2000 1X (IMT-2000 - *International Mobile Telecommunications in the year 2000*, ITU) tem sido implantado em diversos países e possui um grande potencial de crescimento. Este padrão é um dos primeiros produtos da família IMT-2000, consistindo em um primeiro passo da evolução do padrão IS-95B de segunda geração, e utiliza a tecnologia CDMA de banda larga para satisfazer algumas das necessidades levantadas pelo IMT-2000. Em [6], é feita uma descrição detalhada desta evolução, onde são apresentados seus objetivos finais. As vantagens deste sistema são: a oferta de serviços Internet em regiões remotas e mercados emergentes, a compatibilidade retroativa com os padrões CDMA anteriores e a possibilidade de implementação gradativa em áreas selecionadas, onde capacidade adicional e serviços de banda larga são necessários.

Embora este padrão seja baseado na tecnologia de comutação de circuitos, só é permitido que usuários utilizem canais de alta velocidade por um curto tempo, chamado modo de operação em rajada [20]. Desta maneira, procura-se minimizar o problema de se ter um canal de alta velocidade dedicado a um usuário que fica a maior parte do tempo ocioso.

O acesso ao meio é disponibilizado através de códigos CDMA, que constituem canais de comunicação. As taxas nominais de cada canal, seja direto (estação base-usuário) ou reverso (usuário-estação base), podem ser de 9.6, 19.2, 38.4, 76.8 ou 153.6kbps. Canais diretos e reversos são pedidos separadamente e a entidade do sistema responsável por receber e processar os pedidos de atribuição de canal é a estação base.

O CDMA2000 1X pode prover dois tipos de serviço com relação à mobilidade do usuário [29]: (1) o serviço *Simple IP* (SIP), que permite, aos clientes móveis, o acesso a serviços de rede baseados no protocolo IP, sobre um enlace CDMA sem fio, sem dar suporte à continuidade da conexão e à manutenção do endereço IP, quando ocorre mudança de células e (2) o serviço *Mobile IP*, que permite a manutenção da conectividade com a rede quando os usuários mudam de uma célula para outra, conservando o mesmo endereço IP.

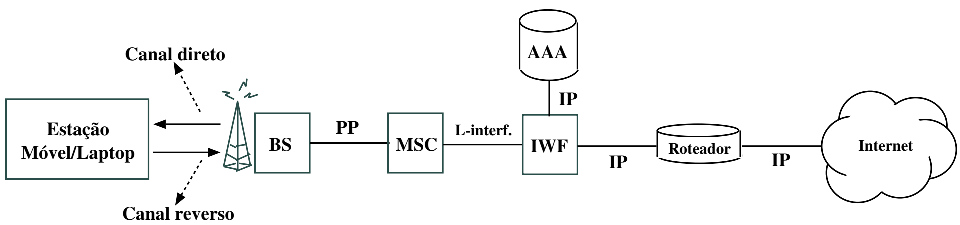


Figura 2.10: Arquitetura do sistema CDMA2000 1X

Como mostrado na Figura 2.10, a arquitetura do sistema de dados [30, 31]

CDMA2000 1X é constituída pela estação base CDMA (BS), pelo MSC (*Mobile Switching Center*), pelo servidor IWF (*InterWorking Function*) e pelo servidor AAA (*Authentication, Authorization and Accounting*).

A interconexão destes dispositivos é implementada através dos diferentes padrões ilustrados na Figura 2.11: o terminal móvel e a BS usam enlaces CDMA sem fio (IS-95); a BS e o MSC utilizam o *packet pipe* (PP), que consiste em um grupo de canais de comunicação disponibilizados através de um enlace T1; o MSC e o IWF fazem uso do *L-interface*, que define um canal T1 dando suporte ao *Frame-Relay*; o IWF, o AAA e um roteador IP são interligados por uma rede *Ethernet*.

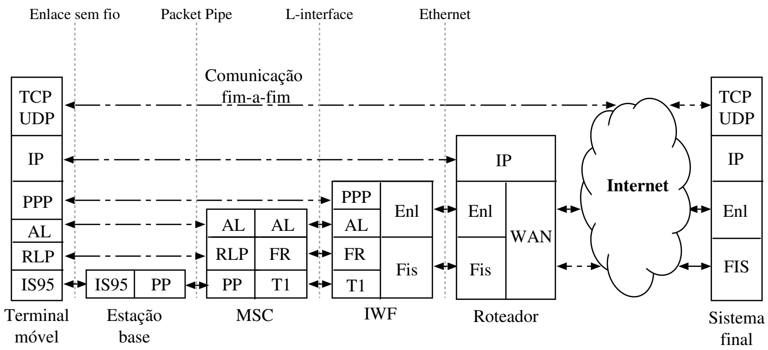


Figura 2.11: Modelo de camadas do CDMA2000 1X

No modelo apresentado na Figura 2.11, a camada de adaptação (AL - *Adaptation Layer*) é utilizada principalmente para sinalização e faz parte do nível de enlace. Esta camada é mostrada à parte por conveniência, com o objetivo de destacar o fato de o tráfego de sinalização entre o usuário e a estação base usarem o protocolo MAC, discutido mais adiante.

Camada de Enlace e o Acesso ao Meio

Para dar suporte a aplicações IP, o esquema de acesso ao meio do CDMA2000 1X possui novas funcionalidades [32, 33] que buscam melhorar a utilização do sistema diante deste tipo de aplicação, em que o tráfego é gerado em rajadas. Transmissões que possuem este comportamento resultam em um número muito maior de transições entre os estados da camada de enlace, quando comparados ao serviço convencional de telefonia. Os estados da camada de enlace serão apresentados posteriormente nesta seção.

Alguns tipos de canais são utilizados, principalmente, para fins de sinalização do controle de acesso ao meio. O protocolo de sinalização entre a estação móvel e a estação base é o SRBP (*Signaling Radio Burst Protocol*), que faz parte da camada AL. Além disso, certos canais também suportam mensagens curtas de dados (SDB - *Short Data Burst*), como as geradas pelos serviços de mensagens de texto entre usuários (SMS - *Short Message Service*) ou de verificações de números de cartão de crédito. Por isso, é importante definir a função de cada um deles antes de abordar o protocolo MAC:

- REACH - O *Reverse Enhanced Access Channel* é um canal de acesso aleatório utilizado pelos clientes móveis para iniciar comunicações com a estação base. Canais REACH podem ser usados através do modo de acesso básico ou do modo de acesso com reserva (detalhados adiante). São exemplos de mensagens enviadas por este canal: tráfego de sinalização SRBP contendo pedidos/liberações de canais dedicados de dados ou reservas de canais RCCCH, e mensagens SDB;

- RCCCH - O *Reverse Common Control Channel* é usado também para tráfego de sinalização SRBP e mensagens SDB. Contudo, este canal só pode ser utilizado mediante a reservas. Estas reservas são feitas através de requisições dos usuários, enviadas pelo REACH; ou pela estação base a fim de agilizar o envio de respostas às mensagens de sinalização SRBP originadas pela mesma;

- FCPCH - O *Forward Common Power Control Channel* é utilizado pela estação base para prover informações de controle de potência, dando suporte às transmissões do REACH ou do RCCCH;

- FCACH - O *Forward Common Assignment Channel* é reservado para que a estação base envie alocações de RCCCHs em resposta às requisições de acesso com reserva, feitos através do REACH. Existe um par de canais FCACH e FCPCH para cada canal REACH do sistema operando no modo de reserva;

- FCCCH - O *Forward Common Control Channel* é um canal direto, de controle, usado para transmissão de mensagens de sinalização da estação base aos usuários, bem como para enviar SDBs. Para cada canal FCCCH suportado pelo sistema, podem haver até 31 REACHs e 31 RCCCHs.

O controle de acesso para os canais REACH e RCCCH é dividido em dois modos de operação denominados Modo Básico de Acesso e Modo de Acesso com Reserva [32]. O fator que determina o modo de acesso a ser usado pela estação móvel é o tamanho da mensagem (de sinalização ou SDB) a ser enviada. Caso a mensagem seja maior que um limite definido, o modo de acesso com reserva é usado. Caso contrário, é usado o modo de acesso básico.

No modo básico de acesso, ilustrado na Figura 2.12, o cliente móvel escolhe aleatoriamente um dos canais REACH e transmite um preâmbulo no próximo *slot*. O preâmbulo é formado por um grupo de *bits* de controle transmitidos a uma potência elevada para ajudar a estação base a receber a mensagem que o sucede. A estação base, através do efeito de captura [32], tem a capacidade de decodificar duas transmissões em um REACH, a menos que dois ou mais clientes iniciem no mesmo *slot*.

Após terminar uma transmissão, o cliente aguarda por um ACK no FCCCH. Se esta confirmação não chegar em um determinado período de tempo, o cliente realiza uma espera ditada pelo algoritmo *binary exponential backoff*, aumenta a potência e reinicia a tentativa de transmissão, utilizando o mesmo método. Um ACK só é enviado pela estação base quando a mensagem é recebida e decodificada com sucesso.

Já o modo de reserva utiliza canais adicionais como o FCACH, o FCPCH e o RCCCH. Como indicado na Figura 2.13, assim que o cliente resolve transmitir, um RAP (*Reservation Access Probe*) é enviado pelo REACH. O RAP é composto por um preâmbulo e um cabeçalho que contém campos como a taxa de transmissão prevista, o tamanho da mensagem que será transmitida e o número de identificação da requisição do usuário, entre outros. Em seguida, o cliente monitora o FCACH para esperar pela mensagem de atribuição de canal. Após recebê-la, o cliente pode interromper o monitoramento do FCACH e começar a transmitir sua mensagem

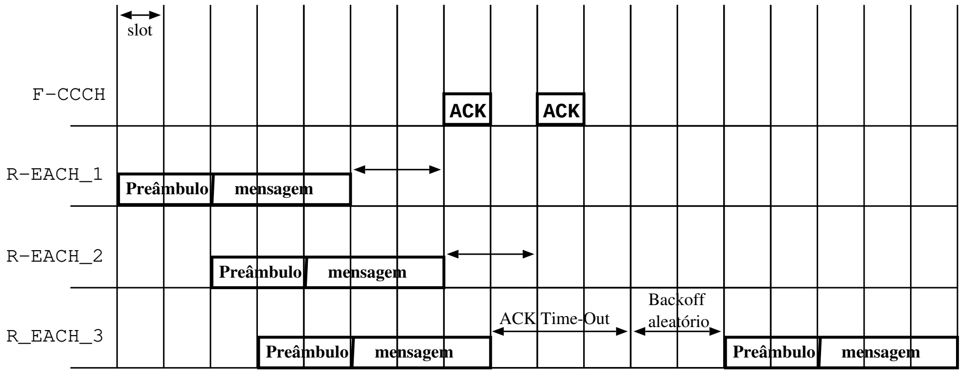


Figura 2.12: Modo básico de acesso

através do RCCCH a ele atribuído, utilizando o canal FCPCH para controle de potência. A transmissão da mensagem pelo RCCCH deve ser precedida de um preâmbulo a fim de ajudar a estação base a receber o sinal corretamente.

Caso nenhuma mensagem de atribuição de canal seja recebida depois de um certo tempo, o cliente realiza uma espera de acordo com o algoritmo *binary exponential backoff*, aumenta a potência e envia novamente o RAP usando o mesmo método explicado anteriormente.

A atribuição de um RCCCH é válida somente para uma tentativa de transmissão. Por isso, se depois de ter efetuado a transmissão de uma mensagem, nenhum ACK é recebido pelo FCCCH dentro de um tempo pré-determinado, um tempo aleatório é esperado (*binary exponential backoff*) até que todo o processo seja novamente reiniciado. Apesar de não haver disputa pelo canal RCCCH, interferências e atenuações no sinal, por exemplo, podem causar o descarte da mensagem recebida pela estação base.

O modo de acesso com reserva ajuda a economizar recursos, tais como: potência da estação móvel e atrasos. Isso acontece porque o cliente transmite apenas um RAP no canal REACH e a mensagem em si é enviada pelo canal RCCCH a ele reservado. Além disso, o controle de potência ajuda a reduzir os efeitos da interferência do sinal, o que diminui o número de mensagens descartadas.

Até o momento foram descritos dois modos de acesso para transmissão de uma única mensagem de dados (SDB). No entanto, também está previsto, no padrão

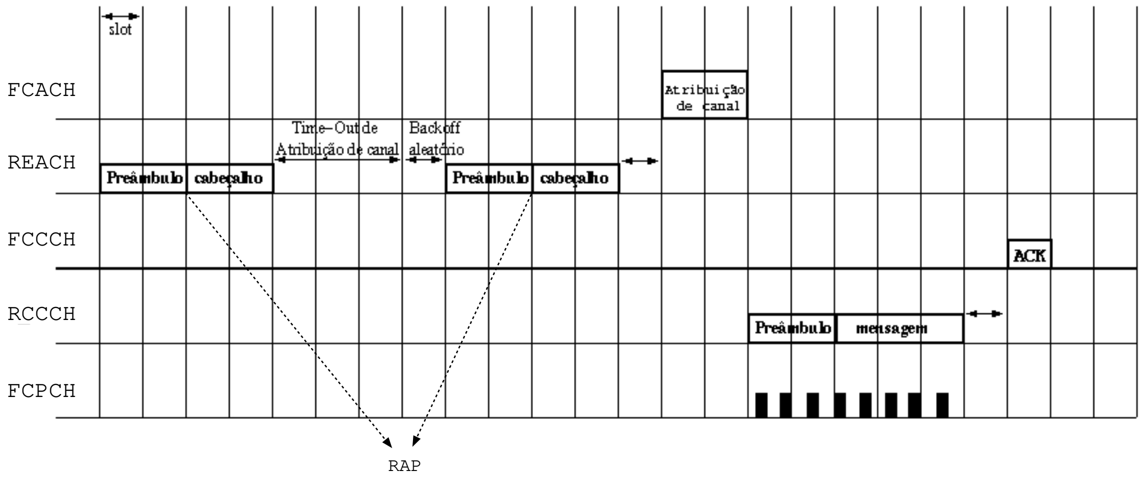


Figura 2.13: Exemplo de acesso com reserva, o primeiro RAP falha

CDMA2000 1X, um serviço para o envio de um conjunto de mensagens que prevê a alocação de um ou mais canais dedicados. Chamaremos este serviço de serviço de dados.

No serviço de dados, os canais REACH e RCCCH são usados para envio de mensagens de sinalização SRBP conforme descrito anteriormente. Além destes canais, são usados dois outros canais para transmissão de dados: o canal fundamental (FCH - *Fundamental Channel*) e o canal suplementar (SCH - *Supplemental Channel*). A alocação e liberação dos canais, fundamentais e suplementares, é governada pelo controle de admissão de chamadas.

O serviço dedicado é orientado à conexão. Uma conexão de camada de enlace (LLC - *Link Layer Connection*) é criada quando o cliente deseja transmitir dados. A LLC é estabelecida entre o terminal móvel e o IWF. Podemos ver na Figura 2.11 que uma conexão de camada de enlace requer o estabelecimento de conexões em outras camadas inferiores.

Uma conexão deste tipo é ativada quando o cliente faz uma chamada para transmissão de dados. No estado ativo, os dados do usuário podem ser enviados e recebidos. Neste estado o cliente possui um canal de comunicação de rádio freqüência (IS95), enlace RLP e uma conexão PPP com o IWF. Existe também, um circuito FR entre o MSC e o IWF reservado para este cliente.

Para manter o controle sobre o tempo que uma chamada pode ficar no estado ativo sem efetuar transmissões, um temporizador de inatividade, denominado RLP *inactivity timer*, é mantido no MSC e também no equipamento cliente. Se não houver atividade de transmissão de dados por um determinado período de tempo, a conexão passa para o estado dormente. Neste estado, não existem mais o enlace RLP nem o circuito FR. Contudo, a conexão LLC continua aberta, embora esteja no estado de dormência. Tanto o IWF quanto o cliente mantêm o estado da conexão PPP. A LLC pode ser reativada do estado dormente quando há chegada de dados no IWF ou quando o cliente deseja enviar mais dados, através da troca de sinalização de controle.

Logo que o cliente ou a estação rádio base finaliza a chamada de dados, a LLC é fechada. A partir deste momento, o servidor IWF não tem nenhuma informação de conexão LLC associada ao respectivo cliente. Neste estado (fechado) também não há conexão PPP entre os dois pontos (cliente - IWF). Conseqüentemente, nenhum serviço de dados pode ser oferecido ao usuário. Para isso, uma nova conexão da camada de enlace deverá ser aberta.

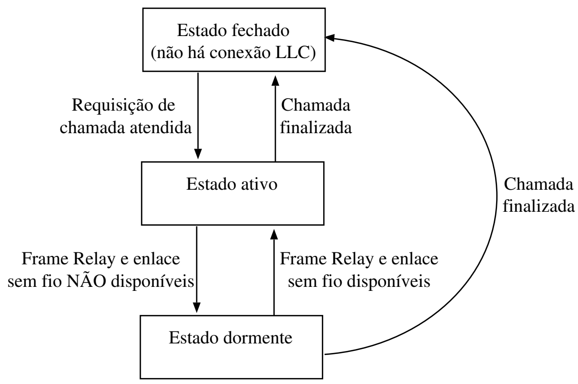


Figura 2.14: Diagrama de estados da camada de enlace do CDMA2000

O processo de abertura de uma conexão da camada de enlace se inicia quando a estação base recebe, por intermédio do SRBP, uma requisição de um canal fundamental. Esta é encaminhada ao MSC, que tem a função de identificar o seu tipo (voz ou dados) e comutar serviços orientados a circuitos, através do controle de admissão. Se a chamada aceita, pelo controle de admissão, for de dados, uma conexão RLP é aberta entre o cliente e o MSC. O MSC passará a intermediar a comunicação entre o usuário e o IWF através da abertura de um circuito virtual FR com o IWF. Esta estrutura permite que uma conexão PPP seja estabelecida entre a estação móvel e o servidor IWF, como ilustrado na Figura 2.11.

Através da conexão PPP, o IWF receberá do cliente móvel um pedido de abertura de uma sessão IP. Um servidor de autenticação (AAA) é consultado e, de acordo com a resposta, a conexão é estabelecida ou negada. Quando a conexão é estabelecida, o servidor IWF busca um endereço IP em um conjunto de endereços disponíveis e o aloca para o cliente. Este IP será, então, o endereço do cliente para a Internet. Assim, o servidor IWF passa a encaminhar os pacotes vindos desta conexão para o roteador a ele conectado. Informações de contabilidade dos clientes IP, como duração e número de chamadas, também são coletadas pelo IWF e passadas ao servidor de contabilidade (AAA).

Um canal fundamental entre o cliente e a estação base é alocado assim que uma conexão LLC é aberta. Este canal é *half-duplex* e permanece dedicado ao cliente enquanto a LLC estiver no estado ativo. Se, em um dado momento, a fila de transmissão do cliente (fila reversa) ou da estação base (fila direta) ultrapassar um valor estipulado, uma requisição de canal suplementar (SCH - *Supplemental Channel*), direto ou reverso, é gerada. Este evento resulta na alocação de um SCH na taxa mais alta que o sistema suporta naquele momento. Durante todo o período em que um canal suplementar permanecer dedicado ao cliente (modo de operação em rajada), ambos os canais FCH e SCH são utilizados para transmissão de dados e de mensagens de sinalização. O SCH ficará alocado ao cliente até que sua fila se esvazie ou até que o tempo máximo de rajada[20] seja alcançado.

FCHs tem prioridade sobre SCHs. Caso um cliente solicite uma conexão da camada de enlace e não haja recursos suficientes no sistema para realizar a abertura desta conexão, um SCH será retirado de um cliente, com o intuito de liberar recursos para efetuar o atendimento da requisição pendente. A estação base procura pelo canal suplementar de mais alta taxa atribuído para realizar esta tarefa. Caso a fila do cliente que perdeu um SCH ainda esteja acima do limite estipulado, outro pedido de SCH será gerado e, possivelmente, um SCH de taxa inferior será atribuído.

Controle de Admissão de Chamadas

Diversos métodos de controle de admissão de chamadas para o serviço de comutação de canais de dados [34, 35] do CDMA2000 estão disponíveis na literatura. Estes algoritmos possuem diferentes níveis de detalhamento e, geralmente, levam em consideração o monitoramento de variáveis como a carga e o nível de interferência no sistema, causado tanto por usuários da própria célula, quanto por usuários de células vizinhas. O padrão CDMA2000 1X, contudo, não define um algoritmo específico, deixando a seleção do método de controle de admissão por conta do fabricante do equipamento.

Neste trabalho, algumas hipóteses são consideradas de modo a obter um controle de admissão que não onere demasiadamente as simulações relativas aos experimentos descritos no capítulo 5. Estas considerações são apresentadas a seguir:

- A primeira suposição é a de que o sistema é formado por apenas uma célula;
- Assume-se que os usuários do sistema estão simetricamente distribuídos em volta da estação base [21]. Além disso, considera-se que o sistema possui usuários estacionários (sem movimento);
- Considera-se também que as potências de todos os outros usuários permanecem inalteradas diante da ativação de um canal de comunicação adicional [34];
- O tipo da antena considerada é omnidirecional [36].

O escalonamento de canais de dados considerado neste estudo é baseado na análise feita em [21], juntamente com os parâmetros disponíveis em [21, 20].

Como as suposições feitas para o ambiente de experimentos desta dissertação desconsideram interferências geradas por células vizinhas, o controle de admissão torna-se semelhante para os canais diretos e reversos.

Considere o $\frac{E_b}{N_t}$ como a qualidade de um canal de rádio CDMA entre o móvel e a estação base. Esta qualidade é dada pela relação entre a energia de cada *bit* de informação (E_b) e a densidade espectral de ruído (N_t), onde esta última é referida na literatura como um bom descritor genérico do ruído de um sistema. O padrão CDMA relaciona a qualidade de um canal de rádio com o FER (*Frame Erasure Rate*), isto é, o número de quadros descartados devido a presença de erros. Assim, se o nível $\frac{E_b}{N_t}$ aumenta, o FER diminui, e a qualidade de transmissão do sistema é melhorada.

O ponto chave a ser analisado para a alocação de um novo canal está na sensibilidade do receptor. Este conceito é entendido como a potência mínima requerida pelo receptor de um sinal para que uma determinada condição como por exemplo, um FER máximo, seja garantido.

Em [21], a sensibilidade do receptor é definida como uma função monotônica proporcional ao d_{min} requerido, à taxa de transmissão R do canal, à carga u da célula e ao piso de ruído no receptor FN_0W:

$$E\{S\} = d_{min}R\left\lfloor\frac{1}{1-u}\right\rfloor FN_0W \tag{2.3}$$

O d_{min} é a mínima qualidade do canal requerida pelo receptor e é escolhida de forma que o FER fique abaixo de um certo limite.

A carga u é definida como $\frac{N}{N_{max}}$ onde N é o número de canais ativos e N_{max} é a capacidade de pólo [37], ou seja, o número máximo de canais simultâneos teoricamente suportados caso os usuários possuíssem potência ilimitada para suas transmissões. Cada tipo de canal, FCH ou SCH, possui o parâmetro (R) correspondente à taxa nominal de transmissão de dados do canal.

A condição para que um novo canal de dados seja aceito é que a potência disponível Pot_{disp} que o transmissor pode usar para um canal subtraída de toda a atenuação A_{total} até o receptor seja maior do que a potência $E\{S\}$ ditada pela sensibilidade do receptor:

$$Pot_{disp} - A_{total} \geq 10 * log_{10}(E\{s\}) \tag{2.4}$$

onde

$$A_{total} = HL + fade + BVL + PL - AG + CL \tag{2.5}$$

e $10 * log_{10}(E\{s\})$ é usado para converter a potência $E\{s\}$ para decibéis.

Na equação 2.5, HL representa a atenuação causada pela absorção do sinal no corpo do usuário (dB), ($fade$) é o decaimento do sinal (dB) causado por mudanças nas características do caminho de propagação no tempo, (BVL) é a perda de sinal pela penetração em prédios ou veículos (dB), (PL) é a atenuação do sinal pela distância (*path loss*), (AG) é o ganho de sinal na antena (dB_i) e (CL) representa a perda de sinal nos cabos ($cableloss$) (dB).

Uma vez que a distância entre dois pontos é conhecida, existem várias maneiras de se calcular a perda de potência causada pela propagação no espaço [38, 39, 40] (*path loss*). Neste caso específico, a perda de sinal é calculada considerando propagação em espaço livre (*free space path loss*), já que as outras componentes da atenuação total do sinal são consideradas em 2.5. Este parâmetro foi ajustado para diferentes valores para realização de experimentos que definiram o cenário de avaliação de desempenho, como será apresentado no capítulo 5.

O algoritmo para alocação de um novo canal pode ser resumido nos seguintes passos:

1. Cálculo da sensibilidade do receptor $E\{s\}$;
2. Cálculo da atenuação do sinal até o receptor A_{total};
3. Se $Pot_{disp} - A_{total} \geq 10 * log_{10}(E\{s\})$ então o canal pode ser alocado.

Capítulo 3

A Ferramenta de Modelagem TANGRAM-II e as Novas Funcionalidades

NESTE capítulo é feita a descrição das principais características do TANGRAM-II, uma ferramenta que permite a construção de modelos matemáticos de um sistema e possibilita a sua solução, tanto pela utilização de métodos analíticos quanto por meio de simulação. As novas funcionalidades implementadas para a ferramenta neste trabalho também são apresentadas.

3.1 Introdução

O conjunto de ferramentas que forma o ambiente de modelagem do TANGRAM-II [7, 8] foi desenvolvido no intuito de fornecer suporte ao ensino e pesquisa, e permitir que, através de uma interface gráfica, pudessem ser criados e analisados modelos que reproduzem o comportamento de sistemas. A metodologia de descrição de modelos adotada pela ferramenta é orientada a objetos, assim como foi proposto em [41]. Como resultado da solução dos modelos descritos, pode-se destacar medidas de confiabilidade e desempenho.

O projeto deste ambiente de modelagem vem sendo continuamente desenvolvido, desde sua versão original [42], de 1991. Desde então, foram adicionados à ferramenta vários melhoramentos e funcionalidades [43] que posteriormente foram integrados em uma única interface gráfica desenvolvida em Java. No segundo semestre de 2000, o TANGRAM-II passou a ser gratuitamente distribuído pela Internet [16].

Entre os módulos que, atualmente, compõem o TANGRAM-II, destacam-se o Ambiente de Modelagem e Análise, a ferramenta para geração e coleta de estatísticas de tráfego [8, 44], o VivaVoz [45] e o Whiteboard [46].

3.2 Ambiente de Modelagem

A elaboração de um modelo utilizando o ambiente de modelagem, mostrado na Figura 3.1, é feita através de uma ferramenta de domínio público chamada TGIF - *Tangram Graphic Interface Facility*. Os objetos podem ser criados a partir de exemplos disponibilizados. Os objetos interagem entre si através da troca de mensagens e o seu estado interno é armazenado em estruturas chamadas de variáveis de estado. O comportamento de um objeto é ditado por eventos que possuem condições ligadas às variáveis de estado e, também, por ações causadas pela chegada de mensagens ou pelo disparo de eventos.

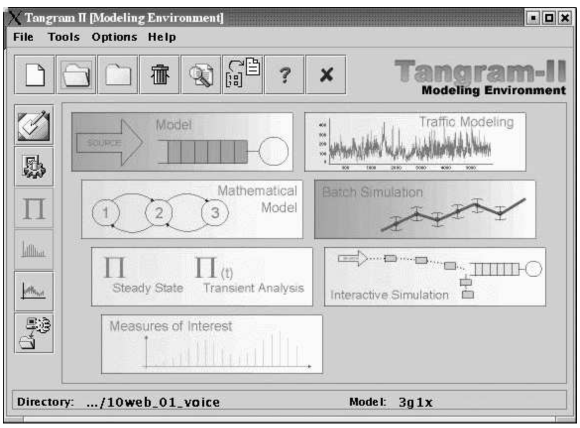


Figura 3.1: Ambiente de modelagem do TANGRAM-II

Quando uma variável de estado assume um valor que satisfaz a condição de um ou mais eventos, eles são habilitados e um disparo é agendado para cada evento de acordo com a distribuição estatística definida. Várias distribuições estão disponíveis na ferramenta, como a determinística, a uniforme, a exponencial, a erlangiana, a normal, a lognormal, a FBM, a FARIMA e a *file* que especifica que as amostras serão lidas a partir de um arquivo.

O evento só irá disparar se a sua condição continuar válida até o momento agendado para o disparo. Caso a variável de estado a ele ligada mude de valor e não mais satisfaça sua condição, o evento será desabilitado.

Quando um evento é disparado, uma ação relativa a ele é executada imediatamente. Uma ação pode resultar na modificação do valor de variáveis de estado do objeto e pode, também, gerar uma mensagem que possibilita, no paradigma da ferramenta, a interação imediata entre diferentes objetos. Uma ação também é executada quando uma mensagem é recebida. As mudanças nas variáveis de estado decorrentes de um evento ou do recebimento de uma mensagem, só alteram o estado dos objetos no final de uma ação. Assim, durante a execução de uma ação, o sistema pode passar por um conjunto de estados evanescentes. Estes estados não fazem parte do espaço de estados do modelo, já que estes podem ocorrer enquanto mensagens estiverem sendo trocadas por objetos ou enquanto variáveis de estado vão sendo alteradas no decorrer de uma ação. Após o tratamento de todas as mensagens, e a finalização da ação em execução, o sistema terá seu estado alterado para um outro estado, chamado de estado tangível.

Após a criação do modelo, ele pode ser resolvido para que possam ser obtidas as medidas de interesse desejadas. A solução de um modelo pode ser feita utilizando-se o módulo de soluções analíticas ou através do módulo de simulação.

3.3 Uso de Recompensas para Obtenção de Medidas de Interesse

Existem, basicamente, duas maneiras diferentes de se definir recompensas para um modelo: as recompensas de taxa e as recompensas de impulso (a definição formal de recompensas pode ser encontrada em [43]). A recompensa de taxa está associada aos estados do modelo. Cada vez que um estado do modelo é visitado, é acumulada uma recompensa proporcional ao tempo que o processo passou no estado em questão. Suponha que se deseja obter a medida de desempenho "utilização" de uma fila. Neste caso, pode-se definir uma recompensa que acumule o valor 1 sempre que a fila não estiver vazia. Esta recompensa contabilizará o tempo que a fila ficou ocupada. O seu valor acumulado, dividido pelo tempo total de simulação, representará a utilização da fila em questão.

Assim como é possível associar recompensas aos estados de um modelo, pode-se, também, associá-las às transições entre os estados. Estas últimas são denominadas recompensas de impulso. Através da utilização destas recompensas, diversas outras medidas podem ser obtidas. Por exemplo, pode-se medir o número de pacotes servidos em um sistema associando-se uma recompensa de impulso de valor 1 ao evento "Serviço". Toda vez que este evento ocorrer, o valor acumulado da recompensa é incrementado, indicando mais uma transmissão de pacote. Como as transições de estado ocorrem tanto no disparo de eventos quanto no recebimento de mensagens, as recompensas de impulso pode ser associadas a ambos.

No TANGRAM-II, cada objeto pode ter várias recompensas de taxa e/ou impulso definidas de forma que várias medidas de interesse possam ser obtidas. Os conceitos de recompensas, bem como sua utilização na ferramenta TANGRAM-II, são de vital importância para os modelos de avaliação de desempenho propostos neste trabalho.

3.4 Novas Funcionalidades Implementadas

Devido aos requisitos dos modelos de fonte de tráfego, aos princípios de funcionamento dos modelos propostos e à necessidade de representar os componentes apresentados no capítulo 4, tornou-se necessária a implementação de alguns recursos adicionais na ferramenta TANGRAM-II. Estas funcionalidades foram integradas à nova versão do ambiente de modelagem e fazem parte do conjunto de recursos disponíveis para a criação de modelos.

3.4.1 Modificação da Geração de Variáveis Aleatórias

Ao iniciar os primeiros experimentos com os modelos criados, onde foi necessário realizar simulações de longos períodos de tempo, foi possível observar um comportamento indesejável dos eventos gerados pelo simulador. Este tipo de comportamento era causado pela maneira que a ferramenta implementava a geração de variáveis aleatórias para o agendamentos de eventos.

O simulador mantém uma lista estática, construída apenas no início da simulação, que contém todos os eventos descritos no modelo criado pelo usuáro. Cada evento possui um indicador que informa se ele está habilitado, ou não, de acordo com o estado atual do sistema.

Na abordagem original, cada evento, que possuía condição habilitada, ganhava uma semente gerada a partir do *clock* da máquina. Cada evento, então, tinha um caminho amostral próprio definido pela sua semente inicial. A cada amostra gerada, o evento recebia uma nova semente para a geração da próxima amostra, como ilustrado na Figura 3.2.

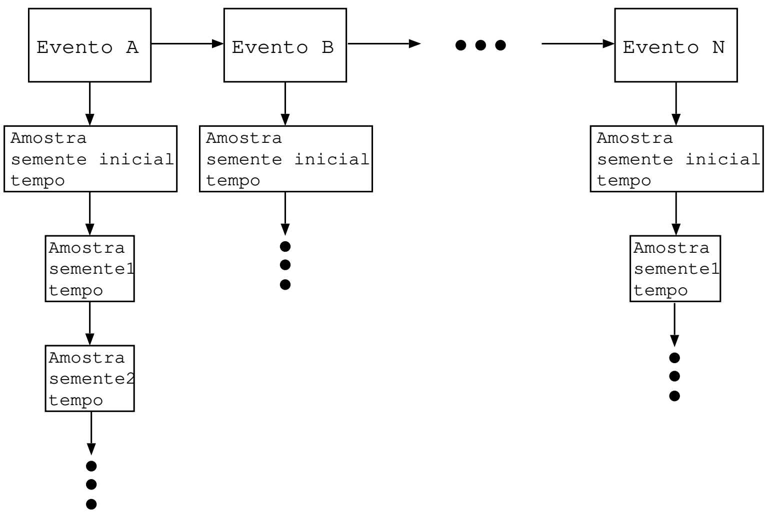


Figura 3.2: Arquitetura original da geração de variáveis aleatórias do simulador de eventos discretos

Quando um modelo possuía vários eventos, habilitados no início da simulação, as sementes, inicialmente escolhidas para estes eventos, acabavam tendo valores muito próximos uns dos outros. Como conseqüência, os primeiros disparos destes eventos eram gerados em sequência, resultando em correlação entre as amostras. Para que a correlação entre as amostras se tornasse insignificante e o sistema entrasse em estado estacionário, o tempo de simulação tinha que ser configurado para um valor consideravelmente maior.

Para solucionar este problema, uma nova abordagem foi adotada para a geração de variáveis aleatórias do simulador de eventos discretos, onde apenas uma semente é gerada a partir do *clock* da máquina. Os tempos de ocorrência de todos os eventos são gerados a partir de amostras obtidas de uma mesma semente inicial. Quando um primeiro evento gera um número aleatório, a semente modificada obtida, por padrão, como retorno da função de geração de números aleatórios é armazenada em uma estrutura de dados independente dos eventos.

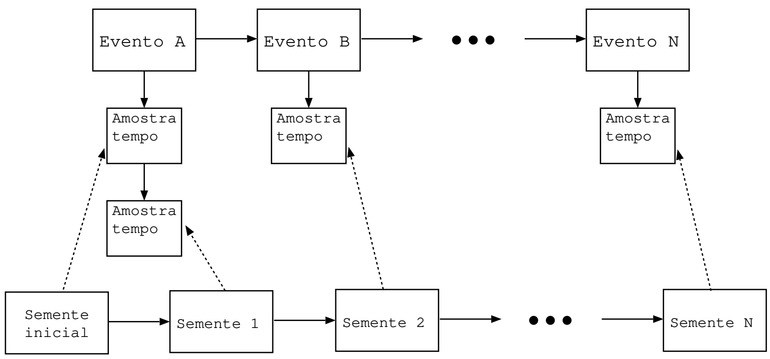


Figura 3.3: Nova arquitetura de geração de variáveis aleatórias do simulador de eventos discretos

Como mostrado na Figura 3.3, se um segundo evento fizer a chamada de geração de uma amostra, ele utilizará a semente previamente modificada e obterá uma terceira semente de retorno, a qual será utilizada pelo próximo evento a gerar uma amostra.

3.4.2 Funções de Distribuição de Probabilidade Adicionadas

Como será apresentado no capítulo 4, o comportamento de usuários *web* é melhor descrito quando são utilizadas variáveis aleatórias que possuem distribuição de cauda longa. Essas variáveis aleatórias têm sido utilizadas para modelar o comportamento de inúmeros eventos nos sistemas atuais, como: o tamanho de arquivos de páginas *web*; o tempo que um usuário acessando, a *web*, fica ocioso durante uma sessão; e o número de *bytes* em uma rajada TCP. Por isso, tornou-se interessante adicionar à ferramenta estas distribuições. Particularmente, foi necessário utilizar duas distribuições de probabilidade nos modelos: a Weibull e a Pareto.

No capítulo 5, são apresentadas as dificuldades encontradas na simulação de sistemas com eventos representados por distribuições de cauda longa. Estes problemas tornaram necessária a implementação de uma terceira distribuição: a Pareto truncada, onde o valor da amostra é truncado em um valor máximo definido, caso ela seja maior do que um limite estipulado. A distribuição Pareto truncada possui um parâmetro adicional que indica o limite superior da amostra a ser gerada.

A função de distribuição de uma variável aleatória Weibull é dada por:

$$F_X(x) = 1 - e^{-(x/\theta)^k}, \tag{3.1}$$

onde θ é o parâmetro de escala e k é o parâmetro da forma, e de uma variável aleatória Pareto é definida como:

$$F_Y(y) = 1 - \left(\frac{k}{x}\right)^{\alpha}, \tag{3.2}$$

onde α e k definem o formato e a inclinação da curva, respectivamente.

O método utilizado para a geração de variáveis aleatórias é o da transformada inversa [47]. Este método se baseia na seguinte proposição [47]: considere U uma variável aleatória uniforme (0,1). Para qualquer função de distribuição contínua F, a variável aleatória X definida como $X = F^{-1}(U)$, tem distribuição F. Assim, para a geração de uma amostra de distribuição Weibull onde a $F_X(x)$ é dada por 3.1, primeiramente é gerada uma amostra de variável aleatória uniforme u. Em seguida a amostra x será dada por $x = F_X^{-1}(u)$, ou seja,

$$x = \theta\left(-ln(u)\right)^{\frac{1}{k}} \qquad \theta, k > 0. \tag{3.3}$$

Usando um processo análogo, implementou-se a distribuição Pareto. Neste caso,

$F_Y(y)$ é dada por 3.2, onde, através da transformada inversa, a geração de uma amostra será dada pela equação:

$$y = \left(\frac{k}{u^{\frac{1}{\alpha}}}\right) \qquad \alpha, k > 0. \tag{3.4}$$

Para exemplificar a especificação de eventos, que utilizam essas distribuições, considere uma fonte ON-OFF, onde o tempo em ON possui distribuição Weibull com parâmetros θ e k. A definição do evento, que representa a transição entre os estados ON e OFF, é feita através da instrução: `event= On_Off( WEIB,` θ`, k )`. Supondo que a distribuição do tempo que o sistema permanece no estado OFF é dado pela Pareto truncada, com parâmetros α, k e j, onde j representa o valor máximo da amostra a ser gerada, o código deste evento será: `event= Off_On( TRUNCPAR,` α`, k, j )`. Neste último evento, se não houvesse a necessidade de truncar o valor das amostras, o quarto argumento seria omitido e o parâmetro `TRUNCPAR` seria modificado para `PAR`, resultando no código: `event= Off_On( PAR,` α`, k )`.

3.4.3 Funções Matemáticas Adicionadas

O modelo de canal de acesso do padrão CDMA2000 1X, apresentado no capítulo 4, considera grandezas relativas a potências de sinais, o que normalmente requer algumas funções matemáticas, como potenciação e logaritmos, para sua manipulação. No que diz respeito a funções de potenciação foram disponibilizadas duas variantes: a função *pow(arg1,arg2)*, que eleva o primeiro argumento ($arg1$) passado à potência do segundo argumento ($arg2$); e a função *sqrt(arg)*, que retorna o valor da raiz quadrada do argumento (arg) passado. Por exemplo, na instrução `y = pow( j,k )`, y assumirá o valor de j^k e, em `y = sqrt( x )`, y receberá o valor relativo à $\sqrt{x}$.

Para as funções de logaritmo, também foram implementadas duas variantes: o *log*, que retorna o logaritmo natural do argumento passado; e o *log10*, que retorna o logaritmo de base 10. Exemplos destas duas instruções são: `y = log( x )` e `y = log10( x )`. Em todos estes exemplos, x pode ser uma constante, uma variável ou uma expressão aritmética.

Capítulo 4

Modelos de Avaliação de Desempenho

NESTE capítulo serão apresentados o comportamento de aplicações *web* e de voz, destacando as principais características do tráfego gerado por estes tipos de usuário. Em seguida são apresentados os dois modelos de canal de acesso propostos: do mecanismo de acesso ao meio do protocolo DOCSIS e da rede de acesso do padrão CDMA2000 1X. Além disso, são comentados os principais trabalhos relacionados.

4.1 Aplicações *Web*

4.1.1 Introdução e Trabalhos Relacionados

O tráfego *web* continua a crescer e já representa mais de 50% do tráfego que, hoje, passa pela Internet. As características deste tráfego precisam, primeiramente, ser analisadas para que seja possível fazer o dimensionamento e o projeto de redes de suporte a este tráfego.

Na modelagem de tráfego em redes de computadores, é comum a utilização do processo de Poisson [48] para reproduzir a distribuição das chegadas de pacotes e de pedidos de conexões devido às suas propriedades atrativas do ponto de vista teórico. A modelagem de tráfego *web* é mais complexa por duas razões. Primeiramente, existem diversas versões de protocolos, navegadores e servidores que são usados para fornecer o serviço de acesso à *web*. Servidores *web* e navegadores *web* de diferentes fabricantes se comportam diferentemente e possuem parâmetros específicos. O protocolo HTTP (*Hyper Text Transfer Protocol*) possui diferentes versões que coexistem e interagem e, além disso, existem diferentes implementações do protocolo TCP. Em segundo lugar, o comportamento do usuário pode variar muito devido à natureza dinâmica deste ambiente. Os padrões de navegação de diferentes usuários são diversificados, um usuário pode propositalmente ou acidentalmente abrir vários navegadores e gerar pedidos para determinadas páginas ao mesmo tempo, ou pode abandonar a transmissão de uma página simplesmente clicando em outra ou no botão "voltar" ou "parar". O padrão HTML, *Hypertext Markup Language*, permite que um navegador busque múltiplas páginas de um mesmo servidor ou de vários servidores diferentes de uma só vez (*frames* e *Java-scripts*). O resultado de todo esse dinamismo é que não existe um único modelo que sirva como padrão de modelagem de tráfego *web*.

Para que seja possível entender as técnicas utilizadas para modelar o comportamento de um usuário *web*, é necessário que esteja bem claro o funcionamento básico do protocolo HTTP.

Uma página *web* é composta de vários objetos. Um objeto nada mais é que um arquivo, podendo ser uma página HTML, um clipe de áudio ou vídeo, uma imagem e um *applet*, entre outros. Normalmente uma página *web* possui um objeto principal que geralmente é um arquivo HTML e diversos objetos referenciados. Como exemplo, se uma página possui um arquivo base HTML (objeto principal) e nove imagens (objetos referenciados), então esta página tem dez objetos no total.

Ao clicar em um ponteiro de uma página ou digitar um endereço de uma determinada página, o cliente dispara o início de uma sessão. O navegador se conectará via TCP ao endereço indicado pelo usuário e solicitará o objeto principal. Uma vez que este objeto tenha sido recebido, ele será analisado e todos os objetos ali referenciados serão solicitados ao servidor. Após receber todos os objetos pertencentes a uma página *web*, o usuário normalmente passa alguns minutos observando o resultado na tela até que decide fazer a requisição de outra página. Este processo é então reiniciado.

Vários estudos já foram realizados com o intuito de caracterizar o tráfego gerado por aplicações *web*. Em [49] um conjunto de *traces* do tráfego gerado por usuários *web* é analisado. As evidências de autosimilaridade desse tráfego são apresentadas, e as possíveis explicações para esta característica de autosimilaridade são analisadas. Porém, a dificuldade de se criar modelos de simulação que reproduzem estes tipos de comportamentos não é abordada.

Em [50], os *logs* de quatro diferentes servidores *web* são analisados. Segundo este trabalho, o tráfego WWW possui características de autosimilaridade, com exceção do tempo entre requisições durante os momentos de maior ocupação dos servidores. A distribuição Log-normal é apontada como a mais apropriada para modelar a distribuição do tamanho dos objetos *web*. O canal de acesso é modelado através de uma fila M/G/1/P/S.

Já em [18], foi identificado que os tamanhos dos objetos principais e dos objetos referenciados de uma página e o tamanho das requisições *web* possuem distribuição Log-normal. Além disso, são sugeridas a distribuição Normal para o tempo entre chegadas dos objetos referenciados e para o número de objetos referenciados por um objeto principal e a distribuição Weibull para o tempo que os usuários passam analisando uma página recebida com sucesso.

Em [17], Deng monitorou o tráfego WWW entre o Laboratório da GTE e a Internet em duas sessões com duração de duas horas e meia na parte da manhã e duas horas à tarde e um modelo foi proposto a partir das estatísticas coletadas. O modelo de [17] será explicado detalhadamente, pois os experimentos realizados nesta dissertação foram baseados nele. Essa escolha foi feita no intuito de reproduzir o estudo feito em [19], constituindo em um primeiro passo para o então projeto de pesquisa entre a UFRJ, o CPqD e a Lucent e, também, porque o modelo de [17] se mostrou bastante adequado para representar o tráfego *web*.

	Distribuição	Parâmetros
Período ON	Weibull	$\theta = 90.01, k = 0.88$
Período OFF	Pareto	$k = 60.0, \alpha = 0.9$
Tempo entre requisições	Weibull	$\theta = 4.48, k = 0.5$

Tabela 4.1: Parâmetros das distribuições para o tráfego WWW

Em [17], o modelo usado para as requisições *web* é do tipo *ON-OFF*. Quando um usuário WWW clica em uma ligação de hipertexto, várias requisições WWW podem se seguir. A primeira transmite a requisição do usuário ao servidor, e as subseqüentes são geradas automaticamente pelo cliente de forma a obter os objetos referenciados. Quando a requisição do usuário é satisfeita, normalmente segue-se um intervalo para que os dados sejam lidos pelo usuário antes de uma nova requisição.

A identificação de um período *ON* (que pode incluir mais de uma requisição de usuário) depende de um valor arbitrado durante as medições. Em [17] requisições a intervalos inferiores a 60 segundos foram considerados como pertencentes ao período *ON*. Conseqüentemente um intervalo entre requisições maior que 60 segundos é considerado como *OFF*. Desta forma, foram identificadas duas distribuições para o modelo de um usuário WWW: (1) a distribuição Weibull foi usada para o período *ON* e (2) para representar o período *OFF* foi utilizada a Pareto. A Tabela 4.1 resume os valores dos parâmetros obtidos em [17] para cada uma das distribuições.

A seguir, será descrito o modelo baseado em [17] que foi elaborado no TANGRAM-II. A Figura 4.1 apresenta três objetos que constituem este modelo. Cada um deles é um servidor infinito usado para representar o tempo em ON, o tempo em OFF e o intervalo entre a geração de pacotes.

4.1.2 Comportamento do Usuário

O objeto *Initializer*, mostrado na Figura 4.1, possui a função específica de configurar a população de usuários ao início de cada simulação. Para isso ele é previamente alimentado com os parâmetros: (1) número (*MAX_ USERS*) total de usuários e (2) número (*INIT_ ACTIVE_ USERS*) inicial de usuários ativos, que podem ser

Parâmetro	Descrição	Tipo
Initializing	Indica a inicialização do modelo	Variável de estado
MAX_ USERS	População de usuários	Constante
INIT_ ACTIVE_ USERS	Usuários inicialmente ativos	Constante

Tabela 4.2: Variáveis e constantes do objeto *Initializer*

visualizados na Tabela 4.2. Quando a simulação é disparada, mensagens são enviadas aos objetos *Delay_ On*, *Delay_ Off* e *Ger. Pac.*, informando o número inicial de usuários no estado ON e no estado OFF. Após ter realizado sua tarefa, este objeto muda sua variável de estado *initializing* para o valor 0 a fim de terminar sua participação no sistema.

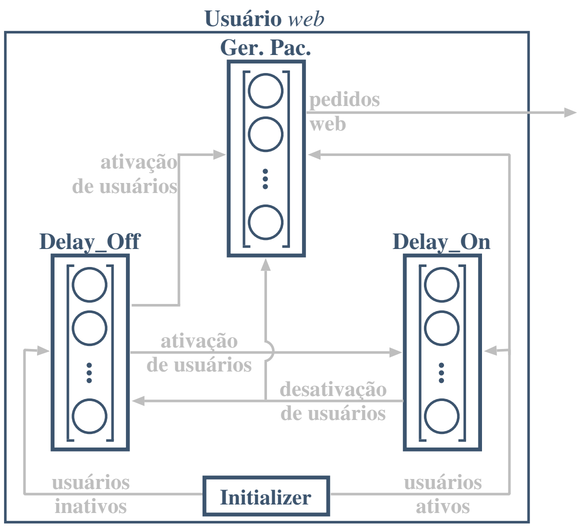


Figura 4.1: Visão geral do comportamento de um usuário *web*

A finalidade do objeto *Delay_ Off* é representar o comportamento de todos os usuários *web* do sistema que estão em OFF. Inicialmente este objeto recebe n mensagens do objeto *Initializer*, onde n é o número inicial de usuários no estado OFF. Para cada mensagem recebida, a variável de estado *Inactive_ Users*, descrita na Tabela 4.3 é incrementada e um evento que representa a transição, de um usuário, para o estado ON é escalonado através de clonagem de eventos [51]. Este evento possui distribuição Pareto truncada e sempre que ocorre um disparo, uma mensagem é enviada aos objetos *Delay_ On* e *Ger. Pac.* e o número de usuários em OFF (*Inactive_ Users*) é decrementado.

Parâmetro	Descrição	Tipo
Inactive_ Users	Número de usuários inativos	Variável de estado
OFFON_ SCALE	Parâmetro k da distribuição Pareto	Constante
OFFON_ SHAPE	Parâmetro α da distribuição Pareto	Constante
MAXIMUM	Valor máximo para as amostras da Pareto	Constante

Tabela 4.3: Variáveis e constantes do objeto *Delay_ Off*

Parâmetro	Descrição	Tipo
Active_ Users	Número de usuários inativos	Variável de estado
ONOFF_ SCALE	Parâmetro θ da distribuição Weibull	Constante
ONOFF_ SHAPE	Parâmetro k da distribuição Weibull	Constante

Tabela 4.4: Variáveis e constantes do objeto *Delay_ On*

Ao receber uma mensagem, o objeto *Delay_ On* utiliza, também, clonagem de eventos para escalonar novos disparos que representam transições do estado ON para o OFF. Além disso, a variável de estado *Active_ Users*, mostrada na tabela 4.4 , tem seu valor somado para representar a ativação de mais um usuário. A função deste objeto é a de reproduzir o comportamento de todos os usuários em ON e, sempre que ocorre o evento *OnOff*, que possui distribuição Weibull, uma mensagem é enviada aos objetos *Delay_ Off* e *Ger. Pac.* reportando a desativação de um usuário.

Quando o objeto *Ger. Pac.* recebe uma mensagem vinda do *Initializer* ou do *Delay_ OffOn*, um evento relativo à geração de pacotes (pedidos *web*) é escalonado e são incrementadas as variáveis de estado *Active_ Users* e *Packets*, que representam o número de usuários ativos e de eventos de geração de pacotes escalonados, respectivamente.

Para monitorar a desativação de usuários, a cada mensagem recebida de *Delay_ OnOff*, decrementa-se *Active_ Users*. Contudo, *Packets* não pode ser decrementada, já que não é possível retirar da fila de escalonamento um evento de geração de pacotes, devido às características do simulador utilizado. Para manter a consistência entre o número de usuários e o de pacotes sendo gerados, é utilizada uma aproximação onde, a existência de diferença entre as duas variáveis de estado em questão é verificada como explicado a seguir.

Parâmetro	Descrição	Tipo
Active_ Users	Número de usuários inativos	Variável de estado
Packets	Número de pacotes sendo gerados	Variável de estado
SOURCE_ SCALE	Parâmetro θ da distribuição Weibull	Constante
SOURCE_ SHAPE	Parâmetro k da distribuição Weibull	Constante
PACKET_ SIZE	Tamanho de cada pedido *web*	Constante

Tabela 4.5: Variáveis e constantes do objeto *Ger. Pac.*

Ao disparo do evento de geração de pacotes de distribuição Weibull, se *Packets* tiver valor maior que *Activer_ Users*, então *Packets* é decrementada. Além disso, em todas as ocorrências deste evento, uma mensagem é enviada a um outro objeto que represente o serviço da fila do canal de acesso, o que indica a geração de um pedido *web*. Modelos de canais de acesso serão apresentados posteriormente neste capítulo.

Essa aproximação para consistência entre as variáveis *Packets* e *Active_ Users* é válida, já que a taxa de geração de pacotes é consideravelmente maior do que a taxa de transição do usuário para o estado OFF, minimizando a ocorrência da diferença entre as duas.

Resumidamente, usuários são representados por fichas que circulam entre os objetos *Delay_ Off* e *Delay_ On/Ger. Pac.* da Figura 4.1, sendo que o tempo de permanência de uma ficha em cada objeto segue a distribuição correspondente. Uma vez em *ON*, é ativado o módulo *Ger. Pac.*, responsável pela geração de pacotes. Considera-se ainda, neste modelo, que um pacote é igual a 400 *bytes*, um tamanho típico de uma requisição *web*, medido em nossa rede local.

4.1.3 Tráfego *Web*

Uma vez que, utilizando o modelo descrito em 4.1.2, o comportamento do usuário *web* foi representado, é necessário descrever também o tráfego gerado por este usuário. Como a implementação desta modelagem depende de cada rede de acesso específica, será apresentada, nesta subseção, apenas o comportamento geral do modelo proposto, deixando o detalhamento do modelo específico de um padrão para subseção 4.4.2.

O modelo proposto é composto de três objetos adicionais aos que representam o comportamento do usuário *web*. Para facilitar a descrição que será feita, o modelo do comportamento do usuário foi representado por apenas um objeto e chamado de fonte *On-Off*, como pode ser visto na Figura 4.2. Ainda nesta figura, o objeto indicado por A representa a transmissão de pedidos *web* feita pela rede de acesso à Internet. Quando uma mensagem chega a este objeto, é verificado se este é o primeiro pedido de uma sessão (página principal) ou se é um dos pedidos referenciados por uma página principal e o pedido é, então, adicionado à fila, juntamente com o seu tipo. A taxa de serviço é uma função do tamanho médio do pedido e da taxa de serviço do canal. Sempre que um evento "transmissão de pacote" é disparado, uma mensagem relativa a este pedido é enviada ao objeto B.

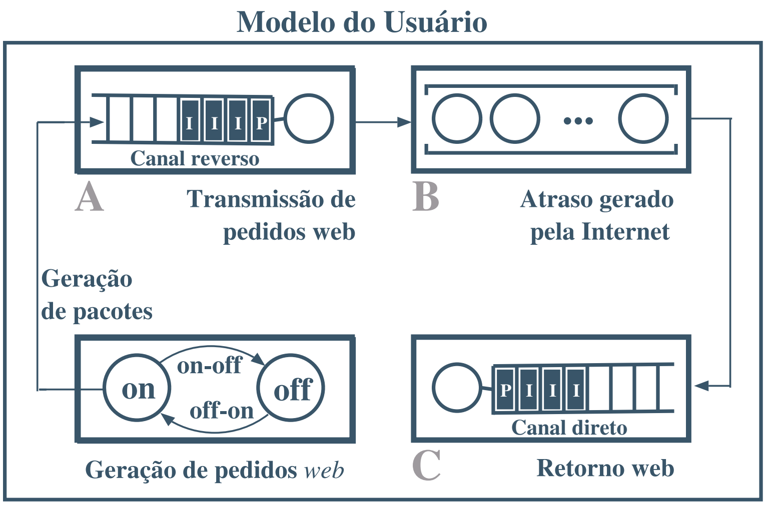


Figura 4.2: Modelagem do retorno do tráfego *web*

A função do objeto B é a de reproduzir o RTT (*Round Trip Time*) da Internet. Para isso, uma fila com infinitos servidores é implementada. Sempre que o evento relativo à chegada de resposta *web* ocorre, uma mensagem é enviada ao objeto C indicando a chegada de uma resposta ao respectivo pedido *web* anteriormente enviado. O atraso das respostas possui distribuição Gaussiana e os parâmetros, que foram baseados em [52], utilizados em sua implementação são o atraso médio e a variância deste atraso, como pode ser visto na Tabela 4.6.

O objeto C modela a transmissão da página *web*, através da rede de acesso, de

Parâmetro	Descrição	Tipo
MEAN_DELAY	Atraso médio (RTT) da Internet	Constante
DELAY_VARIANCE	Variância do atraso médio	Constante

Tabela 4.6: Parâmetros do objeto B

Parâmetro	Descrição	Tipo
Mean_Packet_Size	Tam. médio próximo objeto a ser enviado	Var. de estado
Packet_Size_Variance	Variância de *Mean_Packet_Size*	Var. de estado
MEAN_MAIN_OBJECT_SIZE	Tam. médio dos objetos principais	Constante
MAIN_OBJECT_VARIANCE	Variância de *MEAN_MAIN_OBJECT_SIZE*	Constante
MEAN_INLINE_OBJECT_SIZE	Tam. médio dos objetos referenciados	Constante
INLINE_OBJECT_VARIANCE	Variância de *INLINE_OBJECT_VARIANCE*	Constante

Tabela 4.7: Parâmetros do objeto C

volta ao usuário que a requisitou. Quando este objeto recebe uma mensagem de B, a fila de objetos *web* a serem transmitidos de volta ao usuário é incrementada. Para representar essas transmissões, foi implementado um evento que decrementa esta fila sempre que é disparado. Diferentes taxas foram usadas na implementação deste evento para reproduzir transmissões de dois tipos diferentes de objetos *web*: os principais e os referenciados, representados na Figura 4.2, por "P" e "I". Estas taxas são funções da banda disponível do canal de acesso e do tamanho do próximo objeto a ser transmitido. A Tabela 4.7 introduz as variáveis de estado e constantes utilizadas neste módulo.

Como pode ser visto em [18], os tráfegos gerados por diferentes tipos de requisições *web* são sensivelmente diferentes. Mais especificamente, a partir do momento em que um usuário *web* está ativo, a primeira requisição por ele gerada resultará no retorno de um arquivo HTML - o objeto principal. Todas as requisições *web* geradas a partir deste momento são decorrentes dos objetos referenciados pelo objeto principal.

Para tornar o modelo mais preciso, levou-se em consideração a diferença entre as distribuições do tamanho dos dois tipos de retorno *web* acima citados. Assim, o objeto C utiliza a distribuição Log-normal para modelar os tamanhos destes objetos, sendo que os parâmetros são diferentes para os dois casos em questão, como mostrado na Tabela 4.8.

	Distribuição	Média	Variância
Objeto principal	Log-normal	10710	25032
Objeto referenciado	Log-normal	7758	126168

Tabela 4.8: Parâmetros das distribuições para os tamanhos dos objetos *web* (*bytes*)

4.1.4 Problemas de Simulação com Distribuição de Cauda Longa

A simulação de eventos que possuem distribuições com cauda longa pode trazer efeitos adversos, como observado em [53], onde é mostrado que pode ser impossível atingir convergência para o estado estacionário. Se forem geradas amostras de variáveis aleatórias com este tipo de distribuição, pode-se observar que a maioria das amostras terá valores da mesma ordem de grandeza ou diferirão de poucas ordens de grandeza em relação à média. No entanto, um pequeno conjunto das amostras diferirá da média de muitas ordens de grandeza. Assim, à medida que se aumenta o tempo de simulação, o valor médio das amostras, para o tempo simulado, também é aumentado, já que aumenta-se também, a probabilidade de que sejam geradas amostras bem maiores do que todas as anteriores, fazendo com que não seja obtido um valor médio para o estado estacionário.

Este problema pode ser observado, por exemplo, em simulações contendo a distribuição Pareto com $\alpha < 1.7$, que é o caso do modelo utilizado neste trabalho, onde o tempo em *OFF* de usuários *web* é modelado como uma distribuição Pareto com $\alpha = 0.9$.

Para contornar este problema, Crovella e Lipsky [53] propuseram uma forma de escolher um valor para truncar a distribuição. Foi sugerido que o sistema real fosse observado durante um intervalo de tempo t e que a maior amostra obtida durante o tempo de observação (X_{max}) fosse usada para truncar o distribuição, ou seja, toda vez que for gerada uma amostra na simulação com valor superior ao da maior amostra obtida no sistema real (x_{max}), este valor será substituído pelo valor de X_{max}.

Este procedimento, sugerido por [53], foi aplicado nos experimentos realizados.

Para isso configurou-se o valor máximo do tempo em OFF (*MAXIMUM*), de acordo com o que foi observado em [17] e, sempre que um disparo do evento de transição de OFF para ON é sorteado para um intervalo de tempo acima do estipulado (*MAXIMUM*), então o disparo é escalonado para *MAXIMUM* segundos, como apresentado no apêndice A.

4.2 Aplicações de Voz

Para o estudo do comportamento das redes de acesso diante do tráfego gerado por aplicações de voz sobre IP, dois paradigmas de um mesmo modelo foram implementados. Nesta seção, estas duas abordagens serão apresentadas. Novamente, como a implementação do modelo específico para uma rede de acesso depende de cada padrão que se deseja estudar, os detalhes de cada modelo serão deixados para as subseções 4.3.2 e 4.4.2.

Inicialmente o paradigma de simulação de eventos discretos foi usado para a implementação do modelo de aplicação de voz. Em linhas gerais, este modelo é baseado em uma cadeia de Markov simples, e pode ser visto na parte esquerda da Figura 4.3.

Considera-se, neste modelo, que um usuário gera uma sessão de voz a cada 90 minutos e a duração média da sessão é de 3 minutos, ou seja, um usuário gera um tráfego de 0.0333 Erlangs.

Assim, o estado ON representa um usuário participando de uma sessão de voz sobre IP, onde o tempo médio do sistema neste estado foi considerado igual a 3 minutos, reproduzindo a duração média de uma sessão de voz. Após terminar uma sessão, o usuário permanece um tempo inativo, o que é representado através da passagem, do sistema, para o estado OFF. O tempo médio de permanência considerado para este estado foi de 90 minutos. Na Tabela 4.9, as variáveis de estado e constantes utilizadas na implementação deste esquema são apresentadas.

Se a fonte está ativa, são gerados pacotes de 160 *bytes* a intervalos constantes

Parâmetro	Descrição	Tipo
DATA_VOICE_INCOMING_RATE	Taxa de disparos do evento Off_On	Constante
DATA_VOICE_OUTGOING_RATE	Taxa de disparos do evento On_Off	Constante
TIME_BETWEEN_PACKETS	Tempo entre pacotes de voz	Constante
VOICE_PACKET_SIZE	Tamanho dos pacotes de voz	Constante
Status	Estado do usuário	Variável de estado

Tabela 4.9: Parâmetros utilizados na modelagem do usuário de voz

de 20ms, representando o tráfego gerado por áudio codificado em PCM (*Pulse Code Modulation*) [54]. Foi suposto que não há detecção de silêncio de forma que o tráfego gerado por uma sessão de voz será constante.

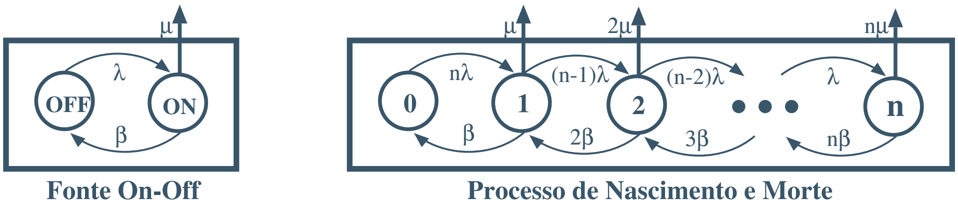


Figura 4.3: (a) Modelo de um único usuário de voz sobre IP e (b) modelo de n usuários de voz sobre IP

Considerando o modelo básico de um usuário de voz já apresentado, através da introdução de algumas modificações é possível representar vários usuários de voz sobre IP em um mesmo objeto, conforme descrito abaixo.

O modelo usado para n usuários de voz é um processo de nascimento e morte [48] onde o estado do processo representa o número de usuários ativos. Como mostrado na parte direita da Figura 4.3, as taxas λ_i de chegada e β_i de partida de usuários de voz ($i = 0, 1, 2, ..., n$) dependem do número i de usuários ativos. Já a taxa de geração de pacotes em um determinado estado é dada por $i\mu$, onde μ é a taxa de geração de cada usuário.

Porém, a realização de experimentos tornou-se computacionalmente inviável devido ao grande número de eventos e mensagens de controle geradas na simulação do canal de acesso do padrão CDMA2000 1X, que será apresentado posteriormente neste capítulo, juntamente à diferença entre a taxa de geração de pacotes de cada usuário de voz e a taxa de transição entre os estados. Este problema criou a necessidade de se tentar minimizar o custo computacional destes experimentos e motivou a criação de uma segunda implementação de usuários de voz, utilizando a técnica de simulação de fluido.

O ganho em termos de custo computacional do paradigma de fluido, em relação ao de eventos discretos se dá, basicamente, pelo número reduzido de eventos a serem tratados para que um mesmo sistema seja analisado. No entanto, existem alguns pontos que precisam ser considerados, como por exemplo o fato de que os eventos de fluido são mais custosos de serem tratados. No paradigma de fluidos, eventos podem ser escalonados, e por razões próprias desta abordagem pode ser que alguns destes venham a ser retirados da fila de eventos. O processamento requerido para verificar a existência de um evento escalonado, e retirá-lo da lista de eventos, ou até mesmo reescaloná-lo para um tempo posterior, onera o custo do tratamento de eventos. Entretanto, se o número de eventos processados em um ambiente de simulação de eventos discretos for suficientemente maior do que a quantidade de eventos a serem tratados no modelo de fluido, então o custo adicional de tratamento dos eventos de fluido não invalida os benefícios desta forma de simulação [9].

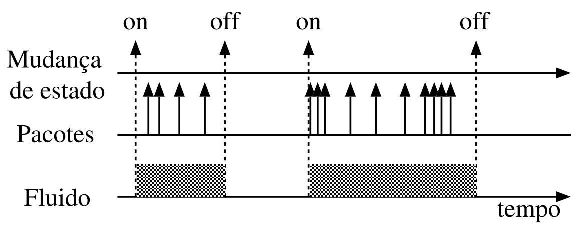


Figura 4.4: Fonte *ON-OFF* em um modelo de eventos discretos (pacotes) e em um modelo de fluido

No caso do modelo de usuários de voz, os eventos de geração de pacotes não são mais representados. Ao invés disso, reproduz-se apenas eventos que geram mudança na taxa de geração de dados ou de serviço do canal de acesso. Estes eventos são listados a seguir:

- Recebimento de mensagens relativas à atribuição de canais de comunicação;
- Recebimento de mensagens relativas à retirada de canais de comunicação;
- Ativação da fonte, ou seja, passagem do estado OFF para o ON;

- Desativação da fonte, ou seja, passagem do estado ON para o OFF;
- Fila (*buffer*) de transmissão se esvazia.

Considerando que os eventos de ativação e desativação de usuários são da ordem de minutos, que o recebimento de mensagens de atribuição/retirada de canais e os eventos de esvaziamento de fila são da ordem de segundos e que os eventos de geração de pacotes possuem freqüência de disparos da ordem de milisegundos, a diferença entre a taxa de eventos a serem processados, ao se usar simulação de eventos discretos ou de fluidos, é de magnitude suficiente para tornar o paradigma de fluidos atraente, quanto ao custo computacional.

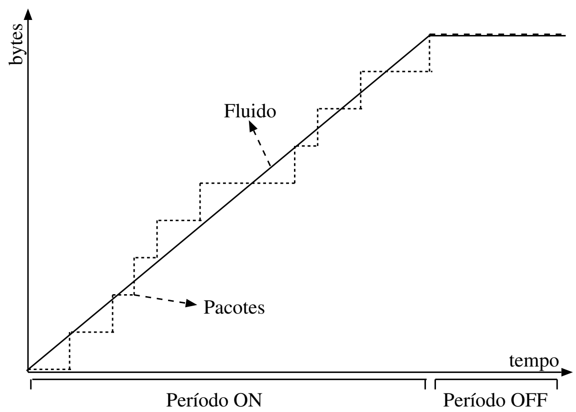


Figura 4.5: Tráfego acumulado em fontes *ON-OFF* e de fluido

No intuito de auxiliar o entendimento das abordagens em questão, é apresentada a Figura 4.4 onde, no momento em que o evento de transição do estado *OFF* para o *ON*, ambas as técnicas processam este evento. Porém, enquanto no paradigma de eventos discretos, os eventos de geração de pacotes são processados um a um, a abordagem de fluido só processa o próximo evento que resulte em mudança de alguma das taxas do sistema, como por exemplo, desativação da fonte. O fluxo de dados neste caso é representado por um líqüido que flui de um recipiente fonte a outro (i.e. fila do canal).

Já a Figura 4.5 apresenta o tráfego acumulado, durante um período de atividade da fonte, tanto para o modelo de fluido como para o de pacotes. Durante o período ON, o número de *bytes* acumulado cresce constantemente na abordagem de fluido.

A taxa τ de crescimento do recipiente que representa a fila é dada pela diferença entre a taxa λ de emissão de fluido e a taxa μ de escoamento pelo canal, ou seja, $\tau = \lambda - \mu$. Durante o período em *ON* vários eventos relativos a chegadas de pacotes são processados no esquema de pacotes, enquanto apenas o evento de transição para *OFF* está escalonado para a técnica de fluido.

4.3 DOCSIS

4.3.1 Introdução e Trabalhos Relacionados

Como apresentado na seção 2.3.1, a comunicação em redes HFC é assimétrica, sendo que a taxa do canal de descida é significativamente superior à do canal de subida. Além disso, a comunicação é do tipo multiponto-ponto para o canal de subida e ponto-multiponto para o canal de descida. O controle de acesso ao meio do canal de subida é do tipo acesso múltiplo e, portanto, sujeito a colisões.

Já o canal de descida não implementa controle de acesso, dado que o único nó da rede interessado em transmitir através deste canal é a estação base que transmite seus dados aos usuários por difusão.

Considerando as características gerais do sistema em questão, pode-se observar que para o dimensionamento do sistema quanto ao número de usuários suportados, o gargalo será o canal de subida, já que este será diretamente disputado por todos os usuários interessados em transmitir em um dado momento. Por isso, o modelo criado neste trabalho descreve o funcionamento do canal de subida.

Recentemente alguns trabalhos de avaliação do DOCSIS têm aparecido na literatura. O objetivo principal do trabalho de [55] foi o de verificar a influência do tamanho dos pacotes na vazão máxima atingida e no retardo de acesso do canal de subida. Dois tipos de tráfego foram considerados: isócrono e ON-OFF. Estes tráfegos entretanto não foram considerados em conjunto e não foi feito um modelo detalhado do tráfego de acesso WWW como no caso deste trabalho. Os resultados foram obtidos através de um simulador proprietário, e cada ponto levou 5-20 horas de simulação.

O trabalho de [3] comparou os protocolos DOCSIS e IEEE 802.14, usando a plataforma de simulação NIST ATM. O objetivo deste estudo foi o de analisar: os algoritmos de disputa e a transferência de tráfego ATM sobre os dois protocolos e IP também sobre ambos os padrões. Comparações são feitas entre estes cenários e o tráfego gerado tem distribuição Poisson apenas.

Em [56] o objetivo foi o de verificar a vazão máxima atingida quando tráfego isócrono é injetado no canal de subida do DOCSIS. Taxas de 8 a 64Kbps foram consideradas, sem nenhum outro tráfego. A influência do tamanho dos pacotes também foi estudada, e do *overhead* causado pelo encapsulamento. Um simulador proprietário foi utilizado.

O objetivo do trabalho de [19] foi o de estudar a adequação de protocolos usados em HFC para redes de acesso celular. Três tipos de tráfego foram considerados em separado, incluindo o tráfego *web*. As comparações, semelhantes a alguns dos trabalhos acima, levaram em conta o efeito do tamanho dos pacotes. Naquele trabalho, pelos resultados apresentados, o tamanho dos pacotes estão bem acima do tamanho típico das requisições de subida de um tráfego *web*. Uma explicação provável para o uso de tamanhos atípicos pode ser a opção de se considerar a influência da tamanho dos pacotes no desempenho do protocolo.

Nos trabalhos relacionados acima, não foi realizado um estudo do desempenho do DOCSIS quando submetido ao tráfego gerado por aplicações distintas. Hoje em dia, o tráfego gerado em uma rede de acesso é originado de diversas aplicações, portanto, é importante realizar um estudo que considere tráfegos com características diferentes.

No modelo proposto neste trabalho, são considerados em conjunto os tráfegos isócrono (voz sobre IP) e WWW.

4.3.2 Descrição do Modelo

Os objetos correspondentes ao modelo do canal de acesso podem ser vistos no lado direito da Figura 4.6, que representa o modelo global, incluindo a interação entre o modelo dos usuários e do canal de acesso. Os pedidos gerados pelo objeto *Ger. Pac.* são encaminhados inicialmente para o módulo *Fila_ de_ Requisições*, que simula o mecanismo de reserva de *slots*. Conforme indicado na seção 2.3.1, os *slots* do canal de subida são divididos em *slots* de disputa (CS) e de dados (DS). Foram consideradas três configurações para a relação CS/DS: 5/23, 8/22 e 15/21 [19]. Estas relações devem ser escolhidas de modo a que os quadros (*slots* de disputa + *slots* de dados) tenham o tamanho definido pelo padrão DOCSIS.

O módulo *Fila_ de_ Requisições*, ao receber uma mensagem relativa à geração de um pedido *web*, escolhe aleatoriamente um CS para inserí-lo. Para isso é incrementada uma das posições da variável de estado vetor *queue_ vector[CS]*, onde cada posição i ($i = 0, 1, 2, ..., CS - 1$) representa um *slot* de disputa.

Para modelar o término do período de disputa de um quadro, é usado o evento determinístico *TimeOut* de taxa $\lambda_t = (\frac{num\_cs*cs\_size}{link\_capacity})^{-1}$, onde o significado de cada variável desta equação é introduzido na Tabela 4.10. Ao disparo deste evento, o vetor *queue_ vector[CS]* é percorrido à procura de posições de valor $k > 1$, o que indica que k pedidos foram feitos no mesmo CS. Neste caso, k mensagens que reportam os pedidos colididos são enviadas ao módulo *Atraso_ de_ Requisições*, que será apresentado posteriormente. Caso contrário, e se houver espaço no período de transmissão, a fila *Queue* de serviço é incrementada com os *bytes* relativos ao pedido e a variável de estado *Gated* recebe o número de *bytes* a serem servidos no próximo quadro. Além disso, o *Token* que indica se o sistema está em período de disputa tem seu valor modificado para indicar o início do período de transmissão.

Quando *Token*= 1, o evento determinístico *Service* é habilitado. Este evento, de taxa $\lambda_s = (\frac{DS\_SIZE}{LINK\_CAPACITY})^{-1}\frac{1}{Gate}$, representa a fila de acesso para atendimento, onde o gerenciamento da banda é feito de forma a que o canal atenda alternadamente as reservas (*Queue*) e os dados que são livres do período de disputa (*Ugs_ Queue*),

Nome	Descrição	Tipo
NUM_ CS	Número de CS por quadro	Constante
CS_ SIZE	Tamanho do CS em *bytes*	Constante
DS_ SIZE	Tamanho do DS em *bytes*	Constante
LINK_ CAPACITY	Banda do canal de subida *bytes/s*	Constante
Token	Indica período de disputa/serviço	Variável de estado
Queue	Número de *bytes* na fila de serviço (*web*)	Variável de estado
Ugs_ Queue	Número de *bytes* na fila de serviço (voz)	Variável de estado
Gate	Número de *bytes* do próximo DS	Variável de estado

Tabela 4.10: Parâmetros do objeto *Fila_ de_ Requisições*

como o tráfego de voz, que tem prioridade sobre o tráfego WWW. Assim, de acordo com o protocolo, assume-se que uma vez estabelecida a reserva inicial para a aplicação de voz, esta permanece enquanto durar a chamada.

Quando o evento *Service* ocorre, primeiramente é servida a fila *Ugs_ Queue* e, se ainda houver tempo no período de transmissão, a fila *Queue* é servida. O Tempo máximo de serviço é fixo, por isso, caso as filas ainda tenham dados ao final de um período de transmissão, elas esperarão pelo serviço do próximo.

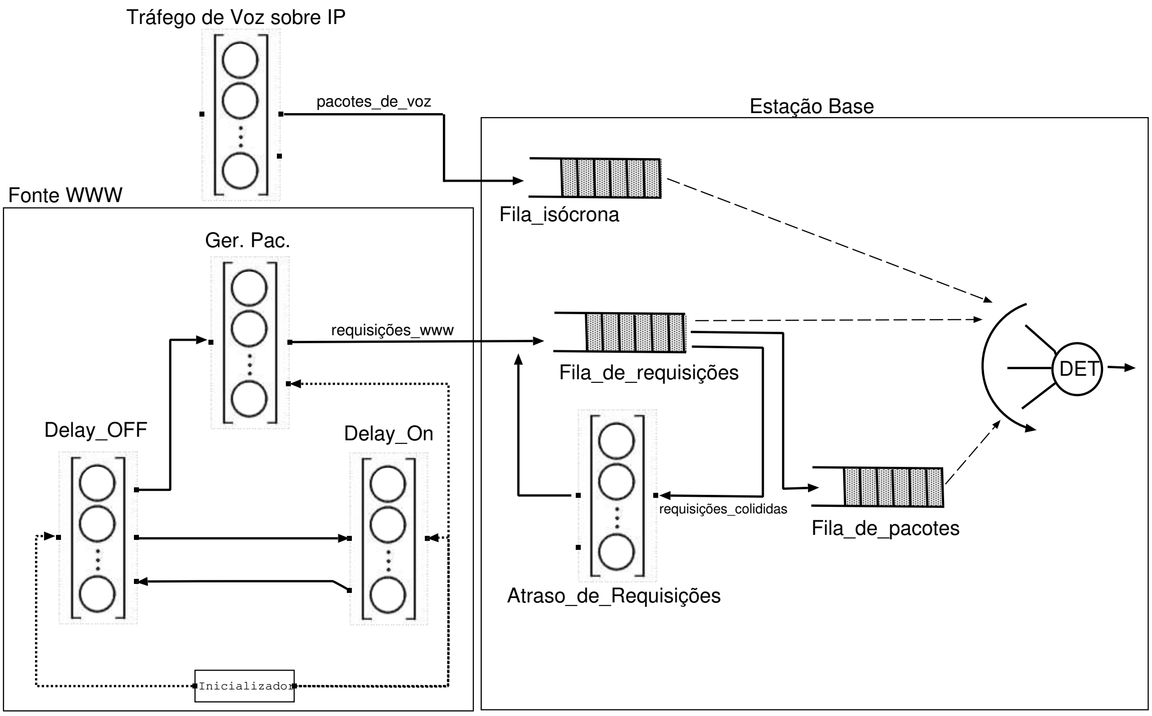


Figura 4.6: Visão geral do modelo de avaliação de desempenho do protocolo DOCSIS

Existe um tamanho máximo para um quadro, que consiste em um conjunto

Nome	Descrição	Tipo
Retrying_ Users	Número de pedidos aguardando nova tentativa	Constante
p	Prob. de enviar pedido no próximo disparo	Constante

Tabela 4.11: Parâmetros do objeto *Atraso_ de_ Requisições*

de CSs seguidos de DSs (1920 *bytes*). O tempo que cada quadro leva para ser processado e servido pelo canal depende da banda de subida. Quando se utiliza a banda de 5.12Mbps, este tempo é de 3 ms, e para a banda de 5.6Mbps este tempo é de 2.697ms para as relações de 5/23 e 15/21; e 2.743ms para a relação de 8/22. Sendo assim, se o número de reservas feitas for superior ao número máximo de *slots* de dados disponíveis no quadro, os pedidos só serão atendidos no próximo quadro de dados.

No objeto *Atraso_ de_ Requisições*, cada recebimento de mensagem resulta na clonagem do evento *Request_ Delay* e no incremento da variável de estado *Retrying_ Users*. Esse procedimento indica que mais um pedido colidiu e está efetuando uma espera aleatória para tentar novamente. Assim, ao disparo do evento *Atraso_ de_ Requisições*, com probabilidade p, uma mensagem é enviada ao objeto *Fila_ de_ Requisições*, e com probabilidade $1-p$, outro evento *Request_ Delay* é clonado e o pedido esperará mais um tempo aleatório para tentar novamente. Para que apenas uma tentativa, aproximadamente, seja feita a cada quadro, o evento *Request_ Delay* possui taxa $\lambda_t = (\frac{MAX\_DS}{LINK\_CAPACITY})^{-1}$, onde MAX_DS é o número de *slots* por quadro.

Resumidamente, cada pedido de reserva escolhe aleatoriamente um dentre os *slots* de disputa disponíveis. Em caso de colisão, as solicitações que colidiram são novamente submetidas depois de um atraso aleatório. Como, a princípio, deve ser pequena a probabilidade de uma requisição sofrer mais de duas colisões, o tempo médio de atraso é constante e não variável com o número de colisões conforme o protocolo. Essa suposição é razoável para um sistema bem dimensionado e pôde ser verificada durante as simulações. Os dois tipos de tráfegos são armazenados em filas separadas caso o servidor esteja ocupado quando um pacote já está sendo transmitido. A capacidade do *buffer* foi considerada, neste estudo, com o tamanho máximo de 8000 *slots* de dados, sendo este o valor máximo que a soma das duas filas pode assumir.

4.4 CDMA2000 1X

4.4.1 Introdução e Trabalhos Relacionados

O fator predominante no comportamento do sistema CDMA2000 1X é o escalonamento de canais. Por isso, o ponto chave na avaliação de desempenho deste mecanismo é a modelagem do processo dos pedidos de canais de dados feitos pelos usuários, bem como os temporizadores mantidos e eventos monitorados e as respostas enviadas pela estação base como resultado do algoritmo de controle de admissão.

Neste modelo, também é suposto que a população de usuários acessa a Internet predominantemente com aplicações *web* e voz sobre IP. O modelo é dividido em duas partes: o modelo do cliente, seja ele usuário *web* ou de voz; e o modelo da estação base, que essencialmente faz escalonamento de canais para toda a população de usuários, através de mensagens de controle, e mantém o estado global do sistema, com detalhes de todos os canais atribuídos.

Alguns trabalhos estudando o comportamento do CDMA2000 podem ser encontrados na literatura. Em [57] o principal objetivo foi o de investigar o desempenho e a compatibilidade do TCP/IP quando integrado a protocolos de redes sem fio sobre enlaces de rádio baseados no CDMA2000 utilizando um simulador proprietário. Foi observada uma diferença significante de desempenho quando se alterava a subdivisão do quadro da camada física em unidades lógicas de transmissão menores da camada MAC. Porém, como a avaliação apresentada se preocupou em mostrar diferenças de desempenho mediante modificações no protocolo e a única aplicação modelada foi a transferência de arquivos (FTP), o modelo apresentado não pode ser utilizado para dimensionamento de uma rede CDMA2000. Além disso, o simulador utilizado nos experimentos é proprietário.

Em [20] é feito um estudo sobre o desempenho do serviço de pacotes de dados do

CDMA2000 1X através de experimentos laboratoriais e de campo do sistema experimental, com ênfase em testar diferentes condições dos canais físicos. Contudo, as condições gerais consideradas foram muito específicas: as taxas dos canais utilizadas nos experimentos foram restritas a 153.6 e 38.4kbps, no canal suplementar direto, e 38.4kbps no canal suplementar reverso; o número de usuários considerados nos testes não passou de três; transferências de arquivos utilizando FTP constituem as únicas aplicações testadas e nenhum modelo de simulação foi proposto, já que foi utilizado um equipamento experimental.

No estudo feito em [58], é analisado o desempenho do serviço de pacotes de dados do CDMA2000 através de simulação, com o objetivo de avaliar o sistema sob diferentes situações dos canais de rádio. Os resultados são apresentados em termos de atraso de pacotes e vazão do canal. A simulação é feita utilizando-se como aplicação apenas um modelo simplificado dos usuários *web* com chegadas Poisson e distribuição exponencial dos tamanhos dos pacotes. A ferramenta de simulação não foi especificada.

Derryberry, Ma e Rong [59] analisaram o desempenho de transmissões conjuntas de tráfegos de voz (chamadas telefônicas convencionais) e de dados da evolução do CDMA2000 1X chamada CDMA2000 1XEV-DV, que busca disponibilizar maiores taxas de transmissão no canal direto. Para isso, utilizam a metodologia recomendada em [60] que indica simulação em nível de enlace para determinar a potência requerida para alcançar um determinado FER para um único enlace de rádio, e posteriormente os resultados destas simulações são aproveitados na simulação em nível de sistema para alcançar medidas de interesse como capacidade e a vazão dos canais, entre outros.

Huang, Yao, Bai e Wang [61] avaliaram medidas de interesse como a vazão, a probabilidade de perda de pacotes e a probabilidade de negação de pedidos de canais fundamentais utilizando um modelo de simulação, orientado a eventos, considerando um ambiente de simulação de 19 células. Sob este ambiente, o modelo é alimentado com fontes de tráfego de *web* e fontes de tráfego de *streaming* de vídeo. Para isso, consideraram um modelo de tráfego *web* com distribuição exponencial para: o tempo entre chegadas de pacotes de uma sessão, o tempo de leitura de cada página recebida, o número de pacotes por chamada e o tempo entre chegadas de pacotes. O tamanho dos pacotes é modelado utilizando a distribuição Pareto. Já os modelos de tráfego de vídeo e de voz são bastante simples para manter sob controle a complexidade das simulações. O tempo de acesso de páginas *web* não é analisado.

No trabalho de modelagem do CDMA2000 1X aqui desenvolvido, é considerado em conjunto os tráfegos *web* e de voz sobre IP tanto nos canais diretos quanto nos canais reversos. O modelo de simulação levou em conta características relevantes do padrão, como por exemplo, o escalonamento de canais e a sua política de acesso, para o desempenho do sistema como um todo, considerando os parâmetros e os procedimentos definidos em [60, 21].

Como resultado, tem-se um modelo detalhado, que pode ser utilizado para avaliação e dimensionamento de redes CDMA2000 1X de acesso à Internet, tendo como foco principal o número de usuários suportados. Como o controle de admissão foi implementado separadamente da modelagem do tráfego dos usuários, é possível adicionar ao sistema modelos de fontes de tráfego IP correspondentes a qualquer tipo de aplicação do usuário.

4.4.2 Descrição do Modelo

Os componentes do modelo são: o objeto usuário *web*, o objeto usuário de Voz sobre IP e o objeto Estação Base. Cada objeto usuário representa o comportamento de um usuário pertencente ao sistema executando sua respectiva aplicação. Para modelar o sistema com a presença de diversos usuários, replica-se o objeto usuário relativo ao tipo de tráfego desejado. Por exemplo, para simular o sistema utilizando uma carga de 4 usuários *web* e 2 usuários VOIP, será necessário criar 4 cópias do primeiro objeto e 2 cópias do segundo, como mostrado na Figura 4.7. Este processo é facilmente realizado através do TGIF.

Como pode ser visto na parte esquerda da Figura 4.8, o objeto do usuário abrange tanto o comportamento do tráfego do cliente quanto parte dos mecanismos da esta-

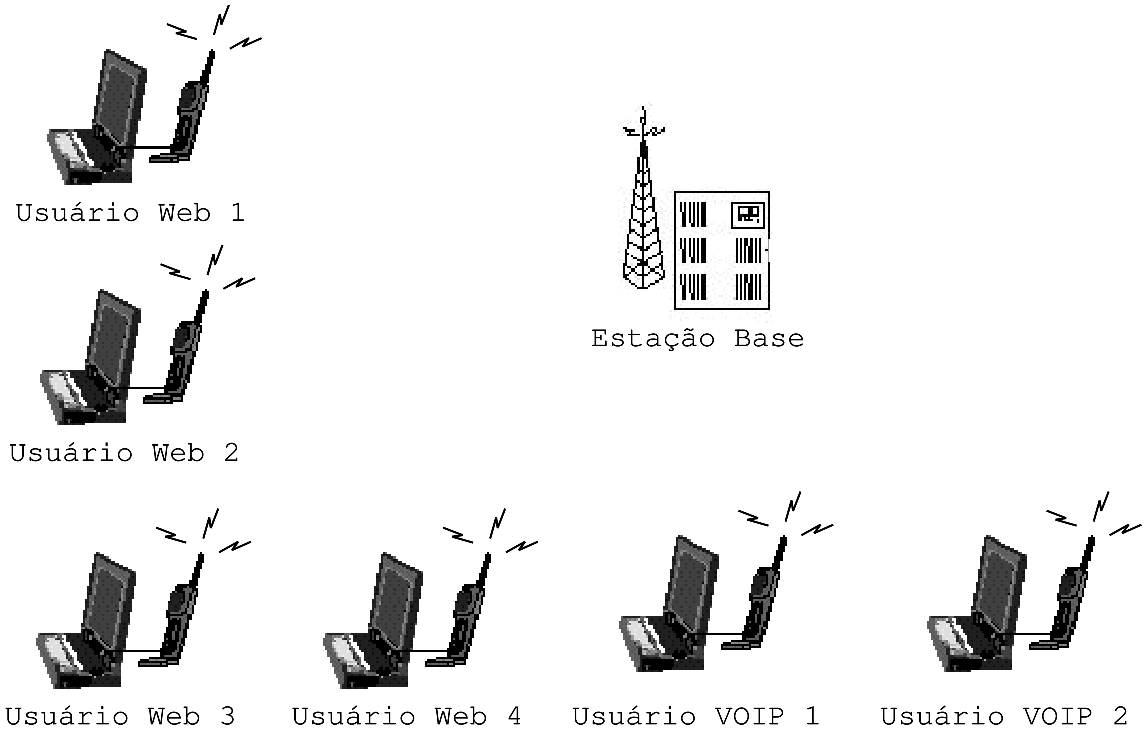


Figura 4.7: Modelo configurado para simular 4 usuários *web* e dois usuários VOIP

ção base relativas ao tráfego de dados. Esta abordagem foi escolhida para minimizar a complexidade da implementação do modelo proposto. Desta maneira, tanto o retorno das requisições *web* quanto do tráfego de voz sobre IP são implementados no próprio objeto do usuário, bem como os temporizadores individuais de estado da conexão da camada de enlace.

A decisão de utilizar essa metodologia tem como objetivo principal fazer com que o modelo de simulação seja escalável, ou seja, novos clientes possam ser adicionados ou retirados de acordo com a carga sob a qual se deseja realizar experimentos, sem que haja necessidade de alterar o código do objeto estação base.

Nesse sentido, cada cliente tem suas próprias filas relativas aos canais diretos e reversos e suas próprias variáveis de estado de controle de atribuição de canais. Se essas estruturas fossem implementadas no objeto estação base, haveria necessidade de criar uma estrutura para cada cliente neste objeto e à medida que o número de clientes fosse modificado, as variáveis também teriam que ser alteradas.

Sabendo que cada objeto usuário possui todo o código relativo ao tráfego gerado pelo mesmo, se novos objetos cliente forem adicionados ao sistema, automaticamente as estruturas necessárias também serão adicionadas.

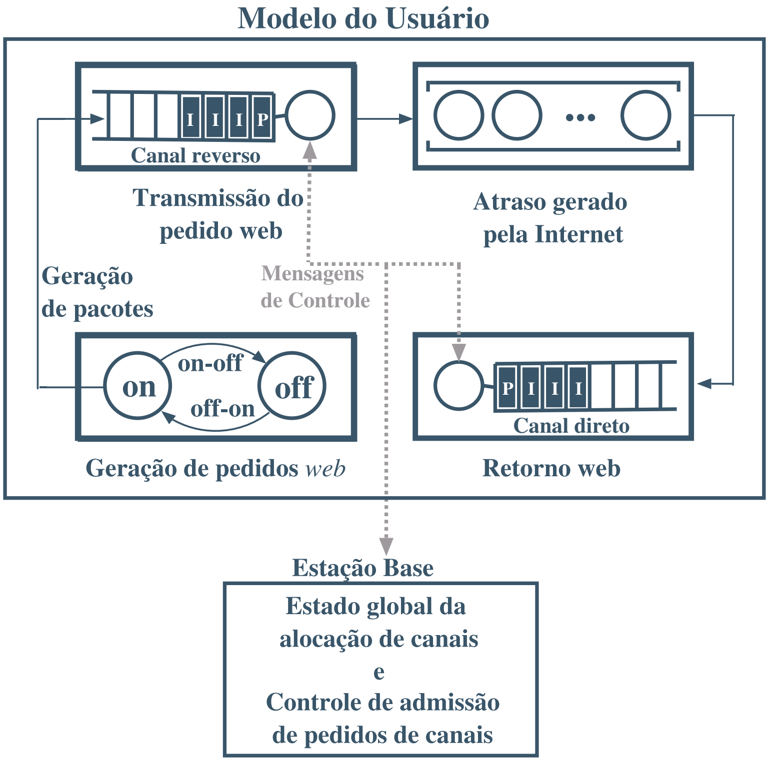


Figura 4.8: Visão geral do modelo de avaliação de desempenho do CDMA2000 1X - interação usuário *web*/Estação Base

O resultado final é semelhante à abordagem adotada pela programação orientada a objeto, onde novos objetos são instanciados de acordo com o andamento do processo. A diferença é que neste caso, o próprio usuário da ferramenta é quem adiciona clientes de acordo com o objetivo da simulação.

Além disso, podem ser realizadas várias otimizações a partir desta abordagem. Dois exemplos importantes são:

1. Otimização do número de mensagens enviadas pelos objetos. Suponha que a tráfego do cliente e da estação base fossem implementadas em dois objetos. Neste caso, toda vez que um pacote fosse transmitido pela rede, uma mensagem relativa ao pacote seria gerada. Na abordagem adotada neste modelo, a conseqüência imediata da transmissão de um pacote é apenas a alteração das variáveis de estado envolvidas, o que torna a simulação mais rápida devido à

Nome	Descrição
FUNDAMENTAL_CHANNEL_REQUEST	Requisição de canal fundamental
F_SUPLEMENTAL_CHANNEL_REQUEST	Requisição de canal suplementar direto
FUNDAMENTAL_CHANNEL_RELEASE	Liberação de canal fundamental
F_SUPLEMENTAL_CHANNEL_RELEASE	Liberação de canal suplementar direto
R_SUPLEMENTAL_CHANNEL_REQUEST	Requisição de canal suplementar reverso
R_SUPLEMENTAL_CHANNEL_RELEASE	Liberação de canal suplementar reverso
INACTIVE	Estado inativo
ACTIVE	Estado ativo
WAITING_CHANNEL_REPLY	Aguardando resposta

Tabela 4.12: Constantes de identificação de mensagens de controle e estados do objeto

supressão do envio de mensagens.

2. Otimização do número de eventos da simulação. Estando todo o tráfego relativo a um cliente implementado no mesmo objeto, há uma liberdade maior para otimizar o código com o objetivo de diminuir o número de eventos criados para o modelo, já que todas as variáveis de estado relativas ao tráfego deste cliente estão sob controle do mesmo objeto.

Tendo sido apresentadas as características gerais do modelo, foram criadas algumas constantes de identificação de mensagens e dos estados de cada objeto do sistema. Estas constantes visam facilitar o entendimento do modelo, através da disponibilização de um código mais legível, e são apresentadas na Tabela 4.12.

Os pedidos de canais são processados em tempo zero e, quando um canal é atribuído, o usuário recebe como resposta um número informando a taxa de transmissão do canal a ele alocado.

Objeto Usuário *Web*

Como pode ser observado na Figura 4.8, a fonte que representa o usuário, ao passar para o estado ON, gera pedidos. Se a fila de pedidos *web* não está vazia e o canal fundamental está inativo, é enviado à estação base um pedido deste canal. Caso o canal fundamental já esteja ativo e a fila de pedidos *web* estiver acima de um limitante (*QUEUE_ THRESHOLD*), que no modelo criado foi configurado para 1000 *bytes* (como considerado em [20]), então, é verificado se existe um canal suplementar ativo. Se não existir nenhum canal suplementar ativo, é enviado uma requisição deste último canal para a estação base.

A requisição de canal fundamental enviada pelo cliente à estação base pode não ser atendida. Neste caso, o evento chamado *Fundamental Channel Request Retransmission* trata da retransmissão desta requisição. Este evento só é habilitado quando o cliente já fez um ou mais pedidos de canais fundamentais, ou seja, quando o canal fundamental está no estado *WAITING_ CHANNEL_ REPLY*. Este evento determinístico é disparado de 2 em 2 segundos [21].

Outro evento, chamado *Reverse Suplemental Channel Request Retransmission*, evita que o cliente fique indefinidamente esperando por uma atribuição de canal suplementar. Enquanto o estado deste canal for igual a *WAITING_ CHANNEL_ REPLY* e a fila de pedidos *web* estiver acima do limitante (*QUEUE_ THRESHOLD*), o evento permanece habilitado. Este evento também é determinístico, sendo disparado de 2 em 2 segundos.

Quando um canal suplementar é atribuído ao usuário, o evento *Reverse Suplemental Channel Release* permanece habilitado enquanto o canal suplementar estiver ativo. Este evento visa garantir que o cliente só utilize o canal suplementar pelo tempo definido pelo protocolo. Assim, como um temporizador, este evento é determinístico e possui a seguinte taxa:

$$\lambda_{release}^{rev} = \frac{1}{t_{burst}}, \tag{4.1}$$

onde t_{burst} é o tempo máximo de utilização de canal suplementar e foi configurado para 5.12 segundos, de acordo com o estudo feito em [20].

Em [20], também é avaliado o impacto do *overhead* de cabeçalhos sobre a taxa efetiva de transmissão dos canais. É mostrado que aproximadamente 13% da taxa nominal é gasta na transmissão de cabeçalhos inseridos pelos protocolos do sistema (ex. RLP, PPP). Este *overhead* é considerado no modelo proposto, de maneira que as taxas de transmissão dos canais foram parametrizadas levando este detalhe em consideração.

O evento *Packet_ Transmission*, relativo à transmissão dos pedidos *web* gerados pelo cliente para a estação base, possui distribuição exponencial que simula a variação dos tamanhos dos pedidos, e possuem seguinte taxa:

$$\lambda_{tx} = \frac{\lambda_{fund} * \frac{Q_{rev}}{Q_{rev}+Q_{for}}, + \lambda_{rev}}{Size_{Req}} \tag{4.2}$$

onde

- λ_{fund} é a taxa de transmissão do canal fundamental;
- λ_{rev} é a taxa de transmissão do canal suplementar reverso;
- $Size\_Req$ é o tamanho médio dos pedidos *web*;
- Q_for é o tamanho da fila do canal direto;
- Q_rev é o tamanho da fila do canal reverso.

A condição para que este evento de transmissão ocorra é que a fila do canal reverso Q_{rev} seja maior que zero e que pelo menos o canal fundamental esteja ativo. Quando o evento de transmissão de pacotes é disparado, o tamanho da fila Q_{rev} é verificada. Caso a fila tenha se esvaziado, uma mensagem de liberação de canal suplementar é enviada à estação base e enquanto a fila permanecer com tamanho nulo o canal fundamental permanece no estado inativo.

Se o canal fundamental estiver inativo por um determinado período de tempo, um evento determinístico chamado *Fundamental_ Channel_ Release* que só é habilitado se as duas filas Q_{for} e Q_{rev} estiverem vazias e se o canal fundamental estiver ativo, dispara uma mensagem para a estação base reportando a liberação deste canal. Este evento na verdade funciona como o temporizador de inatividade do canal, e possui taxa $\lambda_f = \frac{1}{t_{in}}$, onde t_{in} representa o tempo configurado para o temporizador de inatividade do canal fundamental (*RLP_ INACTIVITY_ TIMER*)[21].

Ao efetuar a transmissão de um pacote, a variável de estado *Delaying_ Packets* é incrementada e o evento *Base Station Packet Delay* é clonado. A variável *Delaying Packets* representa o número de pacotes sendo transmitidos na Internet. O evento *Base Station Packet Delay* que representa o tempo desde o momento em que um pedido *web* é transmitido até o momento que uma resposta é recebida pela estação base, possui distribuição Gaussiana com parâmetros (*MEAN_ DELAY* e *DELAY_ VARIANCE*) que representam a média e a variância do atraso dos pedidos *web* na Internet [52], respectivamente.

Quando este evento é disparado, a resposta ao pacote já pode ser gerada. Assim, a fila Q_{for} é incrementada e a estação base passa a ter uma resposta ao pedido do cliente para transmitir pelo canal direto, como conseqüência, a fila *Delaying_ Packets* é decrementada.

As mesmas verificações de limitantes de fila que são feitas na modelagem do tráfego do cliente, também são utilizadas na modelagem do retorno de tráfego, que no sistema real é tarefa da estação base.

Assim, se a fila da estação base Q_{for} não estiver vazia e não houver canal fundamental atribuído ao respectivo cliente, então, uma mensagem contendo uma requisição de canal fundamental é enviada ao objeto Estação Base. Se o tamanho de fila estiver maior do que o limitante *QUEUE_ THRESHOLD*, é enviado um pedido de canal suplementar direto (*Forward Suplemental Channel*).

A estação base possui os mesmos temporizadores descritos na modelagem do tráfego do usuário com a diferença de que o foco, neste caso, é a liberação ou retransmissão de pedidos de canais diretos, e não reversos.

O evento responsável pelo serviço de pacotes na estação base (*Base Station Service*) decrementa a fila de pacotes Q_{for}. Caso a fila se esvazie quando um canal suplementar direto estiver alocado, é feita a liberação deste canal.

Este evento possui distribuição lognormal com os seguintes parâmetros:

$$E_{lognorm} = \frac{E[obj]}{(\frac{\lambda_{fund}*Q_{for}}{Q_{for}+Q_{rev}} + \lambda_{for})}, \tag{4.3}$$

$$\sigma_{lognorm} = \frac{\sigma_{obj}}{(\frac{\lambda_{fund}*Q_{for}}{Q_{for}+Q_{rev}} + \lambda_{for})^2}, \tag{4.4}$$

onde

- $E[obj]$ é a variável de estado que representa o tamanho médio do objeto *web* a ser transmitido;
- σ_{obj} é a variável de estado que representa variância do tamanho do objeto *web*;
- λ_{fund} é a taxa de transmissão do canal fundamental;
- λ_{for} é a taxa de transmissão do canal suplementar direto.
- Q_for é o tamanho da fila do canal direto.
- Q_rev é o tamanho da fila do canal reverso.

Esta abordagem é utilizada para modelar a distribuição do tamanho dos objetos *web* enviados como respostas aos pedidos do usuário. Por isso, a variáveis de estado relativas ao tamanho médio e variância do tamanho das respostas [62] (*Mean_ Packet_ Size* e *Packet_ Size_ Variance*), tem seu valor alterado dependendo do tipo de objeto, principal ou *in-line*, a ser transmitido ao usuário. Na Figura 4.8, estes dois tipos de objetos estão representados, nas filas dos canais direto e reverso, por "P" e "I".

Mudanças de estado podem ocorrer como conseqüência dos eventos gerados pelo próprio objeto usuário, ou quando alguma mensagem de controle vinda da estação base é recebida. Estas mensagens dizem respeito à comutação de canais de rádio freqüência ditada pelo controle de admissão, e podem ser mensagens de atribuição ou de remoção de canais.

Assim, é possível ilustrar o comportamento geral da camada de enlace do serviço de dados do CDMA2000 1X, quando um usuário *web* utiliza o sistema, através da

Figura 4.9. Neste exemplo, ao iniciar uma sessão de dados, o usuário tem direito a efetuar transmissões utilizando apenas o canal fundamental. Caso a taxa de transmissão oferecida por este canal de comunicação não seja suficiente para manter a fila abaixo do limitante definido, um canal suplementar é ativado. Como o canal suplementar só pode ser utilizado por um curto período de tempo, o usuário logo perde o direito de utilizá-lo e, caso a fila ainda esteja acima do limitante definido, é feito um novo pedido de canal suplementar. Este processo se repete até que o usuário não tenha mais dados para transmitir.

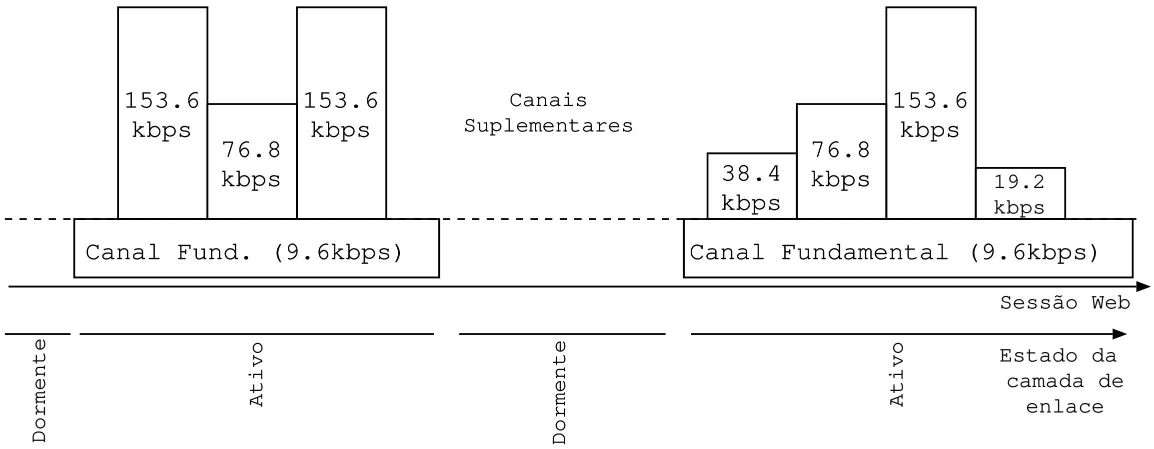


Figura 4.9: Visão geral do comportamento da camada de enlace do serviço de dados do CDMA2000 1X sob uma sessão *web*

Caso o canal fundamental fique ocioso por um tempo definido, o usuário perderá o direito de transmitir neste canal, embora a conexão da camada de enlace não seja finalizada. Se houver nova necessidade de transmissão, um pedido de reativação de canal fundamental terá que ser enviado, já que, no modelo, não é armazenado o estado da conexão, como acontece no protocolo, quando o usuário está no estado dormente.

Objeto Usuário de Voz sobre IP

Os usuários de tráfego de voz sobre IP têm uma implementação similar à apresentada no item anterior, com exceção dos eventos relativos ao tráfego gerado por este usuário, que são implementados de acordo com um processo de nascimento e morte, como apresentado na seção 4.2.

Como informado naquela seção, a abordagem de modelagem escolhida para os modelos de usuários de voz na descrição do ambiente do CDMA2000 1X foi o paradigma de fluido, a Figura 4.10 ilustra o modelo criado utilizando a abordagem em questão.

Duas recompensas de taxa foram criadas para representar as filas dos canais direto e reverso, de cada usuário: *Forward_ Queue_ RW* e *Reverse_ Queue_ RW*. Estas recompensas representam recipientes que acumulam ou escoam fluido de acordo com as taxas de geração e serviço de dados. A cada evento que modifica o estado do sistema quanto às taxas, o IR (*Instantaneous Reward*) destas recompensas são configurados com o valor da diferença entre a taxa de geração de dados e a taxa de serviço. Assim, a cada intervalo de tempo, os recipientes estarão acumulando, se mantendo estáveis, ou escoando dados representados pelo fluido.

Sejam as seguintes recompensas e variáveis de estado:

- $IR_{Q_{for}}$ o valor instantâneo da recompensa *Forward_ Queue_ RW*;
- $IR_{Q_{rev}}$ o valor instantâneo da recompensa *Reverse_ Queue_ RW*;
- S_s é o estado da fonte On-Off (0 inativa e 1 ativa);
- λ_s a taxa de geração de dados do usuário de voz (64kbps);
- S_{fch} o estado do canal fundamental (0 inativo e 1 ativo);
- μ_{fch} taxa de serviço do canal fundamental;
- Q_{for} o valor acumulado da recompensa *Forward_ Queue_ RW*;
- Q_{rev} o valor acumulado da recompensa *Reverse_ Queue_ RW*;
- μ_{for} taxa de serviço do canal suplementar direto;
- μ_{rev} taxa de serviço do canal suplementar reverso.

A atualização da taxa de crescimento/decrescimento das filas é realizada baseada nas equações descritas abaixo. Sempre que ocorre um evento no sistema que reflete em mudança de taxas dos canais fundamental ou suplementar direto, a recompensa *Forward_ Queue_ RW* tem seu IR configurado para

$$IR_{Q_{for}} = (S_s\lambda_s) - (S_{fch}\mu_{fch})(\frac{Q_{for}}{Q_{for} + Q_{rev}}) - \mu_{for} \tag{4.5}$$

e sempre que ocorre mudança nas taxas dos canais fundamental ou suplementar reverso, a recompensa *Reverse_ Queue_ RW* tem seu IR configurado para

$$IR_{Q_{rev}} = (S_s\lambda_s) - (S_{fch}\mu_{fch})(\frac{Q_{rev}}{Q_{for} + Q_{rev}}) - \mu_{rev}. \tag{4.6}$$

Essas equações permitem que o crescimento/decrescimento das recompensas relativas à fila de um usuário, seja semelhante ao comportamento real do canal de acesso, onde o canal fundamental é disputado pelas filas direta e reversa, e os canais suplementares são dedicados à fila de cada sentido.

Mensagens de controle são trocadas com a estação base quando o recipiente de fluído utilizado para representar o *buffer* dos canais de transmissão ultrapassam os mesmos limitantes descritos na seção anterior, indicando a solicitação de canal fundamental ou suplementar, ou quando alguma mensagem de controle vinda da estação base é recebida, indicando a remoção de canais.

Objeto Estação Base

O objeto Estação Base é responsável pelo controle de admissão. Como apresentado nas seções anteriores, por motivos de simplificação, os objetos relativos aos clientes implementam a geração de tráfego dos canais diretos e dos canais reversos. Duas constantes foram criadas para facilitar a manipulação de mensagens recebidas, como mostrado na Tabela 4.13.

Quando uma mensagem é recebida pela estação base, ela é armazenada em um vetor de duas posições chamado *msg_ vec*. As constantes *ID* e *MSG_ DATA* são utilizadas para identificar as posições de *msg_ vec*. Assim, *msg_ vec[ID]* indica o número identificador do objeto usuário remetente da mensagem e *msg_ vec[MSG_ DATA]*

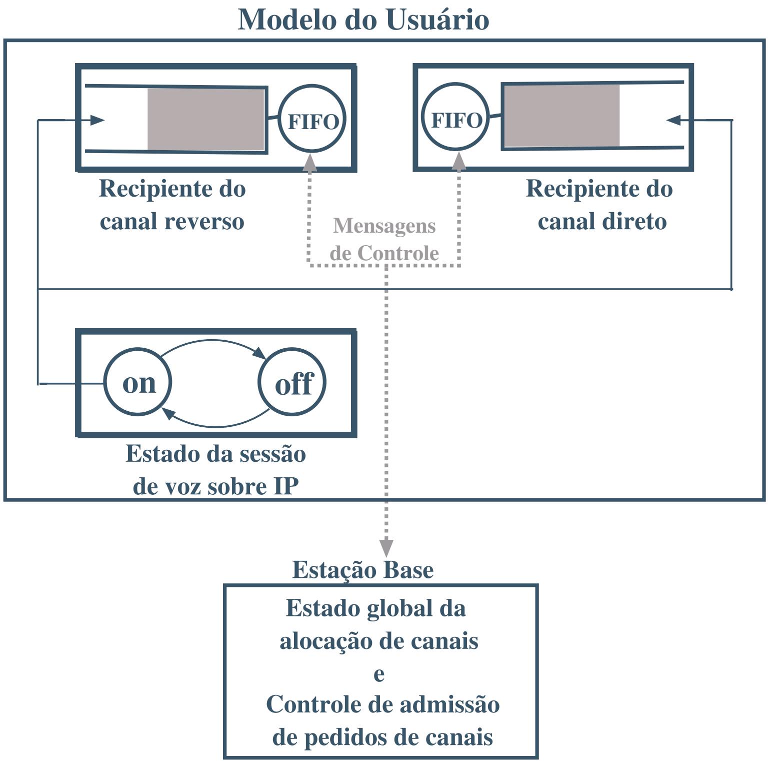


Figura 4.10: Visão geral do modelo de avaliação de desempenho do CDMA2000 - interação usuário de voz/Estação Base

Nome	Valor
ID	0
MSG_DATA	1

Tabela 4.13: Constantes utilizadas para manipulação de mensagens de controle

indica o campo de dados da mensagem.

A primeira verificação feita, após o recebimento de uma mensagem, é se o remetente da mensagem já possui um identificador. Caso negativo, a estação base busca um endereço disponível em um vetor de identificadores local. Estes identificadores são utilizados para que seja possível manter, na estação base, a informação de que tipos de canais de dados estão alocados a cada cliente CDMA2000 1X. Posteriormente, essas informações serão necessárias caso a estação base necessite cancelar a alocação de canais a um determinado cliente.

Após a verificação do identificador do objeto remetente, a estação base verifica o tipo de mensagem (msg_vec[MSG_DATA]). Para identificação destes dados são utilizadas as mesmas constantes de valores exclusivos mostrados na Tabela 4.12.

O vetor *Client_Status[MAX_CLIENTS]* informa que canais estão alocados a cada usuário. Assim, o valor Client_Status[i], onde i é o número de identificação do cliente desejado, informa quais canais estão alocados ao usuário i. A constante MAX_CLIENTS define a população máxima suportada pelo sistema.

As últimas duas variáveis de estado indicam a carga do sistema nos canais direto (*Forward_Load*) e reverso (*Reverse_Load*). Estes valores são utilizados na Equação 2.3, utilizada para o cálculo do controle de admissão cada vez que chega um pedido de um canal. A soma das duas variáveis representam a carga u e são armazenadas na variável de estado *Load*.

Como introduzido no capítulo 2, a carga u é definida como $\frac{N}{N_{max}}$ onde N é o número de canais ativos e N_{max} é o *pole capacity* [37], ou seja, o número máximo de canais simultâneos teoricamente suportados caso os usuários possuíssem potência ilimitada para suas transmissões. O apêndice A indica os valores [21] utilizados para N_{max} no modelo proposto.

A seguir, será descrito o comportamento do objeto caso seja solicitado um canal fundamental ou canal suplementar.

Pedido de um Canal Fundamental

Quando chega um pedido de canal fundamental, que é *half-duplex*, o teste para aceitação deve ser feito duas vezes para satisfazer a sensibilidade do canal direto e do canal reverso. Se algum dos dois testes falharem, o canal não é atribuído.

Como os canais fundamentais tem prioridade sobre canais suplementares, a estação base deve verificar por que razão o controle de admissão rejeitou o pedido de canal fundamental. Se a rejeição foi causada pela falta de recursos de canais diretos, a estação base procura, no vetor *Client_Status*, por canais suplementares diretos já atribuídos. Será selecionado o canal suplementar de maior taxa que está atualmente alocado e uma mensagem será enviada ao respectivo objeto cliente para informar o cancelamento deste canal suplementar. Os recursos liberados por este canal serão, então, utilizados para tentar satisfazer o pedido em questão. O cliente que perdeu o canal suplementar direto fará um novo pedido de canal suplementar assim que o próximo evento de serviço for disparado. Como um canal fundamental já foi atribuído utilizando os recursos recém liberados, o cliente provavelmente vai receber um canal suplementar direto de taxa inferior à taxa anterior, ou o pedido será negado até a liberação de novos recursos.

Caso a falta de recursos para canais reversos seja a razão da rejeição do pedido de canal fundamental, o mesmo processo acima descrito será repetido, mas com foco em liberação de recursos de canais suplementares reversos.

Pedido de Canais Suplementares

Quando o objeto estação base recebe o pedido de um canal suplementar direto, é verificada a possibilidade de alocação de um canal com a maior taxa de transmissão possível. Para isso, o controle de admissão é feito primeiramente para a atribuição de um canal de 153.6kbps. Caso a potência não seja suficiente para um canal deste tipo, o controle de admissão é novamente feito para tentar atribuir um canal de 76.8kbps. Se também não houver recursos para este canal, o teste é novamente feito para um canal de 38.4kbps. Finalmente, uma última tentativa de atribuição é feita para um canal de 19.2kbps. Se este último teste falhar, o pedido de canal suplementar direto não é atendido.

Como os canais suplementares são *simplex*, os testes neste caso são feitos apenas para os canais diretos. No caso do pedido de canais reversos, o teste é feito apenas para os canais reversos, dado que só o cliente transmite através deste. Os parâmetros utilizados no controle de admissão são apresentados no apêndice A, de acordo com exemplos e análises feitas em [20, 21]

Liberação de Canais

Finalmente, a liberação de canais pode acontecer em duas situações: (1) um cliente pode decidir que não precisa mais utilizar um canal e, então, envia uma mensagem à estação base reportando a liberação de um canal (fila = 0); ou (2) como conseqüência do controle de admissão, onde a estação base pode revogar a atribuição de um canal suplementar anteriormente dada a um determinado cliente, como previamente explicado neste capítulo.

Capítulo 5

Experimentos Realizados e Resultados Obtidos

NESTE capítulo serão apresentados os experimentos realizados com os modelos criados para os padrões DOCSIS e CDMA2000 1X, juntamente com os resultados obtidos. Também serão descritos os testes realizados para validar as novas funcionalidades implementadas na ferramenta Tangram-II.

5.1 Validação das Novas Funcionalidades

Alguns experimentos foram realizados com o objetivo de mostrar a corretude de algumas das funcionalidades implementadas. Especificamente, as distribuições de probabilidade e a geração de variável aleatória são validadas nesta seção.

Para verificação da implementação da geração de variáveis aleatórias com distribuição Weibull, foi criado um modelo simples que possui apenas um objeto chamado *Teste_ Weibull*. Este objeto contém um evento que fica indefinidamente habilitado e utiliza a distribuição Weibull com parâmetros $\theta = 90.01$ e $k = 0.88$. Uma simulação foi executada de modo a obter um *trace* com 10^6 amostras do tempo entre eventos. A partir deste *trace*, foi calculada a função de distribuição cumulativa (CDF) que foi comparada à analítica, dada por $F(x) = (1 - e^{-\frac{1}{\theta^k}x^k})$. Através da comparação entre as duas curvas mostradas na Figura 5.1, o comportamento da distribuição implementada pôde ser validado.

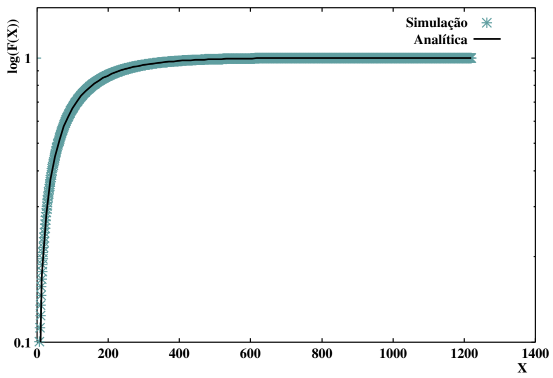


Figura 5.1: CDF simulada/analítica da distribuição Weibull

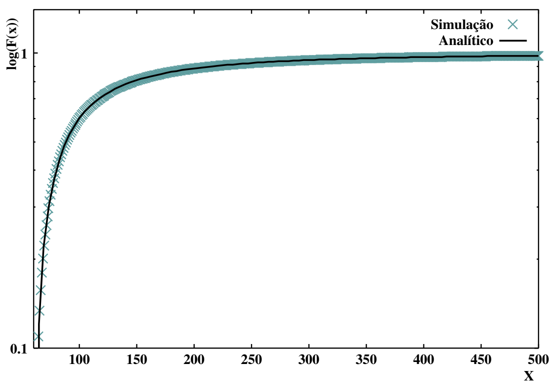


Figura 5.2: CDF simulada/analítica da distribuição Pareto

O mesmo processo foi seguido para verificação da geração de variáveis aleatórias com distribuição Pareto. A CDF desta distribuição é dada por $F(x) = 1 - (\frac{k}{x})^{\alpha}$, onde considerou-se os parâmetros $k = 60$ e $\alpha = 1.8$. A comparação entre as curvas obtidas pode ser visualizada na Figura 5.2 que evidencia a eficiência da simulação.

Para validar o novo método de escolha de sementes para a geração de variáveis aleatórias implementada na ferramenta, foi criado um modelo com um objeto e diferentes eventos com distribuição exponencial. Após executar um experimento de simulação por um tempo longo, foi calculada a correlação entre as amostras dos diferentes eventos. Como resultado, obteve-se valores nulos das correlações, para todos os intervalos testados.

5.2 DOCSIS

Os experimentos realizados tiveram como objetivo avaliar o desempenho do protocolo quando submetido ao tráfego gerado por usuários *web* e de voz sobre IP. Três cenários foram considerados nos experimentos, de acordo com o projeto de cooperação no qual este trabalho esteve envolvido. Estes cenários serão apresentados na seção 5.2.1.

Como o modelo é complexo e envolve eventos com taxas muito distintas, cuidados foram tomados, em sua construção, de forma a aumentar a eficiência da simulação. Por exemplo, o tempo de serviço de um *slot* de dados é de 125 μseg. Por outro lado, para a grande maioria dos usuários *web*, a duração do tempo em OFF é de aproximadamente 5 minutos. Uma maneira de minimizar o problema da diferença entre a taxa desses dois eventos é aumentar a duração de um deles. Consideramos no modelo o serviço de um conjunto de *slots*, ao invés do serviço de um único *slot*, aumentando desta forma, o tempo deste evento. Este conjunto de *slots*, que é servido, é igual ao número de *slots* prontos para serem servidos em um quadro, ou seja, o número de *slots* de dados previamente reservados pelos usuários através do período de disputa do quadro anterior.

Além disso, a simulação de distribuições com cauda longa também pode trazer efeitos adversos, como observado em [53] e comentado no capítulo 4. Em uma simulação contendo tais distribuições pode ser praticamente impossível atingir convergência para o estado estacionário. Este problema pode ser observado, por exemplo, em simulações contendo a distribuição Pareto com $\alpha < 1.7$, como o modelo estudado neste trabalho. Nas simulações realizadas, observou-se este problema, principalmente com relação ao tempo em OFF. Os procedimentos e considerações adotados, neste caso, foram apresentados no capítulo 4.

5.2.1 Cenários de Experimentos e Resultados

O objetivo dos experimentos foi o de investigar a qualidade dos serviços oferecidos pelo DOCSIS, com a variação do número de clientes. Para isso, foram considerados três cenários. Cada ponto mostrado nos gráficos de resultados demoraram, em média, entre 10 e 25 horas.

O objetivo do primeiro cenário foi o de avaliar o comportamento do sistema sob diferentes configurações CS/DS. Para isso, configurou-se a população *web* para 2000 e 3000 usuários, utilizando as relações CS/DS de 5/23 e 15/21, sendo que neste ambiente não foram feitas limitações à capacidade da fila e o canal foi configurado para 5.12Mpbs. Sob as duas populações de usuários *web* foram feitas variações no número de usuários usando o serviço de voz sobre IP (tráfego isócrono). A medida de desempenho calculada é o tempo médio de acesso dos usuários *web*, ou seja, o tempo desde que um usuário *web* gera o seu pedido até que ele seja servido com sucesso pelo canal de comunicação. Através da obtenção do tamanho médio da fila N de um sistema e da sua vazão média μ, é possível calcular facilmente o tempo médio de acesso T, através da relação $T = \frac{N}{\mu}$. O intervalo de confiança (95%) dos resultados, que serão apresentados, permaneceu abaixo de 2% do valor das medidas obtidas, com exceção do intervalo de confiança (95%) do tamanho médio da fila dos usuários, que chegou a 12% do valor das medidas obtidas, quando o sistema foi submetido a altas cargas.

O gráfico da Figura 5.3 mostra o tempo médio de acesso para uma população de 2000 usuários *web*. Observa-se que, para 15 *slots* de disputa, o tempo de acesso é da ordem de poucos milisegundos até uma população de 1000 usuários de voz. A partir de 1100 e até 1200 usuários, o tempo de acesso passa a ser da ordem de poucos segundos. A partir de 1200 usuários, o tempo cresce tendendo a infinito. Quando foi considerado 5 *slots* de disputa, observou-se que o tempo de acesso diminui bastante. Comparando o tempo de acesso para 1200 usuários, observa-se que há uma redução de 6 minutos para 486 ms.

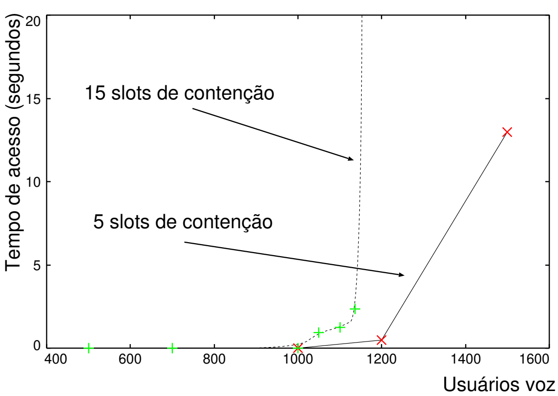


Figura 5.3: Tempo de acesso de usuários *web*. População *web* = 2000

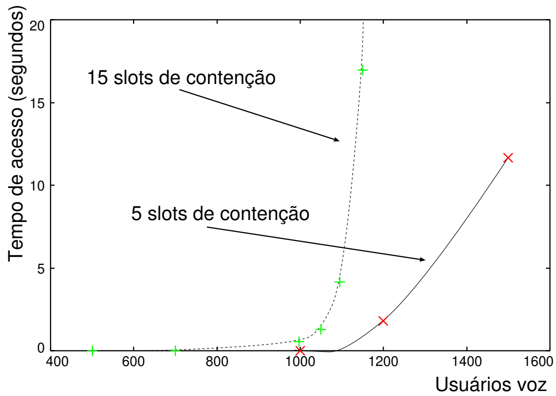


Figura 5.4: Tempo de usuários *web*. População *web* = 3000

O gráfico da Figura 5.4 mostra o tempo médio de acesso para uma população de 3000 usuários *web*. O tempo de acesso para 15 *slots* de disputa é da ordem de poucos milisegundos até uma população de 700 usuários de voz. A partir de 1000 e até 1150 usuários o tempo de acesso passa a ser da ordem de poucos segundos. A partir de 1200 usuários o tempo cresce tendendo a infinito. Já para o caso de 5 *slots* de disputa, pode-se observar uma redução significativa no tempo de acesso. Para uma população de 1200 usuários o tempo de acesso diminui de duas ordens de grandeza.

Como conclusão deste primeiro cenário, pode-se observar o aumento da população de usuários suportados quando é considerado um número menor de *slots* de disputa por quadro.

Com o objetivo de realizar experimentos sob um ambiente mais próximo ao encontrado no equipamento disponível para testes, nos laboratórios das instituições envolvidas no projeto, foi considerado um segundo cenário onde a relação CS/DS foi modificada para 8/22 e a capacidade do canal para 5.6Mbps. Neste cenário, o tamanho do *buffer* isócrono+*web* foi limitado para 8000 *slots* de dados e foram avaliados: (1) o tamanho médio da fila, (2) a vazão média do canal em *slots* de dados por segundo (ds/s), (3) o atraso médio de acesso, (4) a perda de pacotes do tráfego isócrono e do tráfego *web* e (5) a fração de colisões do período de disputa, ou seja, o número de colisões dividido pelo total de tentativas.

É interessante observar que, para a situação acima descrita, o tempo de acesso nunca será maior que 1 segundo, já que a capacidade de serviço do canal passou a ser de 8156 *slots* de dados por segundo. Então, como mostrado na Figura 5.5, sendo a capacidade da fila de 8000 *slots* de dados, mesmo que a fila fique cheia na maioria do tempo, o pacote não esperará mais do que 1 segundo para ser servido. A Figura 5.6 mostra que, aumentando-se bastante o número de usuários de voz, a média da fila aproxima-se de seu valor máximo.

A Figura 5.7 mostra que o canal se torna saturado à medida que o número de usuários de voz ultrapassa 3000 e chega a 4000. Ao se aproximar de 8000 *slots* de dados por segundo, a vazão do canal, neste ambiente, é equivalente a aproximadamente 91,43% da taxa nominal de 5.6Mbps.

A Figura 5.8 aponta que, mesmo aumentando apenas o número de usuários de

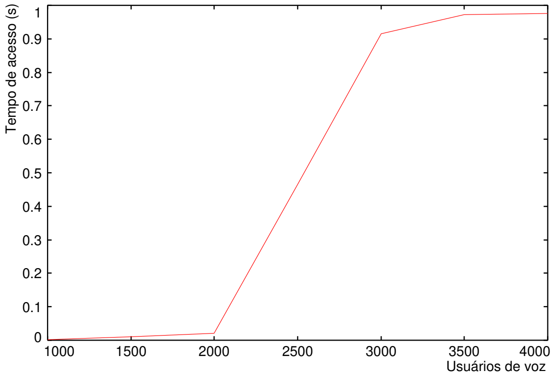


Figura 5.5: Tempo de acesso. População *web* = 2000; cs/ds = 8/22

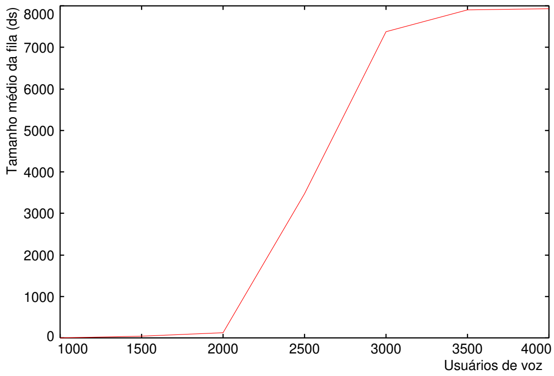


Figura 5.6: Tamanho da fila. População *web* = 2000; cs/ds = 8/22

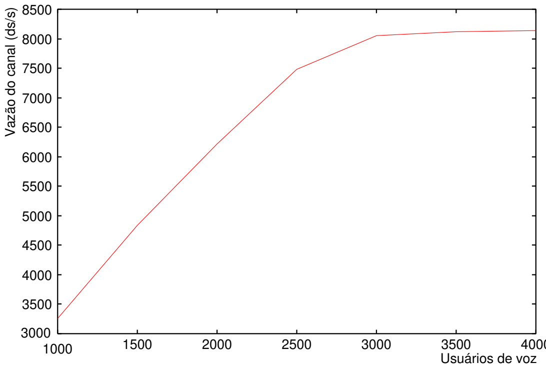


Figura 5.7: Vazão do canal: população *web* = 2000; cs/ds = 8/22

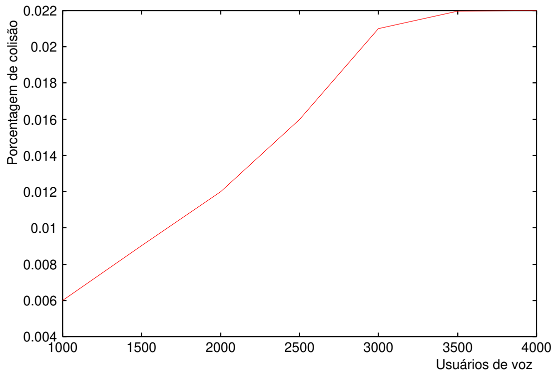


Figura 5.8: Fração de colisões: população *web* = 2000; cs/ds = 8/22

voz, a fração de colisão do tráfego *web* também aumenta. Isso acontece porque parte do quadro destinada à transmissão de dados cresce à medida que mais tráfego de voz é injetado na rede, ou seja, o número de *slots* de dados em um determinado quadro depende do número de pacotes estão na fila. Assim, se não há pacotes para servir, o quadro terá o tamanho mínimo que é justamente o número de *slots* de disputa. Em contrapartida, o quadro pode crescer até um valor máximo (que neste caso é de 22 *slots* de dados) caso a fila aumente. Com o aumento do quadro, o tempo que um usuário *web* deve esperar para fazer um pedido também irá aumentar. Logo, ao final da transmissão do quadro existirão, em média, mais usuários *web* esperando para fazer pedidos e conseqüentemente mais pedidos serão gerados, aumentando o número de colisões.

A fração de perda é mostrada na Figura 5.9. Como o protocolo dá prioridade ao tráfego isócrono, mesmo estando fixo o número de usuários *web*, a perda destes

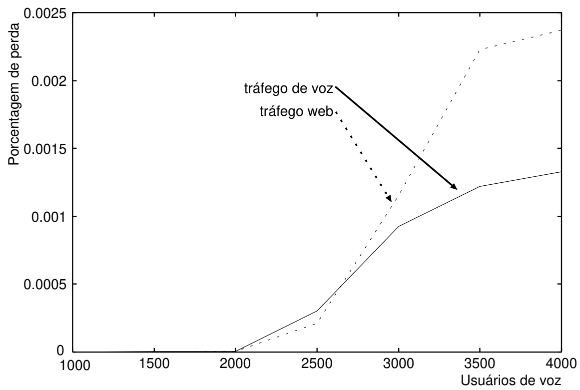


Figura 5.9: Fração de perda: população web = 2000; cs/ds = 8/22

pedidos cresce mais rapidamente que a dos usuários de voz. A perda em ambos os casos, torna-se maior justamente nos pontos em que o tamanho médio da fila aproxima-se da sua capacidade máxima.

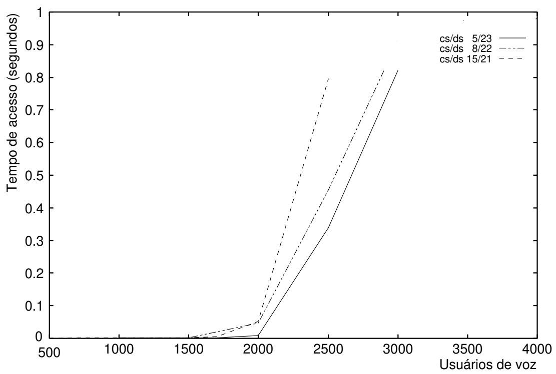


Figura 5.10: Tempo de acesso - pop. *web* = 2000; cs/ds=5/23, 8/22 e 15/21

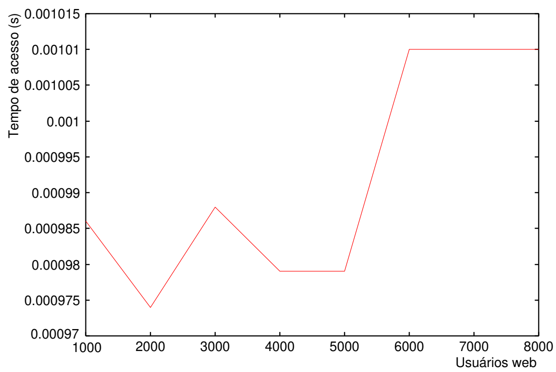


Figura 5.11: Tempo de acesso - população voz = 1000; cs/ds = 8/22

Na Figura 5.10 apresenta-se o comportamento do tempo de acesso, variando a população de usuários de voz, incluindo agora a relação cs/ds de 8/22 e comparando com as diferentes relações avaliadas anteriormente. As relações cs/ds são escolhidas de modo a que os quadros (*slots* de disputa + *slots* de dados) tenham o tamanho definido pelo padrão DOCSIS. Esta figura mostra que a diminuição do número de *slots* de disputa, que tem como conseqüência o aumento do número de *slots* de dados em cada *frame*, diminuiu sensivelmente o tempo de acesso sob um mesmo ambiente. Através deste comportamento pode-se concluir que o gargalo no sistema não é o período de disputa, o que também é observado nas Figuras 5.4 e 5.3.

No terceiro cenário foi considerada uma população de voz constante e igual a

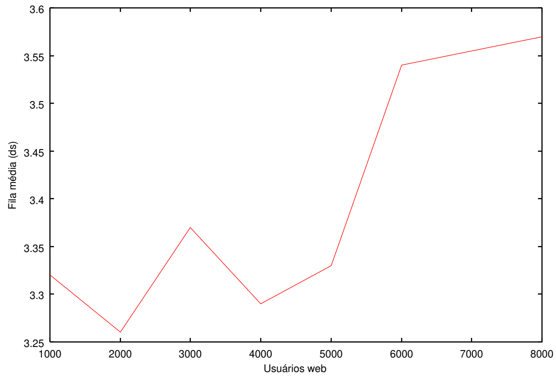


Figura 5.12: Tamanho da fila. População voz = 1000; cs/ds = 8/22

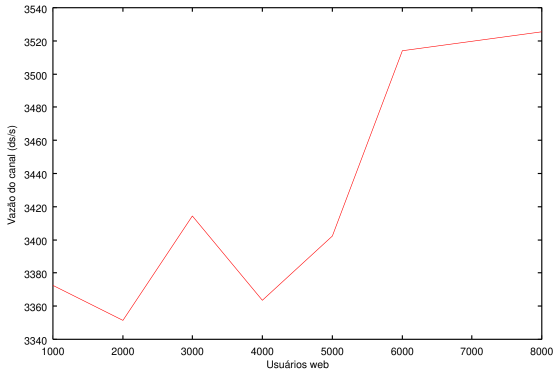


Figura 5.13: Vazão do canal: população voz = 1000; cs/ds = 8/22

1000 usuários. Neste ambiente, variou-se o número de usuários *web* entre 1000 e 8000. Na Figura 5.11, observa-se que, mesmo quando o número de usuários *web* chega a 8000 o tempo de acesso, embora tenha aumentado, não se aproximou de 1 segundo, o que ocorreu para o cenário 2, onde a população de voz variou até 4000.

Nas figuras 5.11, 5.12 e 5.13, o intervalo de confiança (95%) foi de aproximadamente 12% dos valores das medidas obtidas. Por isso, os valores obtidos quando a população *web* foi configurada para 3000, estão dentro do intervalo de confiança, mesmo estando ligeiramente acima dos valores obtidos para população de 4000.

O tamanho médio da fila, visto na figura 5.12, não ultrapassa 3.6 *slots* de dados mesmo quando o número de usuários *web* chega a 8000, enquanto a vazão do canal, mostrada na figura 5.13, também tende a aumentar com o aumento da população *web*, mas não chega a metade da capacidade do canal.

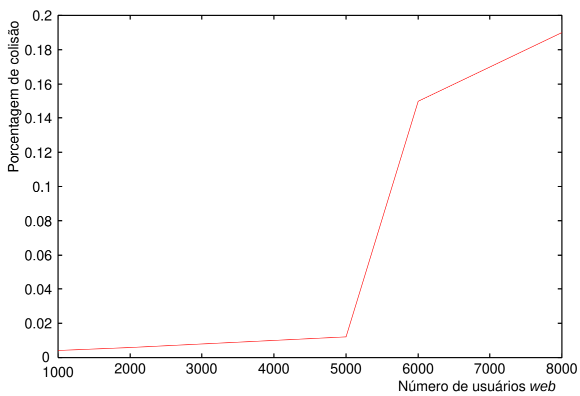


Figura 5.14: Fração de colisões: população voz = 1000; cs/ds = 8/22

Já na figura 5.14, onde é indicada a fração de colisões, pode-se ver facilmente o ponto de ruptura da capacidade de atender pedidos no período de disputa. Enquanto a população *web* não ultrapassa 5000, o número de colisões mantém-se em aproximadamente 2%. Mas, a partir desse momento, esta fração cresce até 19%, à medida que a população *web* cresce para 8000.

Através dos experimentos realizados, pode-se observar que o mecanismo DOCSIS é capaz de atender uma população de até 1000 usuários de voz e 8000 usuários *web*, fornecendo um tempo de acesso, para estes usuários, da ordem de 0.98ms. Percebe-se, também, que o crescimento dos usuários de voz tem um impacto maior no tempo de acesso que o crescimento dos usuários *web*. Este fato deve-se às características do tráfego gerado pelos usuários *web*: mensagens de tamanho pequeno (solicitações de páginas) e intervalo entre mensagens da ordem de segundos.

5.3 Experimentos e Resultados das Simulações do CDMA2000 1X

5.3.1 Definição do Raio da Célula

O primeiro passo para a avaliação de desempenho utilizando o modelo implementado foi a definição do raio da célula à qual os usuários estarão distribuídos. Para isso foi definido um cenário onde se fixava uma distância de $d$km e, neste ambiente, o sistema era testado sob diferentes configurações de carga para verificar se um pedido de FCH ou de SCH seria ou não atendido.

Este ambiente permitiu observar que a potência requerida pelo receptor, para que uma chamada seja aceita, tende a aumentar à medida que a carga do sistema aumenta, e tende a infinito quando a carga se aproxima da capacidade de pólo. Na prática, se a carga do sistema aumenta demasiadamente, os usuários na borda da célula não conseguem fornecer a potência necessária para abrir um novo canal de comunicação, e o sistema começa a ficar saturado. Uma forma de um transmissor

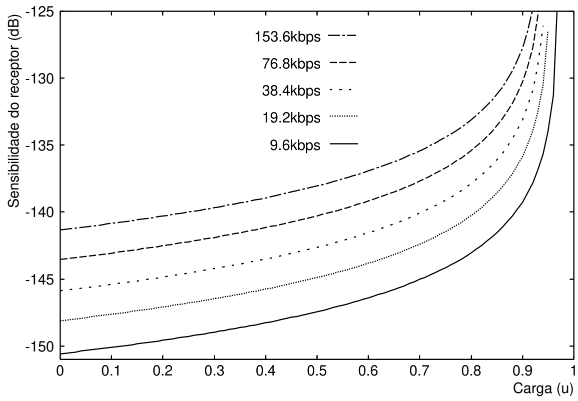


Figura 5.15: Relação entre a carga e a sensibilidade do receptor

aumentar a potência recebida pelo receptor é se aproximar do mesmo, de forma a diminuir a atenuação de sinal causada pelo *path loss*. Em outras palavras, quanto maior for a carga do sistema, menor será sua área de cobertura. Este comportamento pode ser observado na Figura 5.15, onde o eixo das abscissas indica a carga do sistema, em relação à carga pólo, e o das ordenadas indica sensibilidade do receptor, definida na equação 2.3.

Outra característica importante observada foi que, sob uma mesma carga, quanto maior for a taxa de serviço do canal, maior será a potência requerida pelo receptor para que a qualidade mínima estipulada para o canal seja garantida, ou seja, a área de cobertura depende do tipo de canal em questão, o que resulta em diferentes raios de alcance, em uma mesma célula, para os diferentes canais de comunicação, como mostrado na Figura 5.16.

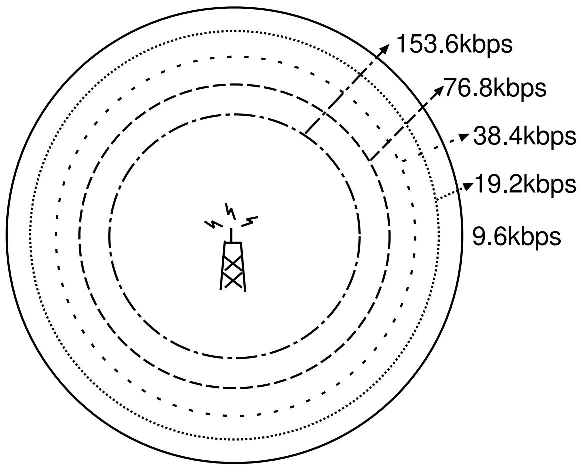


Figura 5.16: Relação entre tipos de canais e área de cobertura

Assim, embora valores típicos [61] para o tamanho de célula encontrados na literatura sejam de 3.4km, no cenário de simulações do modelo proposto, considera-se uma célula de 1.8km, onde os usuários estão distribuídos na borda da célula. Este ambiente foi selecionado no intuito de que houvesse recursos suficientes para alocação de canais de 153kbps, pelo menos quando o sistema estivesse sob cargas baixas.

5.3.2 Resultados dos Experimentos

Nos experimentos executados, o número de usuários de voz sobre IP é constante e a população de usuários *web* é variável. Especificamente, a população de usuários VOIP manteve-se fixa em 10 usuários, e a população de usuários *web* foi variada de 10 a 200 usuários.

Foi mantida, para esta variação, uma granularidade de 10 usuários entre os pontos em que não há mudanças significativas nas curvas, e uma granularidade de 5 usuários nas partes das curvas onde o sistema passa a ficar saturado.

Considerando este cenário, foram avaliadas a vazão do sistema e o tamanho médio das filas dos canais diretos e dos canais reversos. Para os usuários *web* são calculados, também, a utilização das filas de ambos os canais e o tempo de acesso.

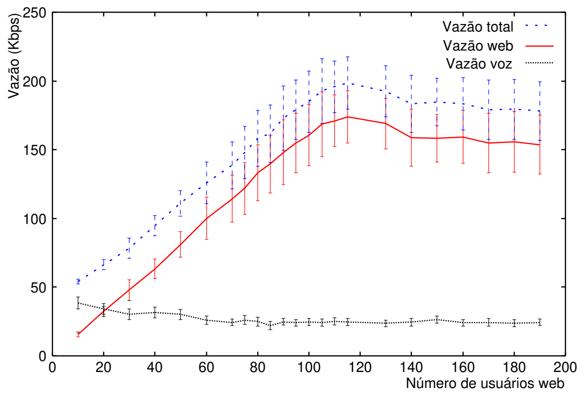


Figura 5.17: Vazão média do sistema

A Figura 5.17 apresenta a vazão média do sistema. Para isso, são apresentadas três curvas: (a) a vazão total do sistema; (b) a vazão dos usuários *web*; e (c) a vazão correspondente aos usuários de voz sobre IP. Como pode ser observado, com uma população de usuários *web* abaixo de 20, a maior parte da vazão do sistema corresponde aos usuários de voz. Isso acontece porque a taxa média de geração de dados dos usuários de voz é maior do que a dos usuários *web*. A partir de 20 usuários, a vazão de tráfego *web* corresponde a maioria do tráfego ao qual o sistema foi submetido. Observa-se, também, uma tendência de queda na vazão média dos usuários de voz causado pela disputa por canais, à medida que o sistema vai se tornando saturado. O ponto de saturação surgiu quando o sistema foi submetido a uma população de aproximadamente 120 usuários *web* e 10 de voz, onde a vazão ficou em torno de 200kbps. Ao continuar aumentando a carga, à qual o sistema era submetido, através de inserção de mais usuários *web*, observou-se uma tendência de queda em sua vazão total.

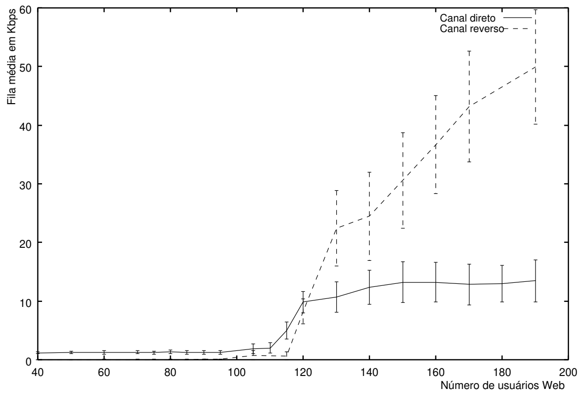


Figura 5.18: Fila média dos usuários *web*

Os tamanhos médios das filas dos canais diretos e reversos dos usuários *web* são mostrados na Figura 5.18. Observou-se que a fila do canal direto tem tamanho médio maior que o reverso enquanto o sistema não está saturado. Este comportamento é esperado já que, em aplicações *web*, o tráfego de retorno suportado por este canal envolve transferência de arquivos HTML e outro objetos como figuras ou arquivos de sons e vídeos, o que gera mais tráfego, quando comparado aos pedidos enviados pelos canais reversos. Quando o sistema se aproxima de seu ponto de saturação, de aproximadamente 120 usuários, a fila do canal reverso ultrapassa a do direto. Esta característica está ligada ao escalonamento de canais. Quando a carga do sistema começa a se aproximar da carga de pólo, pedidos de canais reversos começam a ser negados, já que os usuários tem menor quantidade de potência disponível do que a estação base e, com isso, a fila do canal reverso aumenta. Se usuários não conseguem canais para enviar pedidos *web*, conseqüentemente não haverá tráfego de retorno. O sistema, quando está saturado, passa a servir um número fixo de pedidos mesmo se a população de usuários é aumentada, o que faz com que o tráfego de retorno fique sob controle e não cause sobrecarga nas filas dos canais reversos.

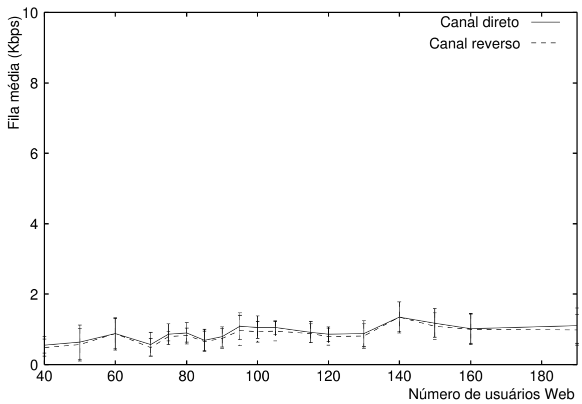


Figura 5.19: Fila média dos usuários de voz

Já as filas dos usuários de voz sobre IP, mostradas na Figura 5.19, não apresentaram mudanças significativas à medida que simulações eram executadas sob diferentes cargas. Este comportamento pode ser explicado pelo algoritmo de escalonamento de canais, que define prioridade dos canais fundamentais sobre os canais suplementares. Se ao receber um pedido de atribuição de canal fundamental, a estação base percebe que não há recursos no sistema disponível para atender o pedido, então um ou mais canais suplementares serão retirados aleatoriamente de um dos usuários que os estejam utilizando. Através deste processo, a carga é diminuída no intuito de atender o pedido em questão. Como a população de usuários *web* é muito maior que a de voz sobre IP, a probabilidade de que uma mensagem de cancelamento de atribuição de canal suplementar chegue a um usuário de voz sobre IP é pequena. Assim, estes usuários estariam sendo "privilegiados" pelo fato de estarem dispersos em uma população de usuários *web* bem maior do que a população de voz.

O gráfico da Figura 5.20 comprova o comportamento observado na Figura 5.18. A utilização da fila do canal reverso permanece menor que a fila do canal direto enquanto o sistema não se aproxima e ultrapassa o seu ponto de saturação. A partir do momento em que o sistema passa a ficar sobrecarregado, a utilização da fila do

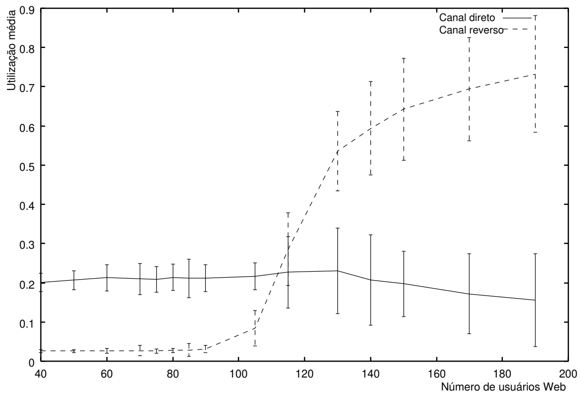


Figura 5.20: Utilização média das filas dos usuários *web*

canal direto cresce mais rapidamente e se aproxima de 75% enquanto que a utilização da fila do canal direto cai para 16%.

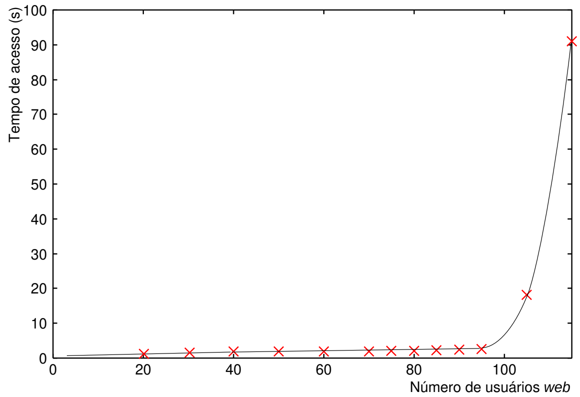


Figura 5.21: Tempo médio de acesso dos usuários *web*

Já a Figura 5.21, apresenta a média do tempo de acesso dos usuários *web*. Nota-se que o tempo de acesso permanece sob controle, entre 1 e 3 segundos, enquanto a população está abaixo de 95 usuários *web*. A partir deste momento, como o sistema começa a ficar sobrecarregado, o tempo de acesso cresce fora de controle, chegando a 91 segundos quando o sistema é submetido a uma população de 95 usuários *web*. Neste caso, o ideal seria não permitir a utilização do sistema por uma população maior que 100 usuários *web* e 10 de voz sobre IP.

Por fim, durante os experimentos observou-se que o sistema apresentou um RTT (*Round Trip Time*) da ordem de segundos, mesmo quando submetido a cargas relativamente baixas. Dado que o tempo de ida e volta máximo para tráfego de voz é de 400ms, conclui-se que a atual versão do sistema não parece ser adequada para este tipo de aplicação.

Capítulo 6

Conclusão

O aparecimento de novas tecnologias de acesso à Internet permitiu a disseminação de aplicações com alta interatividade como navegação *web* e voz sobre IP. Diante deste cenário, é crescente a necessidade de avaliação das novas propostas de mecanismos para acesso à rede de forma a prover a qualidade de serviço exigida pelas aplicações. Neste contexto, buscou-se neste trabalho, desenvolver mecanismos e modelos de simulação para suprir tal necessidade, através da identificação e avaliação dos limites críticos de operação de dois padrões de acesso à Internet, de acordo com a demanda de usuários.

A primeira questão abordada foi a modelagem do tráfego gerado pelas aplicações estudadas, requisito indispensável para a corretude e precisão dos resultados obtidos a partir dos modelos dos mecanismos de acesso considerados. Foram definidos modelos usando a ferramenta TANGRAM-II para representar: o comportamento do usuário *web*; e a transferência de dados entre o cliente e o servidor *web*. Nestes modelos, alguns eventos possuem distribuição de cauda longa. Portanto, foi necessário adicionar este tipo de distribuição à ferramenta TANGRAM-II. Outras funcionalidades necessárias para o desenvolvimento dos modelos também foram adicionadas e fazem parte da versão 2.0 da ferramenta lançada no início de 2003.

O próximo passo foi a modelagem dos dois mecanismos a serem estudados: o DOCSIS e o CDMA2000 1X. Devido à complexidade do padrão CDMA2000 1X, optou-se por uma modelagem híbrida, usando as técnicas de simulação de eventos discretos e simulação de fluido.

Uma vez que os modelos propostos foram implementados, vários testes de simulação foram executados para obter resultados visando avaliar os mecanismos. Os resultados das simulações do modelo do DOCSIS mostraram que: (1) os *slots* de disputa não são o gargalo do sistema, já que a vazão aumenta quando o número de *slots* de disputa por quadro é diminuído; (2) o aumento dos usuários de voz tem um impacto muito maior no tempo de acesso que o aumento dos usuários *web*; (3) é possível atender uma população de até 5000 usuários *web* e 1000 usuários de voz mantendo o tempo de acesso da ordem de poucos milisegundos.

O modelo do padrão CDMA2000 1X permitiu observar, por exemplo, que o gargalo do sistema quando este está sob altas cargas é o canal reverso, e não o canal direto. Foi possível observar, também, que o tempo de acesso do mecanismo para cargas baixas é da ordem de poucos segundos. Dado que, aplicações de voz sobre IP possuem, como requisito, o RTT de até 400ms, este resultado indica que o mecanismo do CDMA2000 1X não é apropriado para este tipo de aplicação.

6.1 Trabalhos Futuros

As avaliações dos mecanismos DOCSIS e CDMA2000 1X consideraram tráfego *web* e voz sobre IP. Pretende-se avaliar o desempenho destes mecanismos para outros tipos de tráfego como, por exemplo, o gerado por aplicações de *streaming* de áudio e vídeo. Além disso, outras medidas de interesse que dão uma noção melhor da QoS, como o *jitter*, poderão ser calculadas.

Como o canal de descida do DOCSIS não foi avaliado, os resultados obtidos são limites que supõem que o canal de descida não será o gargalo do sistema. Pretendemos modelar o canal de descida deste padrão para avaliar seu desempenho quando submetido às cargas utilizadas nestes trabalho.

Uma outra possibilidade de extensão deste trabalho consiste no estudo do sistema CDMA2000 1X considerando as seguintes características: (1) mobilidade dos usuários e (2) inteferência das células vizinhas.

Por fim, é necessário realizar ainda, na ferramenta TANGRAM-II, algumas modificações que facilitem o processo de utilização de um dos modelos propostos, que possui como característica, diversos objetos idênticos (usuários de voz e usuários *web*). Atualmente, estes objetos devem ser replicados pelo usuário da ferramenta. Pretende-se adicionar uma nova funcionalidade no TANGRAM-II, que permita a definição de um vetor de objetos. Desta forma, será possível a representação de um número grande de objetos iguais sem a necessidade de replicação dos mesmos.

Referências Bibliográficas

[1] Cable Labs. DOCSIS, Radio Frequency Interface Specification, December 2000. http://www.cable-modem.com.

[2] Ying-Dar Lin, Wei-Ming Yin, and Chen-Yu Huang. An investigation into HFC MAC protocols: Mechanisms, implementation an research issues. *IEEE - Communications Surveys*, pages 2–13, 2000.

[3] N.Golmie, F.Mouveaux, and D.Su. A comparison of MAC protocols for hybrid fiber/coax networks: IEEE 802.14 vs. MCNS. *IEEE*, pages 266–272, 1999.

[4] IS-2000, a Family of Standards for CDMA2000 Spread Spectrum System. TIA/EIA.

[5] N. Guo, F. Khaleghi, A. Guitierrez, J. Li, and M-H. Fong. Transmission of High Speed Data in CDMA2000. *IEEE - Vehicular Technology Conference*, pages 1442–1445, 1999.

[6] Douglas N. Knisely, Sarath Kumar, Subhasis Laha, and Sanjiv Nanda. Evolution of wireless data services: Is-95 to cdma2000. *IEEE - Communications Magazine*, pages 140–149, 1999.

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

[8] Rosa M.M. Leão and Edmundo de Souza e Silva. The TANGRAM-II Environment. In *Computer Performance Evaluation / TOOLS*, volume 1786 of *Lecture Notes in Computer Science*, pages 366–369. Springer, 2000.

[9] B. Liu and Y. Guo and J. Kurose and D. Towsley and Daniel. R. Figueiredo. A Study of Networks Simulation Efficency: Fluid Simulation vs. Packet-level Simulation. In *INFOCOM*, 2001.

[10] Ana Paula Couto da Silva, Rosa M.M. Leão, and Edmundo de Souza e Silva. Uma técnica de solução transiente para modelos de filas de fluido. *WPerformance - XXIII Congresso da Sociedade Brasileira de Computação*, pages 1613–1628, 2003.

[11] H. Richard Gail Edmundo de Souza e Silva and Reinaldo Vallejos Campos. Calculating Transient Distributions of Cumulative Reward. In *Proceedings of Sigmetrics*, pages 231–240, 1995.

[12] Rosa M.M. Leão, Edmundo de Souza, Silva, and Morganna C. Diniz. Traffic engineering using reward models. *XVII International Teletraffic Congress*, pages 1101–1126, 2001.

[13] Hédi Nabli and Bruno Sericola. Performability analysis for degradable computer systems. In *Computers and Mathematics with Applications*, volume 3-4, page 34, 2000.

[14] David Ros e Geraldo Rubino Raymond Marie, José Incera. FluidSim: A Tool to Simulate Fluid Models of High-Speed Network. In *Computer Performance Evaluation / TOOLS*, volume 1786 of *Lecture Notes in Computer Science*. Springer, 2000.

[15] Kelvin Reinhardt, Rosa M.M. Leão, and Edmundo de Souza. Um ambiente de simulação de fluido para redes multimídia. *SBRC*, pages 1101–1126, May 2003.

[16] URL http://www.land.ufrj.br/. Site do laboratório de modelagem, análise e desenvolvimento de redes e sistemas de computação.

[17] Shuang Deng. Empirical Model of WWW Document Arrivals at Access Link. In *Proceedings of IEEE ICC'96*, pages 1797–1802, 1996.

[18] Hyoung-Kee Choi and John O. Limb. "A Behavioral Model of Web Traffic". *International Conference of Network Protocols (ICNP)*, pages 327–334, 1999.

[19] M.T.Ali, R.Grover, G.Stamatelos, and David D.Falconer. Performance evaluation of candidate MAC protocols for LMCS/LMDS networks. *IEEE - Selected Areas in Communications*, 18(7):1261–1270, 2000.

[20] Gang Li, Ming Lu, Martin Meyers, Devesh Patel, James Stekas, and Andrea Tonello. Performance of lucent cdma2000 3g1x packet data experimental system. *Lucent*, pages 2665–2669, 2001.

[21] Lucent Technologies Inc. Engineering Guidelines. 401-614-040.

[22] James F. Kurose and Keith W. Ross. *Computer Networking: a Top-Down Approach Featuring the Internet (ISBN 0-201-47711-4)*. Addison-Wesley, 2002.

[23] N. Abramson. The Aloha System - Another Alternative for Computer Communications. In *AFIPS Conference Proceedings*, volume 36, pages 295–298. AFIPS, 1970.

[24] L. Kleinrock and F. Tobagi. Random access techniques for data transmission over packet switched radio channels. In *AFIPS Conference Proceedings, NNC*, pages 187–201. AFIPS, 1975.

[25] L. Roberts. Extensions of packet communication technology to a hand held personal terminal. In *Proc. Spring Joint Computer Conference*, pages 295–298. AFIPS, 1972.

[26] L. Kleinrock and F. Tobagi. Packet switching in radio channels/ part i: Carrier sense multiple-access modes and their throughput-delay characteristics. In *IEEE Transactions on Communications*, volume COM-23, pages 1400–1416. IEEE, December 1975.

[27] A. Viterbi. *CDMA: Principles of Spread Spectrum Communications*. Addison Wesley, 1995.

[28] Christian Bettstetter, Hans-Jörg Vögel, and Jörg Eberspächer. GSM phase 2+ general packet radio service GPRS: Architecture, protocols, and air interface. *IEEE Communication Surveys*, 2(3), 1999.

[29] Lucent Technologies Inc. CDMA 2G SIP Packet Data. 401-612-353.

[30] Lucent Technologies Inc. CDMA 3G-1X Packet Data Planning and Implementation Guide. 401-614-039.

[31] Lucent Technologies Inc. CDMA 3G-1X Data Feature. 401-612-388.

[32] Ana Lcia Iacono. Performance Evaluation of the Enhanced Access Channel in CDMA2000.

[33] Kamram Etemad. Enhaced Random Access and Reservation Scheme in CDMA2000. *IEEE Personal Communications*, 2001.

[34] Sarath Kumar and Sanjiv Nanda. Hight Data-Rate Packet Communications for Cellular Networks Using CDMA: Algorithms and Performance. *IEEE Selected Areas in Communications*, 17(3):472–492, 1999.

[35] Yu-Kwong Kwok and Vincent K. N. Lau. On Admission Control and Scheduling of Multimedia Burst Data for CDMA Systems. *Wireless Networks*, 8(5):495–506, 2002.

[36] Fundação Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD). Cenário de Simulacão Fim-a-Fim da rede Móvel CDMA2000/1xRTT. Technical Report.

[37] Venugopal V. Veeravalli and Andrew Sendonaris. The Coverage-Capacity Tradeoff in Cellular CDMA Systems. *IEEE Transactions on vehicular technology*, 1999.

[38] Radio Channel Impulse Response Measurement Systems, Jan 2000. Institute of Telecommunications Sciences (ITS) - Boulder Labs.

[39] William C.Y. Lee. *Mobile Communications Engineering*. New York: McGraw-Hill, 1982.

[40] Aleksandar Neskovic, Natasa Neskovic, , and George Punovic. Modern Approaches in Modeling of Mobile Radio Systems Propagation Environment. *IEEE Communications Surveys*, pages 2–12, 2000.

[41] S. Berson, E. de Souza Silva, and R. R. Muntz. Numerical solution of markov chains. chapter An Object Oriented Methodology for the Specification of Markov Models, pages 11–36. Marcel Dekker, Inc., 1991.

[42] M. C. Diniz and E. de Souza Silva. Especificação e Geração de Modelos Markovianos para Análise de Desempenho e Confiabilidade de Sistemas. *Revista Brasileira de Computação*, 6(3):23–42, 1991.

[43] Kelvin de Freitas Reinhardt. Um Ambiente de Simulação de Fluido com Aplicações a Redes Multimídia. Master's thesis, COPPE/UFRJ, 2002.

[44] Antonio Augusto de Aragão Rocha. Medições ativas na Internet: algoritmos baseados em retardo fim-a-fim e experimentos. Master's thesis, COPPE/UFRJ, Setembro 2003.

[45] Flávio Pimentel Duarte. Algoritmo Adaptativo para Previsão e Recuperação de Perda de Pacotes em Aplicações Multimídias Usando Cadeias de Markov Ocultas. Master's thesis, COPPE/UFRJ, Julho 2003.

[46] Jorge Allyson Azevedo. Transmissão Multicast Confiável: Aplicação Na Ferramenta TANGRAM Whiteboard e Experimentos na Internet. Master's thesis, NCE/UFRJ, Dezembro 2002.

[47] Sheldom M. Ross. *A Course in Simulation.* Macmillan Publishing Company, 1990.

[48] Kishor Shridharbhai Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Wiley-Interscience, 2002.

[49] Mark E. Crovella and Azer Bestavros. "Self-Similarity in World Wide Web Traffic". *IEEE/ACM Transactions on Networking*, 05:835–846, 1997.

[50] Masahiko Nabe and Masayuki Murata Hideo Miyahara. "Analysis and Modeling of World Wide Web Traffic for Capacity Dimensioning of Internet Access Lines". *Performance Evaluation*, 34:249–271, 1998.

[51] Daniel Ratton Figueiredo. O módulo de simulação da ferramenta TANGRAM-II: Suporte para medidas com recompensa, recursos de eventos raros e aplicação a modelos de redes multimídia. Master's thesis, COPPE/UFRJ, Junho 1999.

[52] Tamás Éltetõ and Sándor Molnár. On the Distribution of Round-trip Delays in TCP/IP Networks. In *24th Conference on Local Computer Networks*, page 172. IEEE, 1999.

[53] Mark E.Crovella and Lester Lipsky. Long-Lasting Transient Conditions in Simulations with Heavy-Tailed Workloads. In *Proceedings of the 1997 Winter Simulation Conference*, pages 1005–1012, 1997.

[54] ITU. Pulse code modulation (pcm) of voice frequencies, 1988.

[55] V.Sdralia, C.Smythe, P.Tzerefos, and S.Cvetkovic. Performance Characterization of the MCNS DOCSIS 1.0 CATV Protocol with Prioritised First Come First Served Scheduling. *IEEE - Transactions on Broadcasting*, 45(2):196–205, 1999.

[56] P.Tezerefos, V.Sdralia, C.Smythe, and S.Cvetkovic. Delivery of Low Bit Rate Isochronous Streams Over The DOCSIS 1.0 Cable Television Protocol. *IEEE - Transactions on Broadcasting*, 45(2), June 1999.

[57] Yong Bai, Pengfei Zhu, Ashok Rudrapatna, and Andy T. Ogielski. Performance of TCP/IP over IS-2000 Based CDMA Radio Links. *IEEE Vehicular Technology Conference*, pages 1036–1040, 2000.

[58] Lin Ma and Zhigang Rong. Capacity Simulations and Analysis for CDMA2000 Packet Data Services. *IEEE Vehicular Technology Conference*, pages 1620–1626, 2000.

[59] R. Thomas Derryberry, Lin Ma, and Zhigang Rong. Voice And Data Performance Of The CDMA2000 1XEV-DV System. *Submitted to ICC 2003*, pages 2665–2669, 2001. Nokia Research Center.

[60] 3GPP2. 1XEV-DV Evaluation Methodology - Addendum (V6). pages 2665–2669, July 2003. WG5 Evaluation Ad Hoc.

[61] Joe Huang, Yuqi Yao, Yong Bai, and Szu-Wei Wang. Performance of a Mixed-Traffic CDMA2000 Wireless with Scalable Streaming Video. *Submitted to IEEE Transctions on Cirtuits and Systems for Video Technology*, 2002.

[62] Weibo Gong, Yong Liu, Vishal Misra, and Don Towsley. On the Tails of Web File Size Distributions. 2002.

Apêndice A

Parâmetros Ajustáveis dos Modelos de Simulação

O objetivo desta seção é fornecer as informações necessárias para a utilização dos modelos propostos por este trabalho, bem como informar os parâmetros utilizados nos experimentos feitos durante o período de testes. Para isso, logo após o nome do parâmetro, optou-se por indicar, entre parênteses, um dos valores do parâmetro configurado nos modelos e a descrição de cada parâmetro.

A.1 DOCSIS

A.1.1 Usuários *Web*

Objeto *Delay_Off*

- offon_scale (60) - Parameter k of Pareto distribution used to model the Web user's OFF period;
- offon_shape (0.9) - Parameter α of Pareto distribution used to model the Web user's OFF period.

Objeto *Delay_ On*

- onoff_scale (90.017) - Parameter θ of Weibull distribution used to model the Web user's ON period;
- onoff_shape (0.88) - Parameter k of Weibull distribution used to model the Web user's ON period.

Objeto *Packet_ Generation*

- source_scale (4.4817) - Parameter θ of Weibull distribution used to model inter arrival time of the web user requisitions;
- source_shape (0.5) - Parameter k of Weibull distribution used to model inter arrival time of the web user requisitions;
- packet_size (5) - Packet size, in data *slots*. This value must be the same of the one set to the *Gated_ Queue* and *Backoff_ Delay* objects.

Objeto *Initializer*

- MAX_USERS (1000) - Total population of web users.
- INIT_ACTIVE_USERS (100) - Number of Web users initially active.

A.1.2 Canal de Acesso

Objeto *Gated_ Queue*

- p (0.125) - Probability that a given request arrive at one of the num_cs contention slots.
- packet_size (5) - Packet size, in data *slots*. This value must be the same of the one set to the *Packet_ Generation* and *Backoff_ Delay* objects.
- link_capacity ($7e + 5$) - Link Bandwidth, in bytes/s (5.6Mbps).

- max_queue (8000) - Queue size, in data slots.
- ds_size (80) - Data slot size, in bytes.
- cs_size (16) - Contention slot size, in bytes.
- max_ds (22) - Maximum number of data slots per frame.
- num_cs (8) - Number of contention slots per frame.
- number_voice_ds (2) - Number of data necessary to transmit each voice packet.
- max_frame (2.69) - Time in milisseconds to transmit a frame (24 data slots).
- population (1000) - Total number of voice users.
- time_between_packets (20) - Voice packet inter arrival time, in miliseconds.
- ugs_incoming_rate (0.0001851) - Unsolicited grant service requesting rate, in seconds.
- ugs_outgoing_rate (0.0055555) - Unsolicited grant service outgoing rate, in seconds.

Objeto *Backoff_Delay*

- link_capacity ($7e + 5$) - Link Bandwidth, in bytes/s (5.6Mbps).
- packet_size (5) - Packet size, in data *slots*. This value must be the same of the one set to the *Gated_Queue* and *Packet_Generation* objects.
- p (0.8) - Probability that a request is retransmitted in the next time slot.
- max_ds (22) - Maximum number of data slots per frame.

A.2 CDMA2000 1X

- RLP_INACTIVITY_TIMER (1) - Time, in seconds, that a Fundamental Channel is released if not used.
- BURST_TIMER (5.12) - Time, in seconds, that a Fundamental Channel is released.
- MEAN_REQUESTED_PACKET_SIZE (1.6) - Average size of web requests packets in kilobits.
- MEAN_MAIN_OBJECT_SIZE (83.672) - Size in Kilobits (Web Object) of main object of a Web site.
- MAIN_OBJECT_VARIANCE (3.82) - Variance of main object size.
- QUEUE_THRESHOLD (5) - When Queue, in packets, is above this value, a suplemental channel is requested.
- MEAN_INLINE_OBJECT_SIZE (60.609) - size in bits of inline Web Objects.
- INLINE_OBJECT_VARIANCE (9.71) - Variance of inline object size.
- ONOFF_SCALE (90.017) - Parameter θ of Weibull distribution used to model the Web user's ON period.
- ONOFF_SHAPE (0.88) - Parameter k of Weibull distribution used to model the Web user's ON period.
- OFFON_SCALE (60) - Parameter k of Pareto distribution used to model the Web user's OFF period.
- OFFON_SHAPE (0.9) - Parameter α of Pareto distribution used to model the Web user's OFF period.
- MAXIMUM (6000) - Value, in seconds, where pareto distribution will be truncated.

- SOURCE_SCALE (4.4817) - Parameter θ of Weibull distribution used to model the Web user's requisitions inter arrival time.

- SOURCE_SHAPE (0.5) - Parameter k of Weibull distribution used to model the Web user's requisitions inter arrival time.

- MEAN_DELAY (0.32935) - Mean packet delay (Gaussian Distribution), in seconds.

- DELAY_VARIANCE (0.0064513) - Variance equivalent to 0.08032 seconds of standard deviation (Gaussian Distribution).

- TIME_BETWEEN_PACKETS (0.02) - Time between voice packets generations , in seconds.

- VOICE_PACKET_SIZE (1.28) - Fixed size of voice packets in Kilobits (160 bytes).

- DATA_VOICE_INCOMING_RATE (0.0001852) - The user remains an average of 90 minutes on OFF state.

- DATA_VOICE_OUTGOING_RATE (0.0055555) - The user remains an average of 3 minutes on ON state.

- QUEUE_THRESHOLD (8) - When queue is above this value, a suplemental channel is requested (kbits), relative to 1000 bytes.

A.3 Controle de Admissão CDMA2000 1X

Objeto Base_Station

- MAX_CLIENTS (230) - The maximum number of clients allowed.

- FORWARD_THERMAL_NOISE (-194.5) - Receiver Noise Floor N0*F*W in dBm (Forward).

- REVERSE_THERMAL_NOISE (-194.5) - Receiver Noise Floor N0*F*W in dBm (Reverse).

- REVERSE_MAX_POWER (-9) - Maximum Transmitted power per traffic channel in dBm (Reverse), converted to dB.

- FORWARD_MAX_POWER (11) - Maximum Available power per traffic channel in dBm (Forward), converted to dB.

- HL (0) - Head Loss, in dB.

- FADE (10.3) - Fade Margin, in dB.

- BVL (0) - Building and Vehicle loss, in dB.

- AG (18) - Antenna gain, in dB.

- CL (3) - Cable loss, in dB.

- FS_PATH_LOSS (103.53) - Free space path loss relative to 1.8Km cell radius.

- Required_Eb_Nt $[4.0, 3.4, 2.6, 1.8, 1.0]$ - Required $\frac{E_b}{N_t}$ for channels of each rate 9.6kbps, 19.2kbps, 36.4kbps,76.8kbps and 153.6kbps respectively, in dB.

- Bit_Rate $[39.8, 42.8, 45.8, 48.9, 51.9]$ - Bit rate, converted to dB (Reverse Link), 9.6kbps, 19.2kbps, 36.4kbps,76.8kbps and 153.6kbps respectively.

- NMax $[40, 23, 20, 18, 17]$ - Pole Capacity of each rate in both directions of 9.6kbps, 19.2kbps, 36.4kbps,76.8kbps and 153.6kbps channels respectively.

Apêndice B

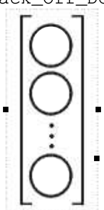
Exemplos do Código dos Modelos de Avaliação de Desempenho

O objetivo deste apêndice é fornecer informações adicionais da implementação dos modelos propostos por este trabalho, através da apresentação do código fonte de alguns dos objetos criados. Para tanto, optou-se por mostrar cada objeto separadamente e apresentar o código em forma de atributos distribuídos pelas figuras.

Os objetos *Gated_Queue* e *Back_Off_Delay* modelam o comportamento do canal de acesso do padrão DOCSIS e incluem a modelagem do tráfego gerado pelos usuários de voz.

Como exposto no Capítulo 4, o objeto *Base_Station* implementa o controle de admissão de canais CDMA e o objeto *Usuário Web* modela o comportamento de ambos os canais diretos e reversos de um usuário *Web*. Estes objetos são exemplos do modelo do padrão CDMA2000 1X.

```
name=Back_Off_Delay
```

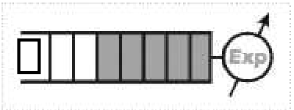


```
State_vars=
 Declaration=
   Var
    State:  retrying_users;
   Const
    Float:  link_capacity, p;
    Integer: packet_size, max_ds;
    Port: port_in,port_out;


Messages=
  msg_rec=port_in
  action= {
            int r;
            r = retrying_users;
            r++;
            clone_ev("Request_Delay");
            set_st("retrying_users", r);
          };
```

```
Initialization=
retrying_users     = 0
link_capacity      = 7e+5     /* bytes/s  (5.6Mbps) */
packet_size        = 5
p                  = 0.8
max_ds             = 22

port_in            = starting_delay
port_out           = upstream


Events=
 event=Request_Delay( EXP, 1/(max_ds/link_capacity ))
 condition=(retrying_users > 0)
 action= {
           int r;
           r = retrying_users;
           msg( port_out,all,0);
           r--;
           set_st("retrying_users",r);
         }: prob = p;
         {
           clone_ev("Request_Delay");
         }: prob = 1-p;

Rewards=
 rate_reward= average_retrying_users
 condition = (TRUE)
 value = retrying_users;

 impulse_reward= packet_collided
 event = Request_Delay, 1
 value = packet_size;
```

Figura B.1: Objeto Back_Off_Delay

```
name=G_queue
```

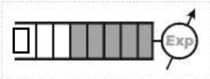


```
State_vars=

 Declaration=
   Var
    State: gate, token, queue, queue_vector[8],
           status, ugs_queue, ugs_users,
           ugs_generation_time, global_queue;
   Const
    Integer: max_queue, ds_size, cs_size, max_ds,
             num_cs, packet_size,
             number_voice_ds, population;
    Float: link_capacity, p, time_between_packets,
           ugs_outgoing_rate, ugs_incoming_rate,
           max_frame;
    Port:  port_in, contention_port_out;

Initialization=
 queue                = 0
 queue_vector         = [0,0,0,0,0,0,0,0]
 gate                 = 0
 ugs_queue            = 0
 ugs_users            = 0
 ugs_generation_time  = 0
 global_queue         = 0

 token                = 0
 status               = 0

 p                    = 0.125

 packet_size          = 5
 link_capacity        = 7e+5

 max_queue            = 8000
 ds_size              = 80
 cs_size              = 16
 max_ds               = 22
 num_cs               = 8
 number_voice_ds      = 2

 max_frame            = 2.6971428571
 population           = 1000
 time_between_packets= 20

 ugs_incoming_rate    = 0.0001851
 ugs_outgoing_rate    = 0.0055555

 port_in              = upstream
 contention_port_out = starting_delay
```

```
Messages=
  msg_rec=port_in
  action= {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[0] = q[0]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }:prob = p;
          {
           int q[num_cs];
            get_st(q,"queue_vector[]");
            q[1] = q[1]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
          {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[2] = q[2]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
          {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[3] = q[3]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
          {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[4] = q[4]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
          {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[5] = q[5]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
          {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[6] = q[6]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
          {
            int q[num_cs];
            get_st(q,"queue_vector[]");
            q[7] = q[7]+1;
            set_st("queue_vector[]",q);
            set_st("status",1);
          }: prob = p;
```

Figura B.2: Objeto Gated_Queue

name=G_queue




```
Events=
event= service(DET, ( 1/(ds_size/link_capacity) )/gate )
condition= ( (gate > 0) && (token == 1))
action={
        int ugs_q, q, global_q;

        q = queue;
        ugs_q = ugs_queue;

        if ( ugs_q <= max_ds )
        { /* voice and web traffic are served */
            ugs_q = 0;
            if ( max_ds - ugs_q <= q )
            /* not all web traffic is served */
            q = q - (max_ds - ugs_q);
            else q = 0;
            /* all web traffic is served */
        }
        else ugs_q = ugs_q - max_ds;
        /* only voice traffic is served */

        global_q = ugs_q+q;

        set_st("global_queue", global_q);
        set_st("queue", q);
        set_st("gate",  0);
        set_st("token", 0);
        set_st("ugs_queue", ugs_q);

      };

event= timeout(DET, 1/(num_cs*cs_size/link_capacity) )
condition= ( (token == 0)&&
             ((global_queue>0)||(status==1)) )

action={
        int t, g, q,global_q, vector[num_cs], i, j;

        t = token;
        get_st(vector,"queue_vector[]");
        q = queue;
        g = gate;

        for (i=0;i<=(num_cs-1);i++)  /* collision detection mechanism */
        {
            if (vector[i]>1)   /* collision detected, requests will wait */
            {
                for (j=1;j<=vector[i];j++)
                msg(contention_port_out,all,0);
        }
            else if (vector[i]==1) /* no collision, queue is incremented */
                if ( q+ugs_queue <= max_queue - packet_size )
                   q = q + packet_size;
        }
        for (i=0;i<=(num_cs-1);i++)  vector[i] = 0;
        if ( queue + ugs_queue <= max_ds )
            g = queue + ugs_queue;
        else g = max_ds;
        if (g > 0) t = 1;
        /* token is passed to service */

        global_q = ( ugs_queue+q );

        set_st("global_queue"  ,global_q);
        set_st("queue_vector[]",vector  );
        set_st("queue" ,q);
        set_st("token" ,t);
        set_st("status",0);
        set_st("gate"  ,g);

      };

event= outgoing_ugs_user(EXP, (ugs_users*ugs_outgoing_rate))
condition= ( ugs_users > 0 )
action={   /* duration of a voice call */
           int u, ugs_g_t;
           u = ugs_users;
           u--;

           /* ugs_g_t is multiplied by 1000
              to result in a truncated integer > 0 */
           if ( time_between_packets/u <= max_frame )
               ugs_g_t = max_frame;
           else ugs_g_t = time_between_packets/u;

           set_st("ugs_users", u);
           set_st("ugs_generation_time", ugs_g_t);
        };

event= ugs_request(DET, 1/(ugs_generation_time/1000.0) )
       /* the value ugs_generation_time is converted
          to sec */
condition= ( ugs_users > 0 )
action={
           int ugs_q, global_q;
           float aux_ugs_q;

           ugs_q    = ugs_queue   ;

           /*the ugs_queue is incremented at each
             ugs_generation_time by the number of
             previous voice requests generated.
             The minimum value of
             ugs_generation_time is 3ms. */

           aux_ugs_q = ugs_q +
                       (ugs_generation_time*ugs_users/
                 time_between_packets)*number_voice_ds;
           if (aux_ugs_q+queue<=max_queue)
              ugs_q = aux_ugs_q;
           else ugs_q = max_queue-queue;

event= incoming_ugs_user(EXP,(ugs_incoming_rate*
                              population))
condition= (TRUE)
action={       /* arrival of voice traffic */
           int u, ugs_g_t;
           u = ugs_users;
           u++;

           /* ugs_g_t is multiplied by 1000 to result
              in a truncated integer > 0 */
           if ( time_between_packets/u <= max_frame )
               ugs_g_t = max_frame;
           else ugs_g_t = time_between_packets/u;

           set_st("ugs_users", u);
           set_st("ugs_generation_time", ugs_g_t);
       };


           global_q = ( queue+ugs_q );

           set_st("global_queue",global_q);
           set_st("ugs_queue", ugs_q);
       };
```

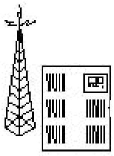
Figura B.3: Objeto Gated_Queue (Continuação)

name=Base_Station

```
Initialization=
Queue                  = 0         /* The number of packets that will be initially on queue*/
PORT                   = air       /* The name of the physical media used for transmission*/
ERROR_PORT             = error     /* msg is sent to this port to generate an error*/
ERROR_PORT2            = error2

Forward_Channels[]  = 0            /* The no. of for channels of each rate allocated to the clients*/
Reverse_Channels[]  = 0            /* The no. of rev channels of each rate allocated to the clients*/
MAX_CLIENTS                  = 80 /* The Maximum number of allowed clients*/
Client_Status[]              = 0  /* The channels allocated to each client */


                            /* Constants used on Simulation*/
                            /*CDMA2000's constants definitions for clearer protocol understanding */
FREE                        = 0
BUSY                        = 1
ID_ASSIGNMENT               = -1  /* ID ASSIGNMENT */
FUNDAMENTAL_RATE            = 8   /*  The rate of Fundamental channel in kbps*/

RATE1                       = 168 /*  The rate of supl channel in kbps*10 running on rate set 1*/
RATE2                       = 336 /*  The rate of supl channel in kbps*10 running on rate set 2*/
RATE3                       = 672 /*  The rate of supl channel in kbps*10 running on rate set 3*/
RATE4                       = 1440/*  The rate of supl channel in kbps*10 running on rate set 4*/

                                  /*The four variables bellow are used on reverse channel signaling*/
R_RATE1                     = 1680/*  The rate of rev supl channel in kbps*100 running on rate set 1*/
R_RATE2                     = 3360/*  The rate of rev supl channel in kbps*100 running on rate set 2*/
R_RATE3                     = 6720/*  The rate of rev supl channel in kbps*100 running on rate set 3*/
R_RATE4                     =14400/*  The rate of rev supl channel in kbps*100 running on rate set 4*/
                                  /*The four variables above are used on reverse channel signaling*/

MAX_OPENNED_SESSIONS        = 20/* under construction */
MAX_ACTIVE_SESSIONS         = 10/*  under construction*/

FUNDAMENTAL_CHANNEL_REQUEST  = 0/* Msg data id for the type of channel being requested  */
F_SUPLEMENTAL_CHANNEL_REQUEST= 1/* Msg data id for the type of channel being requested  */
FUNDAMENTAL_CHANNEL_RELEASE  = 4/* Msg data id for the type of channel being released   */
F_SUPLEMENTAL_CHANNEL_RELEASE= 5/* Msg data id for the type of channel being released   */
R_SUPLEMENTAL_CHANNEL_REQUEST= 6/* Msg data id for the type of channel being released   */
R_SUPLEMENTAL_CHANNEL_RELEASE= 7/* Msg data id for the type of channel being released   */


                                    /* Used to identify positions msg vector */
ID       = 0
MSG_DATA = 1

/* ----------------------   Admission Control Constants Initialization  -----   */

/* Receiver Noise Floor N0 * F * W in dBm  (Forward )*/
FORWARD_THERMAL_NOISE = -194.5

/* Receiver Noise Floor N0 * F * W in dBm  (Reverse )*/
REVERSE_THERMAL_NOISE = -194.5



/* Maximum Transmitted power per traffic channel in dBm ( Reverse ) */
REVERSE_MAX_POWER = -9 /* converted to dB */

/* Maximum Available power per traffic channel in dBm ( Forward ) */
FORWARD_MAX_POWER = 11 /* converted to dB */
```

Figura B.4: Objeto Base Station, atributo Initialization (Parte 1)

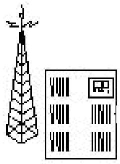
name=Base_Station



```
/* Path Loss Calculation Constants*/
HL   = 0
FADE = 10.3
BVL  = 0
AG   = 18
CL   = 3

/*Load of each diretion*/
Forward_Load = 0
Reverse_Load = 0
F_Load = 0
R_Load = 0
/*Distance from the clients to the base station, in meters (km)*/
FS_PATH_LOSS = 103.53 /*(dB) Free Space Path Loss relative to 1.8Km cell radius */


/* -------------------------- End of Admission Control Constants Declaration  */

/* Required Eb/Nt in dB  (Reverse Link)*/
/* Here, we're initializing  the Required Eb/Nt for channels of each rate*/
/* 9.6kbps, 19.2kbps, 36.4kbps,76.8kbps and 153.6kbps respectively*/
Required_Eb_Nt = [4.0,3.4,2.6,1.8,1.0]

/* Information Rate in dB (Reverse Link)*/
/* 9.6kbps, 19.2kbps, 36.4kbps,76.8kbps and 153.6kbps respectively*/
Bit_Rate = [ 39.8, 42.8, 45.8, 48.9, 51.9 ]

/* Maximum Number of Supported Channels of each rate (Both directions) NMAX*/
/* 9.6kbps, 19.2kbps, 36.4kbps,76.8kbps and 153.6kbps respectively*/
NMax = [ 35, 23, 20, 18, 17 ]

F_Rate = [ 8 ,168, 336, 672, 1440 ]
R_Rate = [ 8 ,1680, 3360, 6720, 14400 ]
```

Figura B.5: Objeto Base Station, atributo Initialization (Parte 2)

```
name=Base_Station
```



```
Messages=
 msg_rec=PORT
 action= {
int r_load, f_load; /*FIXME*/
int forward_channels[5];
int reverse_channels[5];
int     msg_vec[2], source_id;
int     client_stat[MAX_CLIENTS];
/* ----------- Admission Control local variables below --------- */
float receiver_sensibility,
      forward_load, reverse_load, load,
      total_attenuation, remaining_power;
int   channel_assigned, higher_rate, higher_rate_client_id;
int   aux_msg_vec[2], i, j;
int   assignment_failure;
int r_rate[5], f_rate[5];
channel_assigned=0;
forward_load = Forward_Load;
reverse_load = Reverse_Load;
load = forward_load+reverse_load;
/* ----------- Admission Control local variables above --------- */
get_st( r_rate          , "R_Rate[]"           );
get_st( f_rate          , "F_Rate[]"           );
get_st( forward_channels, "Forward_Channels[]" );
get_st( reverse_channels, "Reverse_Channels[]" );
get_st( client_stat     , "Client_Status[]"    );
get_msg_data( msg_vec );
source_id = msg_vec[ ID ];
/*If Client id (msg_vec[0]) is 0, then this client does not have an <id> yet */
/*       and an <id> needs to be allocated to this client                    */
if( msg_vec[ID] == 0 )
{
    int id_status;
    id_status = FREE;
    i = 0;
    while  ( ( i <= MAX_CLIENTS )&&( id_status == FREE ) )
    {
       /*'i' is initialized to 1 because a mobile <id> set to 0 means there is
         no <id> allocated*/
       i++;
       if( client_stat[i] == FREE )
       {
           /*If this "i" <id> is not allocated then it can be allocated to the
             source mobile*/
           source_id      = i;
           client_stat[i] = BUSY;
           id_status      = BUSY;
           msg_vec[ID]    = source_id;
       }
       else if ( i == MAX_CLIENTS )
            {
                /* If no free id is found, then this message generates*/
                /* an intended error in simulation */
                msg( ERROR_PORT, all, 0 );
            }
    }
}
```

Figura B.6: Objeto Base Station, atributo Messages (Parte 1)

```
name=Base_Station
```

```
Messages=
switch( msg_vec[ MSG_DATA ] )
{
    case FUNDAMENTAL_CHANNEL_REQUEST:
         msg_vec[MSG_DATA]      = f_rate[0];
         channel_assigned = 0;
         /* -------------- Admission Control Calculations and Decisions below*/
         assignment_failure = 0;
          while( ( channel_assigned < 1 )&&( assignment_failure == 0 ) )
          {
              channel_assigned = 0;
              if( (forward_load < 0.94)&&(forward_load+reverse_load < 0.94) )
              {
                  receiver_sensibility = ( Required_Eb_Nt[0]+Bit_Rate[0]+
                                         (10*log10(1/(1-( (float)load + (float)1/(float)NMax[0])))) )
                                         +FORWARD_THERMAL_NOISE );
                  total_attenuation = ( HL + FADE + BVL + FS_PATH_LOSS - AG + CL );
                  remaining_power = ( pow( 10, FORWARD_MAX_POWER/10-3 ) / pow( 10, total_attenuation/10 ) );
                      if( 10*log10( remaining_power  ) >= receiver_sensibility  )
                      {
                         channel_assigned++;
                         if( (reverse_load < 0.94)&&(forward_load+reverse_load < 0.94) )
                         {
                             receiver_sensibility = ( Required_Eb_Nt[0]+Bit_Rate[0]+
                                                    ( 10*log10(1/(1-( (float)load + (float)1/(float)NMax[0]))) )
                                                       +REVERSE_THERMAL_NOISE );
                             total_attenuation = ( HL + FADE + BVL + FS_PATH_LOSS - AG + CL );
                             remaining_power = ( pow( 10, REVERSE_MAX_POWER/10-3 ) /
                                                 pow( 10, total_attenuation/10 ) );
                             if( 10*log10( remaining_power  ) >= receiver_sensibility  )
                             {   /*Fundamental Channel Assignment*/
                                 channel_assigned++;
                                 forward_load = ( (float)forward_load + (1/(float)NMax[0])/2 );
                                 reverse_load = ( (float)reverse_load + (1/(float)NMax[0])/2 );
                                 forward_channels[0]++;
                                 reverse_channels[0]++;
                                 client_stat[source_id] = client_stat[source_id] + f_rate[0];
                                 msg(PORT,msg_source,msg_vec);
                             }
                         }
                      }
              } /* closing the if( forward_load < 0.94)     */
              /*  If there is not enough system resource for a fundamental channel  */
              /*   a FORWARD(channel_assigned=0) or REVERSE(channel_assigned=1)      */
              /*   suplemental channel will have to be released so a fundamental    */
              /*                     channel can be assigned                        */
              if( channel_assigned <= 1 )
              {
                float fund_rate;
                higher_rate = 0;
                higher_rate_client_id = 0;
                /*Looking for the client using some kind of channel*/
                for( i = 1; ( client_stat[i] != 0 ); i++ )
                {
                    if( channel_assigned == 0)
                    {
                         fund_rate = f_rate[0];
                         aux_msg_vec[1] = F_SUPLEMENTAL_CHANNEL_RELEASE;
                    }
                    else
                    {
                        fund_rate = r_rate[0];
                        aux_msg_vec[1] = R_SUPLEMENTAL_CHANNEL_RELEASE;
                    }
                        /*This is the higher rate channel AND This isn't a fundamental channel*/
                    if( ( higher_rate < client_stat[i]-1 )&&( client_stat[i]-1 > fund_rate) )
                    {
                        higher_rate_client_id = i;
                        higher_rate = client_stat[higher_rate_client_id]-1;
                    }
                }
```

Figura B.7: Objeto Base Station, atributo Messages (Parte 2)

name=Base_Station

```
aux_msg_vec[0] = higher_rate_client_id;
                /*If there is someone using a suplemental channel*/
                if( higher_rate > fund_rate )
                {
                    /*Here we're finding out which suplemental channels are alocated to this user*/
                    /*Forward Suplemental channel: f_rate[1,2,3,4]=[19.2,38.4,76.8,153.6]kbps*/
                    for( i = 4; ( i > 0 )&&( f_rate[i] != higher_rate - r_rate[j] - 1 ) ; i-- )
                    {
                        /*Reverse Supl. channel:r_rate[1,2,3,4]=[19.2,38.4,76.8,153.6]kbps*/
                        for( j = 4; ( j > 0 )&&( f_rate[i] != higher_rate - r_rate[j] - 1 ); j-- );
                        if( f_rate[i] == higher_rate - r_rate[j] - 1 )
                            i++;
                    }
                    if( channel_assigned == 0)
                    {
                        forward_channels[i]--;
                        forward_load = ( (float)forward_load - (float)1/(float)NMax[i]);
                        client_stat[higher_rate_client_id] = client_stat[higher_rate_client_id]
                                                                        - f_rate[i];
                    }
                    else
                    {
                        reverse_channels[j]--;
                        reverse_load = ( (float)reverse_load - (float)1/(float)NMax[j] );
                        client_stat[higher_rate_client_id] = client_stat[higher_rate_client_id]
                                                                        - r_rate[j];
                    }
                    msg(PORT,all,aux_msg_vec);
                } /* Closes the condition if( higher_rate > fund_rate )*/
                else /* There is no system resource for a new fundamental channel */
                {
                    assignment_failure = 1;
                    msg_vec[MSG_DATA] = ID_ASSIGNMENT;
                    msg(PORT,msg_source,msg_vec); /* ID Assignment to the client */
                }
            } /* Closes the condition if( channel_assigned < = 1 ) */
          } /*closes the while( channel_assigned < 1 )  */

         /* -------------- Admission Control Calculations and Decisions above ------------ */
         break;

    case F_SUPLEMENTAL_CHANNEL_REQUEST:

         /* -------------- Admission Control Calculations and Decisions below ------------ */
             if ( (forward_load > 0.94)&&( load < 0.94 ) )
             {
                 for( i = 4;(channel_assigned == 0 )&&( i > 0 );i-- )
                 {
                      receiver_sensibility = ( Required_Eb_Nt[i]+Bit_Rate[i]+
                                               ( 10*log10(1/(1-( (float)load + (float)1/(float)NMax[i]))) )
                                                 +FORWARD_THERMAL_NOISE );
                      total_attenuation = ( HL + FADE + BVL + FS_PATH_LOSS - AG + CL );
                      remaining_power = ( pow( 10, REVERSE_MAX_POWER/10-3 ) /
                                          pow( 10, total_attenuation/10 ) );
                      /* If there is enough power for this channel then this channel will be assigned*/
                      if( 10*log10( remaining_power ) >= receiver_sensibility  )
                      {
                             msg_vec[ MSG_DATA ] = f_rate[i];
                             forward_channels[i]++;
                             client_stat[source_id] = client_stat[source_id] + f_rate[i];
                             msg(PORT,msg_source,msg_vec);
                             forward_load = ( (float)forward_load + (float)1/(float)NMax[i]);
                             channel_assigned = 1;
                      }
                 }
             }
         /* -------------- Admission Control Calculations and Decisions above -------------- */
         break;
```

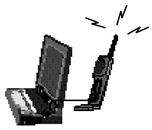
Figura B.8: Objeto Base Station, atributo Messages (Parte 3)

name=Base_Station

```
    case R_SUPLEMENTAL_CHANNEL_REQUEST:

         /* -------------- Admission Control Calculations and Decisions below -------------- */
         if ( (reverse_load > 0.94)&&( load < 0.94 ) )
         {
             for( i = 4;( channel_assigned == 0 )&&( i > 0 );i-- )
             {
                 receiver_sensibility = ( Required_Eb_Nt[i]+Bit_Rate[i]+
                                          ( 10*log10(1/(1-( (float)load +
                                          (float)1/(float)NMax[i]))) )+REVERSE_THERMAL_NOISE );
                 total_attenuation = ( HL + FADE + BVL + FS_PATH_LOSS - AG + CL );
                 remaining_power = ( pow( 10, REVERSE_MAX_POWER/10-3 ) /
                                     pow( 10, total_attenuation/10 ) );
                 /* If there is enough power for a 153.6kbps channel then this
                    153.6kbps channel will be assigned*/
                 if( 10*log10( remaining_power ) >= receiver_sensibility  )
                 {
                        msg_vec[ MSG_DATA ] = r_rate[i];
                        reverse_channels[i]++;
                        client_stat[source_id] = client_stat[source_id] + r_rate[i];
                        msg(PORT,msg_source,msg_vec);
                        reverse_load = ( (float)reverse_load + (float)1/(float)NMax[i]);
                        channel_assigned = 1;
                 }
             }
         }
         /* -------------- Admission Control Calculations and Decisions above -------------- */
         break;

    default : /* This is a channel releasement report */
         /* Now we need to find what kind of channel is beeing released */
         for( i=0;( i<=5 )&&( r_rate[i]!=msg_vec[MSG_DATA] )
                          &&( f_rate[i]!=msg_vec[MSG_DATA] ); i++ );
         /* If this message reports a reverse channel releasement */

         /* If this message reports a reverse channel releasement */
         if( r_rate[i] == msg_vec[MSG_DATA]  )
             {
                 client_stat[source_id] = client_stat[source_id] - r_rate[i];
                 if( i == 0)
                     reverse_load = ( (float)reverse_load - (1.0/NMax[i])/2 );
                 else
                     reverse_load = ( (float)reverse_load - (1.0/NMax[i]) );
                 reverse_channels[i]--;
             }
         /* If this message reports a forward channel releasement */
         if( f_rate[i] == msg_vec[MSG_DATA]  )
              {
                  if( i != 0 )
                  {
                      client_stat[source_id] = client_stat[source_id] - f_rate[i];
                      forward_load = ( (float)forward_load - (1.0/NMax[i]) );
                  }
                  else
                      forward_load = ( (float)forward_load - (1.0/NMax[i])/2 );
                  forward_channels[i]--;
              }
         if( i == 5 ) /* The message received does not follows the protocol */
              msg( ERROR_PORT2, all, 0 );
}
r_load = reverse_load*10000;
f_load = forward_load*10000;
set_st("Forward_Channels[]",forward_channels);
set_st("Reverse_Channels[]",reverse_channels);
set_st("Client_Status[]",client_stat);
set_st("Forward_Load",forward_load);
set_st("Reverse_Load",reverse_load);
set_st("R_Load",r_load);
set_st("F_Load",f_load);
         };
```

Figura B.9: Objeto Base Station, atributo Messages (Parte 4)

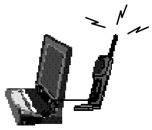
Usuário_Web



```
Messages=msg_rec=PORT
action=
{
    int fch, fsupl_stat,fsupl_rate, rsupl_stat, rsupl_rate, id;
    int msg_vec[2];
    fch         =  Fundamental_Channel_Status;
    fsupl_stat  =  F_Suplemental_Channel_Status;
    fsupl_rate  =  F_Suplemental_Channel_Rate;
    rsupl_stat  =  R_Suplemental_Channel_Status;
    rsupl_rate  =  R_Suplemental_Channel_Rate;
    id          =  Id;
    get_msg_data(msg_vec);
    if( id == 0 ) /*If I have no <id>, so the base station is giving me one*/
        id       = msg_vec[0];
    if (id == msg_vec[0])
    {
        switch( msg_vec[1] )
        {
            case FCH_RATE:
                 /*Msg from base station responding an earlier Fund. Channel request*/
                 fch = ACTIVE;
                 break;
            case RATE1:
                 /*Msg from B.S. containning a 19.2kbps F_Suplem. Channel  assignment*/
                 fsupl_stat   = ACTIVE;
                 fsupl_rate   = msg_vec[1];
                 break;
            case RATE2:
                 /*Msg from B.S. containning a 38.4kbps F_Suplem. Channel  assignment*/
                 fsupl_stat   = ACTIVE;
                 fsupl_rate   = msg_vec[1];
                 break;
            case RATE3:
                 /*Msg from B.S. containning a 76.8kbps F_Suplem. Channel  assignment*/
                 fsupl_stat   = ACTIVE;
                 fsupl_rate   = msg_vec[1];
                 break;
            case RATE4:
                 /*Msg from B.S. containning a 153.6kbps F_Suplem. Channel  assignment*/
                 fsupl_stat   = ACTIVE;
                 fsupl_rate   = msg_vec[1];
                 break;
```

Figura B.10: Objeto Web_Data_Client, atributo Messages (Parte 1)

Usuário_Web



```
            case R_RATE1:
                 /*Msg from B.S. containning a 19.2kbps F_Suplem. Channel  assignment*/
                 rsupl_stat   = ACTIVE;
                 rsupl_rate   = msg_vec[1];
                 break;
            case R_RATE2:
                 /*Msg from B.S. containning a 38.4kbps F_Suplem. Channel  assignment*/
                 rsupl_stat   = ACTIVE;
                 rsupl_rate   = msg_vec[1];
                 break;
            case R_RATE3:
                 /*Msg from B.S. containning a 76.8kbps F_Suplem. Channel  assignment*/
                 rsupl_stat   = ACTIVE;
                 rsupl_rate   = msg_vec[1];
                 break;
            case R_RATE4:
                 /*Msg from B.S. containning a 153.6kbps F_Suplem. Channel  assignment*/
                 rsupl_stat   = ACTIVE;
                 rsupl_rate   = msg_vec[1];
                 break;
            case FUNDAMENTAL_CHANNEL_RELEASE:
                 /*Msg from base station reporting the Fundamental Channel Termination*/
                 fch = INACTIVE;
                 set_cr(Session_Duration_RW, 0);
                 break;
            case F_SUPLEMENTAL_CHANNEL_RELEASE:
                 /*Msg from base station reporting the F_Suplemental Channel Termination*/
                 fsupl_stat   = INACTIVE;
                 fsupl_rate   = 0;
                 break;
            case R_SUPLEMENTAL_CHANNEL_RELEASE:
                 /*Msg from base station reporting the R_Suplemental Channel Termination*/
                 rsupl_stat   = INACTIVE;
                 rsupl_rate   = 0;
                 break;
        }
    }
    set_st(  "Fundamental_Channel_Status",fch         );
    set_st("F_Suplemental_Channel_Status",fsupl_stat  );
    set_st("F_Suplemental_Channel_Rate"  ,fsupl_rate  );
    set_st("R_Suplemental_Channel_Status",rsupl_stat  );
    set_st("R_Suplemental_Channel_Rate"  ,rsupl_rate  );
    set_st(            "Id"              ,id          );
};
```

Figura B.11: Objeto Web_Data_Client, atributo Messages (Parte 2)

Usuário_Web

```
Events=
event=Base_Station_Packet_Delay(GAUSS,MEAN_DELAY,DELAY_VARIANCE)
condition= ( Delaying_Packets > 0 )
action=
{
    int d_packets, f_queue, fsch, fch;
    int msg_vec[2];
    /*Position 0 stores the ID and the position 1 stores the msg_data*/
    fsch      = F_Suplemental_Channel_Status;
    fch       = Fundamental_Channel_Status;
    d_packets = Delaying_Packets;
    f_queue   = Forward_Queue;
    d_packets--;
    f_queue++;
    if  ( f_queue > 0 )
    {/*If the fundamental channel is not active, then a fundamental channel is requested*/
        if  ( fch == INACTIVE  )
        {
            msg_vec[ID] = Id;
            msg_vec[MSG_DATA] = FUNDAMENTAL_CHANNEL_REQUEST;
            msg(PORT,BASE_STATION,msg_vec);
            fch = WAITING_CHANNEL_REPLY;
        }
        /*Else If the fundamental channel is not waiting for reply, and the queue is above
         threshold and the F_Suplemental channel is inactive, then a F_Suplemental channel
         is requested*/
        else if     ( ( f_queue > 0 )&&( fch == ACTIVE )&&( fsch == INACTIVE ) )
        {
            msg_vec[ID]       = Id;
            msg_vec[MSG_DATA] = F_SUPLEMENTAL_CHANNEL_REQUEST;
            msg(PORT,BASE_STATION,msg_vec);
            fsch = WAITING_CHANNEL_REPLY;
        }
    }
    set_st( "Fundamental_Channel_Status"  , fch            );
    set_st( "F_Suplemental_Channel_Status", fsch           );
    set_st( "Delaying_Packets"            , d_packets      );
    set_st( "Forward_Queue"               , f_queue        );
};
```

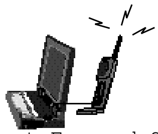
Figura B.12: Objeto Web_Data_Client, atributo Events (parte 1)

Usuário_Web

```
event=Base_Station_Service(LOGNORM, Mean_Packet_Size/
                           (Fundamental_Channel_Status*FCH_RATE*Forward_Queue/
                           (Forward_Queue+Reverse_Queue+EPSILON))+F_Suplemental_Channel_Rate/
                            10.0, Packet_Size_Variance/pow((Fundamental_Channel_Status*
                            FCH_RATE*Forward_Queue/(Forward_Queue+Reverse_Queue+EPSILON))+
                            F_Suplemental_Channel_Rate/10.0,2))
condition= ( (Forward_Queue > 0)&&( Fundamental_Channel_Status == ACTIVE ) )
action=
{
    int f_queue, fsupl_stat, fsupl_rate;
    int msg_vec[2]; /*Position 0 stores the ID and the position 1 stores the msg_data*/
    float mean_packet_size, packet_size_variance;
    int vec[2];
    IntegerQueue reverse_queue_status(2);
    get_st(reverse_queue_status,"Reverse_Queue_Status");
    restore_from_head(reverse_queue_status, vec);
    restore_from_head(reverse_queue_status, vec);
    if( vec[0] == 1)
    {/* If the next object to be served is the first obj of an incoming web page */
     /*Proximo tamanho de objeto serah de um main object*/
        mean_packet_size = MEAN_MAIN_OBJECT_SIZE;
        packet_size_variance = MAIN_OBJECT_VARIANCE;
    }
    else
    {
        /*Proximo tamanho de objeto serah de um main object*/
        mean_packet_size = MEAN_INLINE_OBJECT_SIZE;
        packet_size_variance = INLINE_OBJECT_VARIANCE;
    }
    save_at_head(reverse_queue_status, vec);

    f_queue = Forward_Queue;                  /*Local var getting state var's value*/
    fsupl_stat = F_Suplemental_Channel_Status;/*Local var getting state var's value*/
    fsupl_rate = F_Suplemental_Channel_Rate;  /*Local var getting state var's value*/
    f_queue--;
    if  ( ( f_queue == 0) && ( Status == INACTIVE ) && ( fsupl_stat == ACTIVE ) )
    {/*If the queue becomes empty and no data will be generated, then the
       F_Supl channel is released */
        msg_vec[ID]       = Id;
        msg_vec[MSG_DATA] = fsupl_rate;
        msg(PORT,BASE_STATION,msg_vec);
        /* Send message to base to report a F_Suplemental channel of rate
           fsupl_rate/10 releasement*/
        fsupl_stat = INACTIVE;
        fsupl_rate = 0;
    }
    set_st( "Mean_Packet_Size"            ,mean_packet_size);
    set_st( "Packet_Size_Variance"        ,packet_size_variance);
    set_st( "F_Suplemental_Channel_Status",fsupl_stat);
    set_st( "F_Suplemental_Channel_Rate"  ,fsupl_rate);
    set_st( "Forward_Queue"               ,f_queue);
    set_st( "Reverse_Queue_Status"        ,reverse_queue_status );
    set_st( "Mean_Packet_Size"            ,mean_packet_size);
    set_st( "Packet_Size_Variance"        ,packet_size_variance);
};
```

Figura B.13: Objeto Web_Data_Client, atributo Events (parte 2)

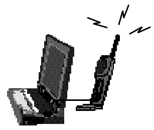
Usuário_Web



```
event=Forward_Suplemental_Channel_Request_Retransmition(DET,1.0/2.0)
 condition= (    ( Forward_Queue > 0 )&&( Fundamental_Channel_Status == ACTIVE )
             &&( F_Suplemental_Channel_Status == WAITING_CHANNEL_REPLY ) )
 action= {
    int status,msg_vec[2];/*Pos 0 stores the ID and the pos 1 stores the msg_data*/
    msg_vec[ID]       = Id;
    msg_vec[MSG_DATA] = F_SUPLEMENTAL_CHANNEL_REQUEST;
    msg(PORT,BASE_STATION,msg_vec);
    status = WAITING_CHANNEL_REPLY;
    set_st( "F_Suplemental_Channel_Status",status);
};
/*-------------------------------------------------------------------------------------*/
 event= On_Off( WEIB, ONOFF_SCALE, ONOFF_SHAPE )
 condition= ( Status == ACTIVE )
 action= {
            int stat;
            stat  = INACTIVE;
            set_st( "Status",stat );
        };
/*-------------------------------------------------------------------------------------*/
 event= Off_On( TRUNCPAR, OFFON_SCALE, OFFON_SHAPE, MAXIMUM )
 condition= ( Status == INACTIVE )
 action= {
            int stat;
            stat = ACTIVE;
            set_st( "Status", stat );
            set_st( "First_Object", 1);
        };
/*-------------------------------------------------------------------------------------*/
event=Packet_Generation( WEIB, SOURCE_SCALE, SOURCE_SHAPE )
condition= ( Status == ACTIVE )
action= {
    int r_queue, fch, rsch;
    int msg_vec[2]; /*Pos 0 stores  the ID and the pos  1 stores the msg_data   */
    int vec[2];
    int first_object;
    IntegerQueue reverse_queue_status(2);
    first_object = First_Object;
    get_st(reverse_queue_status,"Reverse_Queue_Status");
    vec[0] = first_object; vec[1] = 0;
    save_at_tail(reverse_queue_status, vec);
    if( first_object == 1 ) first_object = 0;
    r_queue = Reverse_Queue;
    r_queue++;
    fch = Fundamental_Channel_Status;
    rsch = R_Suplemental_Channel_Status;
    if  ( r_queue > 0 )
    {   /*If the fundamental channel is not active, then a fund channel is requested*/
        if  ( fch == INACTIVE  )
        {
            msg_vec[ID]       = Id;
            msg_vec[MSG_DATA] = FUNDAMENTAL_CHANNEL_REQUEST;
            msg(PORT,BASE_STATION,msg_vec);
            fch = WAITING_CHANNEL_REPLY;
        }
        else if( ( r_queue>QUEUE_THRESHOLD )&&( fch==ACTIVE )&&( rsch==INACTIVE ) )
        {
            msg_vec[ID]       = Id;
            msg_vec[MSG_DATA] = R_SUPLEMENTAL_CHANNEL_REQUEST;
            msg(PORT,BASE_STATION,msg_vec);
            rsch = WAITING_CHANNEL_REPLY;
        }
    }
    set_st( "Reverse_Queue", r_queue );
    set_st( "Fundamental_Channel_Status",fch );
    set_st( "R_Suplemental_Channel_Status",rsch );
    set_st( "Reverse_Queue_Status",reverse_queue_status );
    set_st( "First_Object",first_object );
};
```

Figura B.14: Objeto Web_Data_Client, atributo Events (parte 3)

Usuário_Web



```
event=Packet_Transmission(EXP, ( (float)Fundamental_Channel_Status*FCH_RATE*Reverse_Queue
                                /(float)(Reverse_Queue+Forward_Queue+EPSILON)
                                +R_Suplemental_Channel_Rate/100.0)/
                                MEAN_REQUESTED_PACKET_SIZE )

condition= ( ( Reverse_Queue > 0 )&&( Fundamental_Channel_Status == ACTIVE )  )
           /*A packet service event can occur only if there is some packet on queue, if
           there is at least a fundamental channel assigned to this client and if the
           number of allowed packets to be transmitted within this channel had not been
           reached yet*/
 action=
{
    int r_queue, rsupl_stat, rsupl_rate, d_packets;
    int msg_vec[2];/*Position 0 stores the ID and the pos 1 stores the msg_data*/
    r_queue = Reverse_Queue;/*Queue is decrem. and new value stored at local var*/
    rsupl_stat = R_Suplemental_Channel_Status;/*Local var gets state var's value*/
    rsupl_rate = R_Suplemental_Channel_Rate;  /*Local var gets state vari's value*/
    d_packets  = Delaying_Packets;
    clone_ev("Base_Station_Packet_Delay");/*Pkt is transm. and delayed for response*/
    r_queue--;
    d_packets++;
    if  ( ( r_queue == 0 ) && ( Status == INACTIVE ) && ( rsupl_stat == ACTIVE )  )
    {
        msg_vec[ID]       = Id;
        msg_vec[MSG_DATA] = rsupl_rate;
        /*Send message to base to report a R_Supl channel of rate rsupl_rate release*/
        msg(PORT,BASE_STATION,msg_vec);
        rsupl_stat = INACTIVE;
        rsupl_rate = 0;
    }
    set_st( "Reverse_Queue", r_queue );
    set_st( "R_Suplemental_Channel_Status", rsupl_stat);
    set_st( "R_Suplemental_Channel_Rate"  , rsupl_rate);
    set_st( "Delaying_Packets"            , d_packets);
};
/*--------------------------------------------------------------------------------*/
event=Fundamental_Channel_Release(DET,1.0/RLP_INACTIVITY_TIMER)
condition= ( ( Forward_Queue == 0 )&&( Reverse_Queue == 0)&&
             ( Fundamental_Channel_Status == ACTIVE ) )
action=
{
    int fch;
    int msg_vec[2]; /*Position 0 stores the ID and the pos 1 stores the msg_data*/
    fch = INACTIVE;
    msg_vec[ID] = Id;
    msg_vec[MSG_DATA] = FCH_RATE;
    set_cr(Session_Duration_RW, 0);
    msg(PORT,BASE_STATION,msg_vec);
    /*Send message to base to report a Fundamental channel release*/
    set_st( "Fundamental_Channel_Status",fch);
};
/*------------------------------------------------------------------*/
event=R_Suplemental_Channel_Release(DET,1.0/BURST_TIMER)
condition= (  R_Suplemental_Channel_Status == ACTIVE )
action=
{
    int rsupl_stat, fch;
    int msg_vec[2]; /*Pos 0 stores the ID and the pos 1 stores the msg_data*/
    float rsupl_rate;

    rsupl_stat = R_Suplemental_Channel_Status;/*Local var getting state var's value*/
    rsupl_rate = R_Suplemental_Channel_Rate;  /*Local var getting state var's value*/

    msg_vec[ID] = Id;
    msg_vec[MSG_DATA] = rsupl_rate;
    msg(PORT,BASE_STATION,msg_vec);
    rsupl_stat = INACTIVE;
    rsupl_rate = 0;
    /*Send message to base to report a Fundamental channel release*/

    set_st( "R_Suplemental_Channel_Rate"  , rsupl_rate);
    set_st( "R_Suplemental_Channel_Status", rsupl_stat);
};
```

Figura B.15: Objeto Web_Data_Client, atributo Events (parte 4)

Usuário_Web

```
event=F_Suplemental_Channel_Release(DET,1.0/BURST_TIMER)
 condition= (  F_Suplemental_Channel_Status == ACTIVE )
 action=
 {
     int fsupl_stat, fch;
     int msg_vec[2]; /*Pos 0 stores the ID and the pos 1 stores the msg_data*/
     float fsupl_rate;
     fsupl_stat = F_Suplemental_Channel_Status;/*Local var getting state var's value*/
     fsupl_rate = F_Suplemental_Channel_Rate;  /*Local var getting state var's value*/
     msg_vec[ID]       = Id;
     msg_vec[MSG_DATA] = fsupl_rate;
     /*Send message to base to report a Fundamental channel release*/
     msg(PORT,BASE_STATION,msg_vec);
     fsupl_stat = INACTIVE;
     fsupl_rate = 0;
     set_st( "F_Suplemental_Channel_Rate"  , fsupl_rate);
     set_st( "F_Suplemental_Channel_Status", fsupl_stat);
 };
/*---------------------------------------------------------------------------------------------*/
event=Fundamental_Channel_Request_Retransmition(DET,1.0/2.0)
 condition= ( ( ( Forward_Queue > 0 )||( Reverse_Queue > 0) )&&
              ( Fundamental_Channel_Status == WAITING_CHANNEL_REPLY ) )
 action=
 {
     int fch;
     int msg_vec[2]; /*Pos 0 stores the ID and the pos 1 stores the msg_data*/
     fch = INACTIVE;
     msg_vec[ID] = Id;
     msg_vec[MSG_DATA] = FUNDAMENTAL_CHANNEL_REQUEST;
     msg(PORT,BASE_STATION,msg_vec);
     /*Send message to base to report a Fund channel release*/
     set_st( "Fundamental_Channel_Status",fch);
 };
/*---------------------------------------------------------------------------------------------*/
event=Reverse_Suplemental_Channel_Request_Retransmition(DET,1.0/2.0)
 condition= ( ( Reverse_Queue > QUEUE_THRESHOLD )&&( Fundamental_Channel_Status == ACTIVE )
            &&( R_Suplemental_Channel_Status == WAITING_CHANNEL_REPLY ) )
 action=
 {
     int status, msg_vec[2]; /*Pos 0 stores the ID and the pos 1 stores the msg_data*/
     msg_vec[ID]       = Id;
     msg_vec[MSG_DATA] = R_SUPLEMENTAL_CHANNEL_REQUEST;
     msg(PORT,BASE_STATION,msg_vec);
     status = WAITING_CHANNEL_REPLY;
     set_st( "R_Suplemental_Channel_Status",status);
 };
```

Figura B.16: Objeto Web_Data_Client, atributo Events (parte 5)