UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE TELEMÁTICA

CONTRIBUIÇÕES EM NOVAS ESTRUTURAS DE COMUTADORES PARA REDES DE ALTA VELOCIDADE.

Autora

Magda Patrícia Caldeira Arantes

Mestre em Engenharia Elétrica pela UNICAMP em 1988
Engenheira Eletricista pela PUC-MG em 1983

Orientador

Shusaburo Motoyama

Doutor em Engenharia Elétrica pela Univ. Tókio em 1982

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, Departamento de Telemática, como parte dos requisitos necessário à obtenção do título de:

DOUTORA EM ENGENHARIA ELÉTRICA

Área de Concentração: AG - Telecomunicações e Telemática

Banca Examinadora

Prof. Dr. Shusaburo Motoyama	**(FEEC / UNICAMP)**
Prof. Dr. Paulo Roberto Guardieiro	**(UFU / Uberlândia)**
Prof. Dr. Juan Manuel Adán Coello	**(PUC / Campinas)**
Prof. Dr. Max Henrique Machado Costa	**(FEEC / UNICAMP)**
Prof. Dr. Paulo Cardieri	**(FEEC / UNICAMP)**
Prof. Dr. Akebo Yamakani	**(FEEC / UNICAMP)**

Campinas - SP - Brasil
23/06/2003

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

Ar14c

Arantes, Magda Patrícia Caldeira
Contribuições em novas estruturas de comutadores para redes de alta velocidade / Magda Patrícia Caldeira Arantes. --Campinas, SP: [s.n.], 2003.

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

1. Modo de transferência assíncrono. 2. Computação de pacotes (Transmissão de dados). 3. Sistemas de comunicação em banda larga. 4. Internet (Redes de computação). I. Motoyama, Shusaburo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RESUMO

O aumento explosivo do tráfego na Internet requer o uso de comutadores ATM ou de roteadores de alta capacidade. A função básica desses dispositivos é transportar pacotes das portas de entrada para as portas de saída. Como diversos pacotes que chegam por diferentes portas de entrada podem ser destinados a uma mesma saída, é necessário o armazenamento para resolver o conflito. A localização do componente de armazenamento (*buffer*) na estrutura de comutador afeta o desempenho do comutador e a complexidade do hardware ou dos algoritmos usados para a resolução de conflito.

Neste trabalho são apresentadas novas estruturas de comutadores para serem usados em redes ATM e na Internet. As estruturas propostas usam *buffers* grandes nas entradas e *buffers* pequenos nos pontos de cruzamento. São propostos algoritmos de escalonamento dos pacotes para resolver o problema de contenção na saída. Usando modelos matemáticos e modelos de simulação de eventos discretos, as estruturas propostas são analisadas e os seus desempenhos comparados. As estruturas propostas se mostraram de alta eficiência e apropriadas para serem usadas em redes de comutação de alta velocidade.

ABSTRACT

The explosive growth of Internet traffic demands high switching capacity routers or ATM switches. The basic function of these devices is to transmit packets from input ports to output ports. Since many incoming packets at switch input ports can be routed to a given output port at the same time, buffering devices are necessary to solve the conflict. The location of the buffers in the switch structure affects the switch performance, the complexity of hardware and algorithms used for conflict resolution.

In this work, novel switch structures to be used in ATM networks and in the Internet are presented. The proposed structures use large buffers at the input port and small buffers at the crossing points. Also, some packet scheduling algorithms are proposed to solve the output contention problem. Using mathematical models and discrete events simulation models, the proposed structures are analyzed and their performances are compared. The proposed structures have shown to be highly efficient and well adapted to be used in high-speed networks.

PALAVRAS CHAVES:

Comutador ATM, Estrutura Crossbar, Qualidade de Serviço, Escalonamento distribuído de células, Enfileiramento de saída virtual, Enfileiramento Parcial Ponderado.

Aos meus pais:

José de Paula Arantes

Senhorinha Caldeira Arantes

Aos meus amores:

Borelli, Rafael e Pedro

"O pessimista queixa-se do vento, o otimista espera que ele mude e o realista ajusta as velas."

(Willian George Ward)

AGRADECIMENTOS

Agradeço o Prof. Shusaburo Motoyama por sua paciência, dedicação, apoio e confiança que possibilitaram o desenvolvimento desse trabalho.

Agradeço meus pais José e Senhorinha, que me ensinaram a ter fé, coragem e perseverança para vencer os desafios e alcançar os meus objetivos.

Agradeço meus irmãos José de Paula, Sérvulo, José Eduardo, Márcia e José Ricardo, por sua amizade e apoio incondicional de toda a vida.

Agradeço meu marido Borelli e meus filhos Rafael e Pedro por sua compreensão e seu carinho nos momentos mais árduos dessa jornada.

Agradeço meus colegas de docência, dos quais sempre recebi grande colaboração e incentivo, em particular a Profª Izilda G. G. Capovilla, o Prof. Adriano D. Pila e o Prof. Dilermando Piva Jr.

Agradeço a banca examinadora, pela atenção e cuidado dispensados à avaliação deste trabalho e pelas sugestões dadas.

Agradeço o Departamento de Telemática pela infra-estrutura necessária ao desenvolvimento desse trabalho.

Agradeço o CNPq (Conselho Nacional de Pesquisa) pela bolsa recebida e a FAV (Faculdades de Valinhos) pelo apoio financeiro que recebi para participação em congressos.

Sumário

Índice de Figuras

Capítulo 1

1 Introdução

Os sistemas de comunicação são caracterizados pela especialização dos serviços oferecidos, assim, o sistema telefônico é especializado no tráfego de voz, as redes de difusão ou TV a cabo nos sinais de vídeo ou televisão, as redes de comutação de pacotes no tráfego de dados. Porém, muito esforço tem sido feito para o desenvolvimento de tecnologias e sistemas que possibilitem a implantação de uma rede única, capaz de suportar serviços com requisitos diferenciados como texto, voz e teleconferência oferecendo-lhes as qualidades específicas e necessárias a cada tipo de serviço.

Uma proposta de implementação para uma Rede Digital de Serviços Integrados (RDSI) que seja flexível o suficiente para proporcionar uma plataforma comum aos diversos serviços existentes e que seja expansível aos novos serviços que venham a ser propostos vem sendo estudada pelas organizações internacionais de padronização dos serviços públicos de telecomunicações. Entende-se por serviços públicos, aqueles oferecidos pelas concessionárias de telecomunicações ao público em geral. No setor privado, enquanto alguns fóruns internacionais de discussão apoiam e auxiliam as organizações privadas especificando e propondo alternativas para os sistemas de serviços integrados proprietários, outros discutem novas tecnologias que possibilitem a integração de serviços nas redes de pacotes atuais.

O desenvolvimento da tecnologia digital aliada à tecnologia de transmissão em fibras óticas foi fator determinante para o desenvolvimento das redes de serviços integrados. A tecnologia digital é mais eficiente, econômica e confiável que a tecnologia analógica, e a tecnologia de transmissão em fibras óticas é capaz de garantir altas taxas de transmissão com taxas de erro extremamente baixas.

Ao final da década de 80, teve início uma proposta para implementação de uma RDSI baseada na tecnologia de comutação rápida de pacotes, que levou à adoção do modo de transmissão assíncrono – ATM *Asynchronous Transfer Mode* – para as Redes Digitais de Serviços Integrados de Faixa Larga (RDSI-FL) [1]. Feita pelo ITU-T, tal proposta surgiu como a forma mais promissora para a implementação da RDSI-FL. A comunidade científica se engajou no estudo e na especificação dessa tecnologia e foram

criados fóruns de discussão, como o ATM Fórum, que visavam auxiliar e impulsionar os trabalhos dos grupos de padronização.

A rede ATM, como ficou conhecida, tem como objetivo oferecer uma rede única e universal para a transmissão dos mais variados serviços, com os mais variados requisitos de taxas de transmissão, proporcionando qualidade e eficiência na utilização de recursos.

A tecnologia ATM é baseada em comutação rápida de pacotes, que para garantir a eficiência e a rapidez na transmissão de dados na rede diminui ao máximo o processamento nos nós de comutação. As principais simplificações no processamento efetuado pelos nós são a eliminação da confirmação e da retransmissão de quadros na camada de enlace, a eliminação do controle de erros nos nós internos da rede e a adoção de encaminhamento de pacotes baseado em conexões virtuais. O controle de erro é feito apenas fim a fim, simplificando assim as funções do cabeçalho dos pacotes. Cada pacote, denominado célula, tem comprimento fixo e igual a 53 *bytes*, cinco *bytes* de cabeçalho e 48 *bytes* de dados. A principal função do cabeçalho da célula é a identificação da conexão virtual por um identificador que é selecionado durante a fase de sinalização e garante o encaminhamento apropriado da célula na rede.

O dispositivo básico de comutação é denominado comutador. Na Fig. 1.1 é mostrado o modelo geral de um comutador e ilustrado o princípio básico da comutação [2]. O comutador possui N linhas de entrada que recebem as células ATM e baseado nas informações contidas nos cabeçalhos das células, que apontam o circuito virtual ao qual elas pertencem, selecionam uma das portas de saída e transferem a célula para ser transmitida por uma das N linhas de saída. A estrutura de comutação é responsável pela resolução do encaminhamento da célula e a tabela de tradução é utilizada para efetuar a conversão dos endereços dos cabeçalhos de entrada para os novos endereços dos cabeçalhos das células de saída.

O processo de comutação envolve o armazenamento de células em elementos denominados *buffers*. O armazenamento é necessário para a resolução do conflito criado pelo recebimento simultâneo de células em diversas entradas destinadas à mesma saída. A localização do elemento de armazenamento de células, *buffers*, na estrutura de comutação afeta o desempenho do dispositivo comutador. Tem-se descrito na literatura, basicamente, quatro possibilidades: comutador com *buffer* na entrada, comutador com *buffer* na saída, comutador com memória compartilhada e comutador com *buffer* nos pontos de cruzamento. Cada uma dessas estruturas oferece vantagens e desvantagens, em termos de complexidade de implementação em *hardware* e complexidade do algoritmo de encaminhamento de células utilizado. O dispositivo comutador é um elemento chave em uma rede de transmissão de dados; a qualidade e a

eficiência da rede de transporte de células dependem do desempenho e da eficiência desse dispositivo, justificando a busca por dispositivos eficientes e economicamente realizáveis.

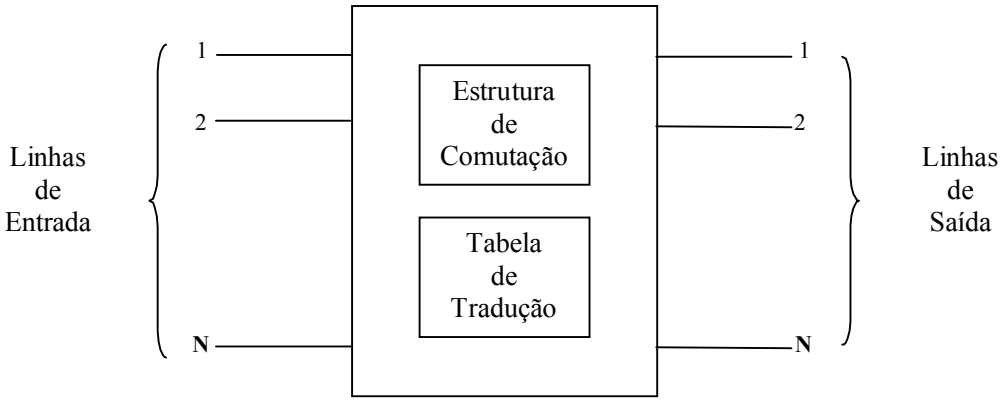


Figura 1-1: *Modelo Geral de um Comutador*

A solução ATM que é capaz de oferecer **Qualidade de *Serviço*** (*QoS Quality of Service*), foi projetada para utilização pelas concessionárias de serviços públicos de telefonia e prevê a atualização gradual da infra-estrutura atual instalada. Porém, se comparada com o funcionamento da Internet, que é a rede mundial de transmissão de dados, a tecnologia ATM introduz uma necessidade de gerência complexa e muda radicalmente sua filosofia de funcionamento.

A explosão do tráfego na Internet transformou os roteadores IP em grandes gargalos da rede. Toda a comunidade de pesquisa e a indústria de telecomunicações despenderam muito esforço para aumentar o desempenho dos roteadores e desenvolver novas tecnologias que pudessem prover QoS em redes baseadas em IP e atender à demanda pelos novos serviços. Assim, muitos mecanismos foram propostos. Os roteadores *IP Switching* da Ipsilon, o *Tag Switching* da Cisco, o ARIS (*Aggregate Route-Based IP Switching*) da IBM e o CSR (*Cell Switching Router*) da Toshiba são exemplos dos primeiros roteadores comerciais que implementaram esses mecanismos. Esses roteadores têm em comum o mecanismo multicamadas de troca de rótulos que é realizado pelo uso de uma semântica apropriada para ligar os rótulos a fluxos de pacotes, o uso de um protocolo para distribuir as informações de ligação entre os roteadores e o envio de pacotes de interfaces de entrada para interfaces de saída, baseado apenas na informação dos rótulos e não no endereço IP.

O envio de pacotes pode ser implementado por hardware, através de uma estrutura de comutação de um roteador ou pode ser feita por software, através da indexação do rótulo de um pacote de entrada a uma interface da Base de Informação de Rótulos (*TIB Tag Information Base*) para determinar a interface de saída correspondente.

O resultado é um roteador com velocidade de um comutador da camada de enlace (camada 2) e com a flexibilidade de um roteador de camada de rede (camada 3).

Um conceito importante, tanto para *IP Switching* como para *Tag Switching* é fluxo. Um fluxo é uma seqüência de pacotes de uma fonte particular para um destino particular, que compartilham algumas características, como o endereço da fonte e do destino, o número de portas, protocolo de transporte, o valor TTL (*Time to Live*) e o tipo de serviço [3]. Em um roteador IP convencional, todos os pacotes de um fluxo devem ser processados pela camada IP. Isso envolve centenas de linhas de processamento de software e introduz uma sobrecarga considerável. Os comutadores *IP Switching* e *Tag Switching* se propõem a melhorar o desempenho reduzindo o número de pacotes processados pela camada 3 (camada IP). Em um *IP Switching*, fluxos de longa duração são identificados e associados a um rótulo. Esses fluxos são então comutados na camada de enlace (camada 2). No *Tag Switching*, é possível pré distribuir rótulos e então todos os fluxos podem ser comutados. Estes dois mecanismos de comutação diferem em suas formas de ligar os rótulos (*tags*) às seqüências de pacotes, de distribuir a informação de ligação e de transferir pacotes através dos roteadores.

O roteador *IP Switching* é composto de um Comutador ATM, que comuta pacotes das portas de entrada para as portas de saída, e um Controlador, que usa o protocolo GSMP (*General Switch Management Protocol*) [4] para controlar o comutador ATM, estabelecendo o mapeamento de um fluxo de entrada, para uma porta de saída. O Controlador do *IP Switching* também executa a identificação de fluxo e usa o protocolo IFMP (*Ipsilon Flow Management Protocol*) [5] para se comunicar com os roteadores adjacentes. Dessa forma, os fluxos podem ser corretamente rotulados e comutados em uma estrutura de comutação ATM. Na Fig. 1.2 é mostrada a estrutura genérica de um roteador *IP Switching*.

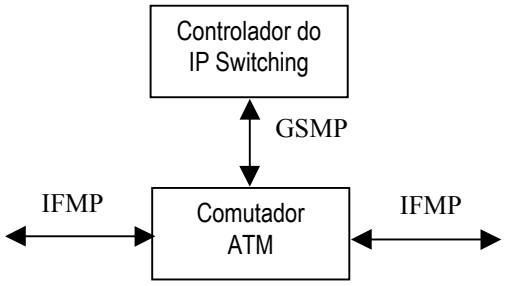


Figura 1-2: *Estrutura genérica de um roteador IP Swiching*

No mecanismo *tag switching* [6], as ligações dos rótulos aos LSPs (*Label Switching Paths*) são criadas quando a informação de controle chega. Os rótulos são atribuídos em resposta ao processamento normal do tráfego do protocolo de roteamento, como por

exemplo pelo tráfego de controle do RSVP (*Resource Reservation Protocol*) ou através de uma configuração estática.

As primeiras soluções de comutador multicamada mantinham uma componente de controle IP e usavam a troca de rótulos do ATM como componente de encaminhamento. Entretanto todas as soluções eram proprietárias e portanto não interoperavam. Isto levou ao desenvolvimento do MPLS (*Multi Protocol Label Switch*), que estabeleceu um padrão. O IETF (*Internet Engineering Task Force*) organizou o grupo MPLS [7], para homogeneizar a plataforma das redes quanto aos quesitos vazão, qualidade de serviço e escalabilidade, diminuindo o conjunto de soluções tecnológicas distintas e aumentando a viabilidade de interfuncionalidade. Esse grupo expediu um plano de trabalho em dezembro de 1997.

Todas as soluções do modelo multicamadas, incluindo o MPLS são compostas por duas componentes distintas: o controle e o encaminhamento. A componente de controle utiliza os protocolos de roteamento usuais (OSPF, IS-IS e BGP4) na troca de informações entre roteadores, para construir as tabelas de roteamento. Quando o pacote chega, a componente de encaminhamento procura a informação na tabela para tomar a decisão de roteamento para cada pacote. A componente de encaminhamento é baseada em troca de rótulos e utiliza o mesmo algoritmo das redes ATM. A sinalização e a distribuição de rótulos são fundamentais para a operação de encaminhamento do algoritmo.

O MPLS integra a arquitetura de Serviços Integrados para a Internet (*IntServ - Integrated Service*). A arquitetura *IntServ* [8], proposta pelo IETF, estabeleceu o início do desenvolvimento da infra-estrutura necessária aos serviços multimídia na Internet, no topo da pilha de protocolos IP.

As arquiteturas de Serviços Integrados têm sido definidas usando os protocolos que estão sendo implementados para os roteadores IP, como o RSVP [9]. O conceito básico de um modelo *IntServ* é o acréscimo de tarefas tradicionalmente executadas em redes baseadas em comutação de circuitos a um roteador IP existente. Desse modo a Internet passa a ter caráter orientado-a-conexão. Portanto, operações como policiamento, moldagem, controle de admissão e gerenciamento de QoS devem ser providas por todos os roteadores de RSVP para cada fluxo IP, conforme mostrado na Fig. 1.3(a). Entretanto, em uma rede de larga escala com milhões de usuários conectados, o número de sessões IP tratadas pelos roteadores, no núcleo da rede, pode ser muito grande. Consequentemente, a execução das funções para todos os fluxos IP ativos no núcleo do roteador resulta em fraco desempenho e dificulta a capacidade de expansão da arquitetura de rede.

Um outro modelo, conhecido como modelo de Serviços Diferenciados (*DiffServ - Differentiated Services*) [8], [10], [11], [12] define um conjunto de classes de tráfego, cada qual projetada para servir aplicações com demandas de QoS similares. Uma classe de tráfego descreve o comportamento por *hop* (PHB *Per Hop Behavior*) que os pacotes de sua classe devem experimentar em cada nó da rede. O comportamento em cada nó determina a prioridade, o atraso máximo na fila de transmissão, a banda de passagem compartilhada do enlace e a probabilidade do pacote ser perdido. O modelo *DiffServ* assegura alta modularidade, separando as operações executadas nas bordas da rede daquelas realizadas no núcleo da rede. Os roteadores de borda executam tarefas complexas como o policiamento e a moldagem, marcando e priorizando rotas, conforme mostrado na Fig. 1.3(b). O processo usado para classificar os pacotes IP, pertencentes a um fluxo específico IP, associando-os à sua classe de fluxo de tráfego apropriada é denominado *marcação*. Todas essas operações são executadas para cada fluxo, como no modelo *IntServ*. Entretanto, o pequeno número de fluxos IP ativos processados pelo roteador da borda não provoca os problemas de expansividade que existe na arquitetura *IntServ*. Por outro lado, os roteadores do núcleo da rede executarão apenas uma única operação, que será o roteamento priorizado.

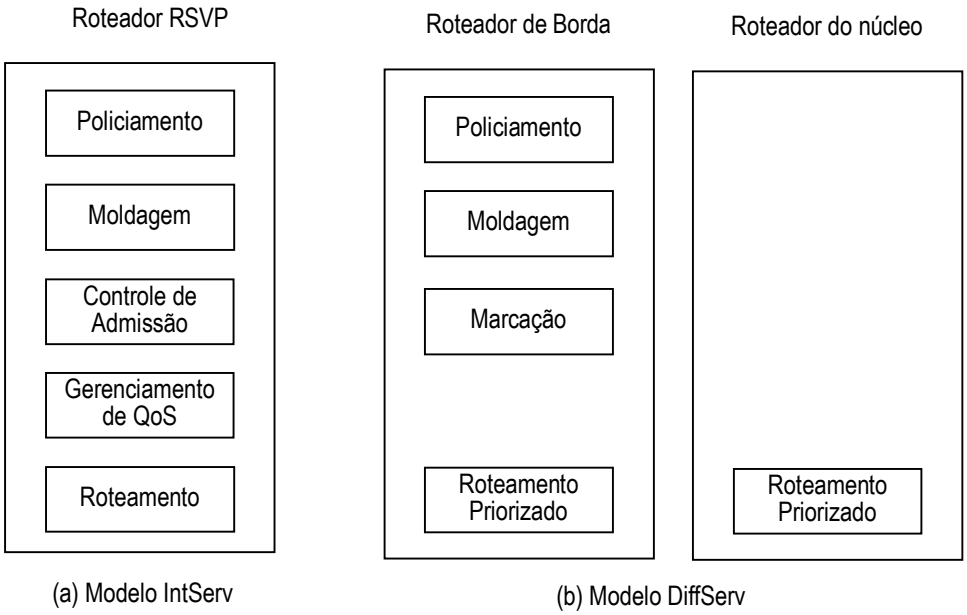


Figura 1-3*: Tarefas executadas nos elementos de uma (a) Rede IntServ e (b) Rede DiffServ*

Os roteadores do núcleo da arquitetura *DiffServ* não guardam qualquer informação do fluxo estabelecido. Ao contrário, eles simplesmente servem os pacotes de acordo com a classe de tráfego que o roteador de borda de Ingresso tenha designado ao pacote. Consequentemente, cada roteador do núcleo da rede *DiffServ* deve conhecer

apenas o número de classes de tráfego e o correspondente comportamento por nó de cada classe.

Outra solução proposta pelo IETF [13] que também busca o provimento de QoS na Internet é o novo padrão do protocolo IP. A versão corrente do IP que é o IPv4 possui espaço de endereçamento limitado. Quando o protocolo IP foi definido, existiam poucas redes de computadores e os projetistas usaram 32 bits para definir um endereço IP, o que permitiu que a Internet incluísse mais de um milhão de redes. Entretanto, a Internet global tem tido crescimento exponencial, e com o crescimento atual que praticamente dobra o número de comutadores conectados a cada ano, o número máximo de redes será logo atingido. Assim, o motivo principal para a definição de nova versão para o padrão IP é a necessidade de aumentar o espaço de endereçamento do protocolo e acomodar um número maior de novas redes. A outra motivação para a nova versão do IP é a necessidade de provisão de QoS para as novas aplicações da Internet. Assim, a especificação do IPv6 inclui o conceito de fluxo, o qual define como uma seqüência de pacotes de uma determinada origem para um determinado destino (unicast ou multicast) e que requer um tratamento especial pelos roteadores. Os campos "*priority*" e "*flow label*" foram criados no cabeçalho do pacote IPv6, especialmente para facilitar o desenvolvimento de protocolos para controle de tráfego em tempo real, como o RSVP, de forma a permitir a implementação de uma Internet com aplicações multimídia e com a integração de serviços de dados, voz e vídeo em tempo real.

Assim sendo, o que se vislumbra atualmente não é mais uma rede única e integrada de serviços usando exclusivamente a tecnologia ATM, e sim um ambiente de rede heterogêneo, formado por nuvens com diferentes tecnologias, adequadas às principais mídias transmitidas, mas interligadas, formando uma rede convergente, utilizando a técnica de comutação de pacotes. Torna-se agora imprescindível, o estudo de estruturas de comutação capazes de comutar pacotes, porém adaptadas às novas demandas de serviço, com diferentes requisitos de qualidade.

O objetivo deste trabalho é apresentar propostas de estruturas de comutação de alto desempenho e algoritmos de escalonamento de pacotes, que sejam flexíveis e eficientes, para serem usadas em redes de transmissão de alta velocidade. Inicialmente são feitas propostas de comutadores para redes que usam pacotes de comprimento fixo, como é o caso das redes ATM. Muitos mecanismos usados para prover QoS na Internet operam usando diretamente comutadores ATM, entretanto, é possível prover maior flexibilidade e melhor desempenho se forem utilizadas estruturas de comutação capazes de comutar, diretamente, pacotes de comprimento variável. Assim, são propostas também estruturas de comutação capazes de comutar pacotes de

comprimento variável e que possuem facilidade de prover qualidade de serviço, ou de prover equidade no uso da largura de banda.

No capítulo 2 é apresentada uma visão geral das principais estruturas de comutação que são usadas em comutadores com comprimento de pacotes fixo, comutadores ATM, e de estruturas de comutadores com comprimento de pacote variável, como é o caso das redes IP. Algumas vantagens e desvantagens são apontadas para cada uma dessas estruturas.

No capítulo 3 são apresentadas as propostas para diferentes estruturas de comutação ATM (apropriadas para comutar pacotes de comprimento fixo) que utilizam buffers combinados nas entradas e nos pontos de cruzamentos [14]. Os algoritmos utilizados para transferências de células dos buffers de entrada para os buffers do pontos de cruzamento, e dos buffers dos pontos de cruzamento para as linhas de saída, juntamente com a análise da vazão de cada estrutura são apresentados. Uma comparação dessas estruturas é feita, considerando a complexidade de implementação em *hardware* e a dificuldade de gerenciamento de *buffers*. Foi desenvolvido um modelo analítico [15] e baseado nesse modelo, foi feita a análise de desempenho da estrutura apresentada, que se mostrou mais eficiente na priorização do tráfego com maior exigência de QoS.

No capítulo 4, a estrutura apresentada no Capítulo 3 que melhor priorizou o tráfego com maior exigência de QoS foi generalizada [16], para trabalhar com algoritmos de escalonamento de tráfego distribuído. Utilizando algoritmos de escalonamento baseado em prioridade, foi desenvolvido um modelo matemático para a análise dessa estrutura. Os resultados da análise de desempenho com base no modelo matemático e na validação desse modelo através de simulação de eventos discretos também são apresentados.

No capítulo 5 são apresentadas duas propostas de estruturas de comutadores para uso em redes de comutação de pacotes, como é o caso da Internet, que usam pacotes de tamanho variável. A primeira delas utiliza um algoritmo que dá prioridade aos pacotes pertencentes aos fluxos que exigem maior QoS [17]. Porém, quando se utiliza algoritmos baseados em prioridade em redes de pacotes de comprimento variável, o uso da largura de banda disponível pode ser ocupado indevidamente, apenas por fontes que transmitem pacotes longos de alta prioridade. Para resolver esse problema, a segunda proposta utiliza algoritmos de escalonamento que previnem o desbalanceamento entre os fluxos no uso da banda de passagem. Para a primeira proposta é feita a análise de desempenho baseado em um modelo matemático desenvolvido e os resultados obtidos são comparados com resultados de simulação,

visando a validação desse modelo. Para a segunda proposta a análise de desempenho é feita baseada em modelo de simulação desenvolvido.

Finalmente, no capítulo 6 são apresentadas as principais conclusões e contribuições desse trabalho e são feitos alguns comentários sobre possíveis trabalhos futuros.

Capítulo 2

2 Comutadores de Redes de Alta Velocidade.

2.1 Introdução

A função básica de um sistema de comutação, seja em uma rede de comutação de células ou em uma rede de comutação de pacotes, é transferir as células (ou pacotes) de uma porta de entrada para uma porta de saída. O armazenamento de células é necessário, pois é possível que várias células, que chegam por diferentes portas de entrada, possam ser endereçadas a uma mesma saída. A localização do dispositivo de armazenamento, e portanto da inevitável fila de espera, afeta o desempenho do comutador e a complexidade do hardware ou dos algoritmos usados para a resolução de conflito.

Neste capítulo é feito um resumo dos principais conceitos de ATM e são descritas as principais estruturas de comutadores em redes de alta velocidade encontradas na literatura. As principais vantagens e desvantagens de cada estrutura serão discutidas.

2.2 Redes ATM

O Modo de Transferência Assíncrono (ATM) é uma tecnologia baseada na transmissão de pequenas unidades de informação de tamanho fixo e formato padronizado que são transmitidas através de conexões com circuitos virtuais. Essa tecnologia permite suportar diferentes serviços, pois é capaz de satisfazer os requisitos exigidos pelos diferentes tipos de tráfego à altas velocidades de transmissão. Por essa razão, o ATM foi escolhido para suportar a diversidade de serviços definida para a Rede Digital de Serviços Integrados de Faixa Larga (RDSI-FL).

2.2.1 Modelo de Referência do ATM

O *Modelo de Referência para Protocolos* [1](*Protocol Reference Model* – PRM) consiste em um **plano de usuário**, **um plano de controle** e um **plano de gerenciamento**. O plano de usuário é utilizado para a transferência de informação dos usuários. O plano de controle é responsável pelas funções de controle, como a sinalização necessária para

ativar, manter e desativar chamadas e conexões. O plano de gerenciamento é responsável pelo **gerenciamento dos planos** e pelo **gerenciamento das camadas**.

O **gerenciamento dos planos** é utilizado para o gerenciamento dos planos de usuário, de controle e do próprio plano de gerenciamento.

O **gerenciamento das camadas** apresenta uma estruturação em camadas. Cada camada trata do gerenciamento dos fluxos de informações de operação e manutenção, incluindo o gerenciamento de recursos e parâmetros associados às entidades de protocolo.

O **plano de usuário** é dividido em três camadas inferiores e camadas superiores. As camadas superiores não são objeto de definição das recomendações da RDSI-FL baseadas no PRM. As camadas inferiores são a Camada Física, a Camada ATM e a Camada de Adaptação (*ATM Adaptation Layer – AAL*). As camadas Física e de Adaptação ainda apresentam uma subdivisão interna. As camadas Física e ATM são comuns ao plano de controle e ao plano de usuário.

Na Tabela 2.1 são apresentadas as principais funções das camadas do modelo de referência do ATM.

Tabela 2-1: ESTRUTURAÇÃO DAS CAMADAS DO PRM-ATM

CAMADAS	SUB-CAMADAS	FUNÇÕES
ADAPTAÇÃO	CS	– Convergência
	SAR	– Segmentação e Remontagem
ATM		– Controle Genérico de Fluxo
		– Geração e extração do cabeçalho
		– Interpretação de VPI/VCI
		– Multiplexação/Demultiplexação de células
FÍSICA	TC	– Desacoplamento de taxa de células
		– Geração e verificação de HEC
		– Delineamento de células
		– Geração e recuperação de *frames*
	PM	– Transmissão pelo meio físico
		– Conversão eletro-ótica

Camada Física

A Camada Física apresenta duas subcamadas: a subcamada de meio físico (*PM - Physical Medium*) e a subcamada de Convergência de Transmissão (*TC - Transmission Converge*).

A Subcamada PM é responsável pela transmissão adequada de bits pelo meio físico, incluindo o alinhamento de bits, a sinalização na linha e a conversão eletro-ótica; também é responsável pela codificação do fluxo de informação recebido da TC em uma forma adequada para a transmissão no meio físico, possibilitando a sincronização entre os circuitos transmissores e receptores.

A codificação está relacionada com uma das formas de delineamento de células. Símbolos obtidos através de violação de código podem ser utilizados como marcadores de início e fim de células.

Subcamada TC

A subcamada TC é responsável pelas seguintes funções:

1. ***Desacoplamento da taxa de transmissão em relação à taxa de geração de células.*** O desacoplamento é feito através da inserção de células especiais (ociosas) na transmissão, que deverão ser descartadas na recepção, de forma a ajustar o fluxo de células à taxa de transmissão utilizada pelo meio.
2. ***Controle de erro do cabeçalho.*** A célula ATM contém um cabeçalho com informações para seu encaminhamento e um campo com informações de redundância para detecção de erros no cabeçalho (*Header Error Check – HEC*). A subcamada TC é a responsável pelo cálculo e a inserção do HEC no cabeçalho da célula, bem como pela sua verificação no lado recebedor. Células com erros irrecuperáveis no cabeçalho são simplesmente descartadas.
3. ***Delineamento de células.*** É a função que permite determinar o início e o fim de uma célula, em um fluxo de bits ou bytes recebidos. Esta função prepara o fluxo de células para permitir que o lado receptor recupere as fronteiras das células.
4. ***Embaralhamento:*** A subcamada de convergência de transmissão é capaz de efetuar o embaralhamento da seqüência de bits da parte de informação da célula a ser transmitida, de forma a evitar longas seqüências de uns ou zeros.

Camada ATM

As funções da camada ATM incluem:

1. Multiplexação e Demultiplexação de células.

2. Adição e remoção do cabeçalho das células.
3. Comutação e encaminhamento de células baseado na informação do cabeçalho (realizado pelos nós de comutação).
4. Controle genérico de fluxo (*Generic Flow Control – GFC*).

O formato das células ATM é especificado e o encaminhamento das células na rede baseando-se na informação contida nos campos do cabeçalho.

O formato da célula ATM difere na UNI (UNI - *Unit Network Interface*) e na NNI (NNI - *Network Network Interface*), como mostra a Fig. 2-1.

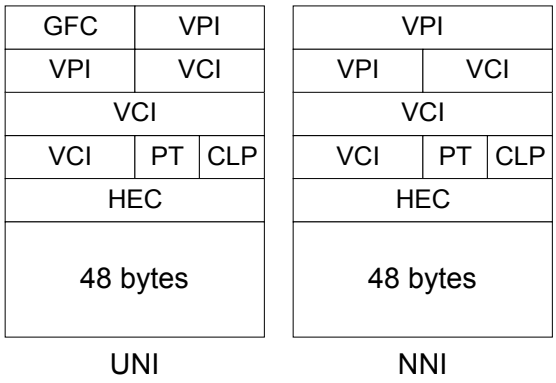


Figura 2-1: *Formato da célula ATM na UNI e na NNI*

A UNI define a interface entre um host e uma rede ATM. A NNI define a interface entre dois comutadores ATM.

A célula utiliza cinco octetos para o cabeçalho e 48 para as informações.

O campo GFC é definido para uso pelo mecanismo de controle de fluxo genérico na UNI.

O campo VPI (*Virtual Path Identifier*) em conjunto com o VCI (*Virtual Channel Identifier*) formam o rótulo da conexão, que é utilizado pelos comutadores para encaminhar as células ao destino.

O campo PT (*Payload Type*) indica o tipo de informação contida na célula (informação de usuário ou de manutenção) e indica também se a célula passou por dispositivos congestionados durante seu trajeto.

O campo CLP (*Cell Loss Priority*) indica a prioridade, caso haja necessidade de descarte de células (células com CLP = 1 são descartadas primeiro).

O campo HEC é utilizado para a detecção de erros do cabeçalho, para a correção de erro em um único bit e para o delineamento de células.

2.2.2 Comutação de Células ATM

Em uma rede ATM, as células são transportadas através de conexões. Uma conexão fim a fim em redes ATM é denominada Conexão de Canal Virtual (*VCC Virtual Channel Connection*). O conceito de conexão com canal virtual é semelhante ao conceito de conexão com circuito virtual. Uma VCC é formada pela concatenação de conexões virtuais estabelecidas nos vários enlaces da rede, da origem ao destino, formando um caminho único, no qual as células serão encaminhadas. Cada conexão virtual em um enlace é denominada Enlace de Canal Virtual (*VCL - Virtual Channel Link*) e é identificada através do Identificador do Canal Virtual (VCI) contido no cabeçalho da célula.

Para que as células sejam encaminhadas pela rede até o destino, é necessário que cada comutador saiba encaminhar corretamente as células de cada VCC estabelecida. Células que chegam a um nó de comutação através de uma VCL devem ser encaminhadas à próxima VCL do caminho estabelecido por uma VCC. Em cada comutador, a próxima VCL do caminho está relacionada a uma das suas portas. Em cada enlace físico da rede existirão diversas VCLs correspondendo a diferentes VCCs. Cada célula deve informar ao comutador por qual VCL ela foi enviada, através de informação contida no seu cabeçalho (VCI).

Quando uma célula chega a um comutador ATM, o rótulo identifica o VCL utilizado pelo comutador anterior do caminho estabelecido pela VCC. De posse dessa informação e da porta de entrada, o comutador consulta uma tabela que relaciona cada VCL e porta de entrada ao próximo VCL e porta de saída a ser utilizada no caminho estabelecido pela VCC. Então, o comutador atualiza o rótulo da célula e a retransmite pela porta de saída especificada na tabela.

Vários VCCs podem ser comutados em conjunto formando uma conexão de caminho virtual (*VPC - Virtual Path Connection*). Os VPCs são formados através da concatenação de Enlaces de Caminhos Virtuais (*VPL - Virtual Path Link*) identificados pelo Identificador de Caminho Virtual (VPI) contidos nos cabeçalhos das células.

Em cada nó de comutação o comutador: recebe a célula na porta de entrada, analisa os campos VPI+VCI do cabeçalho da célula, caso ele seja um comutador de VP e VC, ou o campo VPI caso ele seja apenas um comutador de VP, consulta sua tabela (dinâmica) de roteamento, atualiza o conteúdo destes campos na célula e envia a célula à porta de saída apropriada.

Camada de Adaptação

A Camada de Adaptação ao ATM (ATM Adaptation Layer – AAL) tem como função compatibilizar e oferecer os serviços desejados pelas camadas superiores, utilizando como base a tecnologia ATM e efetuando as adaptações necessárias. A AAL é a primeira camada de protocolo fim a fim no modelo de referência da RDSI-FL.

Para dar suporte a diferentes requisitos, o ITU-T dividiu as classes de tráfego existentes levando em consideração a sua natureza, a necessidade ou não de manter a relação temporal da informação no destino, e se o serviço é ou não orientado à conexão. Já o ATM Fórum [18] divide os serviços ATM em cinco categorias, que serão discutidas na seção seguinte.

2.2.3 Arquiteturas de Serviço ATM

A arquitetura de serviços ATM consiste em cinco categorias de serviço [18]:

- CBR: Constant Bit Rate
- rt-VBR: real-time Variable Bit Rate
- nrt-VBR: non-real time Variable Bit Rate
- UBR: Unspecified Bit Rate
- ABR: Available Bit Rate.

As categorias de serviço estão relacionadas com os requisitos de QoS exigidos da rede. As funções de roteamento, CAC (*Connection Admission Control*) e alocação de recursos da rede são estruturados diferentemente para cada categoria de serviço. Da mesma forma que no ITU-T, as categorias de serviço no ATM Fórum são separadas em serviços de tempo real e serviço sem exigência de tempo real. Os serviços CBR e rt-VBR são de tempo real. Esses dois serviços diferem pelos descritores de tráfego. O descritor do tráfego CBR contém apenas o parâmetro PCR (*Peak Cell Rate*) enquanto que o tráfego rt-VBR contém os parâmetros PCR e SCR (*Sustainable Cell Rate*).

Na definição de cada classe de serviço serão usados os seguintes parâmetros da QoS: *peak-to-peak Cell Delay Variation* (*peak-to-peakCDV*), *Maximum Cell Transfer Delay* (*maxCTD*) e *Cell Loss Ratio* (CLR).

Classe de serviço CBR

Usada por serviços que necessitam de largura de banda constante e disponível durante toda a conexão. A largura de banda é caracterizada pela taxa de pico das células

(PCR). As células que sofrerem atraso acima de maxCTD não terão significado para a aplicação. O serviço CBR foi especificado para suportar aplicações de tempo real com restrição de atraso crítica, como por exemplo, voz e vídeo, mas não se restringe apenas a essas aplicações. As aplicações CBR podem emitir células com taxa PCR em qualquer instante, durante qualquer duração de tempo e a QoS deve ser mantida no padrão negociado.

Classe de serviço rt-VBR

A classe de serviço rt-VBR é prevista para aplicações com restrições críticas de atraso e variação de atraso, como por exemplo as aplicações de vídeo e voz. As conexões rt-VBR são caracterizadas em termos das taxas PCR, SCR e MBS (*Maximum Burst Size*). As fontes de tráfego rt-VBR transmitem a taxas que variam no tempo e podem ser descritas como fontes de tráfego em "rajadas". As células que sofrerem atrasos maiores que maxCTD perdem o significado para a aplicação.

Classe de serviço nrt-VBR

A classe de serviço nrt-VBR é prevista para aplicações sem restrições de tempo real e que são caracterizadas por "rajadas" que podem ser descritas em termos das PCR, SCR e MBS. As células que obedecerem os parâmetros de tráfego contratados deverão sofrer baixa CLR. Nenhum limite de atraso é associado a essa categoria de tráfego.

Classe de serviço ABR

ABR é a classe de serviço ATM, para a qual as características fornecidas pela camada ATM podem ser alteradas durante a conexão. A classe ABR prevê um mecanismo de controle de fluxo que suporta vários tipos de sinais de realimentação para controlar a taxa de emissão de células da fonte de acordo com a variação das características da camada ATM. Os serviços ABR não exigem limite de atraso ou da variação do atraso e não são previstos para suportar tráfego de tempo real. Durante o estabelecimento de uma conexão ABR, os sistemas finais podem especificar a banda de passagem mínima e máxima necessária, o que poderia ser especificado respectivamente pelas taxas PCR e MCR (*Minimum Cell Rate*). A taxa MCR pode assumir o valor zero. A banda disponível para o serviço ABR pode variar, mas não pode se tornar menor que MCR.

Classe de serviço UBR

A classe de serviço UBR é prevista para aplicações sem requisitos de tempo real que não possuem restrições críticas de atraso ou de variação de atraso. Exemplo de tais

aplicações são as aplicações tradicionais de comunicação de dados, tais como transferência de arquivo e de e-mail. .

2.3 Comutadores ATM

A rede ATM é orientada a conexão. Todas as células pertencentes a uma conexão particular seguem um caminho preestabelecido. As células pertencentes a uma conexão particular são identificadas por um campo de identificação de canal virtual VCI, que tem apenas significado local. Os comutadores de pacotes convencionais são construídos por meio de processamento de software em computadores de propósito geral ou através de um conjunto de processadores de propósitos especiais e operam a taxas moderadas. Já os comutadores ATM processam pacotes a uma taxa extremamente alta (na faixa de100.000 a 1.000.000 de pacotes por segundo por linha) [19].

A estrutura geral de um comutador ATM é mostrada na Fig. 2-2. As células que entram no comutador são processadas em um controlador de entrada (CE), pela matriz de comutação e pelos controladores de saída (CS). O Processador de Controle é necessário para executar as funções de alto nível tais como o estabelecimento e término de conexão, a alocação de banda de passagem, manutenção e o gerenciamento da rede. O processador de controle comunica-se com os controladores de entrada e de saída através de enlaces diretos de comunicação ou por intermédio das células através da estrutura de comutação. A interface externa de um comutador é bidirecional e formada por agrupamentos de portas de entrada e portas de saída. As células são sincronizadas nos controladores de entrada, assim, as células entram em um comutador com seus cabeçalhos alinhados. O campo VCI das células de entrada é traduzido nos controladores de entrada através de busca em tabelas. Para cada conexão virtual de entrada, a tabela de conexão pode incluir a informação de etiquetas de roteamento, anexadas às células de entrada, de forma que todas as células que pertencem ao mesmo circuito virtual sejam roteadas para a mesma saída.

A tabela de conexão pode conter outras informações adicionais, como por exemplo, prioridades, classes de serviço e tipo de tráfego da conexão. As informações extras, adicionadas às células, auxiliam a estrutura de comutação nas decisões de roteamento e armazenamento. Os controladores de saída removem a informação adicional, enquanto transmitem as células pelos enlaces de saída.

As células que chegam a um comutador ATM não são escalonadas e, portanto, pode ocorrer mais de uma requisição de entrada para uma mesma porta de saída em uma janela de tempo (ou *time slot*), que é o tempo necessário para transmitir uma célula. Este evento é conhecido como contenção de saída e está presente em qualquer projeto

de estrutura de comutadores ATM. A localização dos buffers, a topologia da matriz de comutação e o mecanismo de resolução de contenção são os três aspectos importantes no projeto de um comutador ATM.

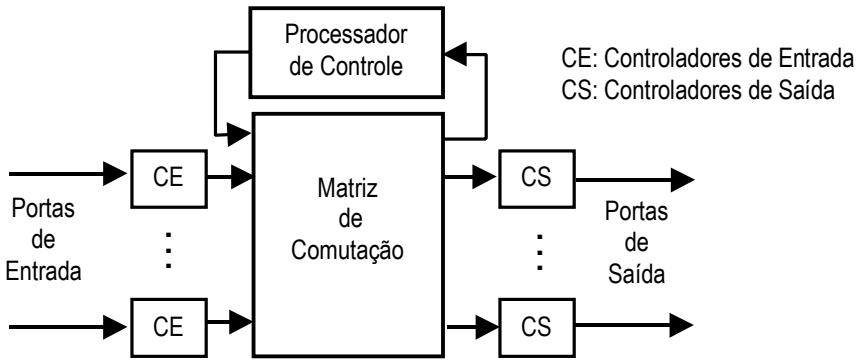


Figura 2-2: Estrutura Geral de um Comutador ATM.

Existem na literatura várias propostas de estruturas para comutadores ATM de alta velocidade [20]-[44]. Algumas propostas utilizam buffers na entrada [20] mas este tipo de estrutura apresenta o problema de contenção de saída, conhecido como HOLB (*Head Of Line Blocking*) e necessita um algoritmo de encaminhamento de células para melhorar a vazão do comutador [21]-[31] o que limita a sua velocidade. Outras propostas utilizam buffers na saída [35], [36], [37] conseguindo vazão de 100%. Entretanto, para se conseguir transferir todas as células que chegam nas entradas do comutador aos buffers de saída durante o intervalo de um "slot" (tempo de duração de uma célula) é necessário um procedimento sofisticado. Existem propostas de estruturas que utilizam buffers na entrada e na saída [39], [40], [41], estas estruturas necessitam técnicas eficientes para transferir as células dos buffers de entrada para a saída. Existem também propostas para se utilizar estruturas do tipo crossbar com buffers nos pontos de cruzamentos [42], simplificando as técnicas de transferência de células da entrada para a saída.

2.3.1 Comutadores ATM com *buffers* na entrada

A estrutura de um comutador ATM com buffer na entrada é mostrada na Fig. 2–3. As células que chegam são armazenadas nos buffers de cada porta de entrada, portanto, cada buffer pode possuir células endereçadas a diferentes portas de saída. O fenômeno conhecido como HOLB pode ocorrer, limitando a vazão do comutador.

A contenção da saída HOLB ocorre quando células chegam simultaneamente em diversas entradas e são endereçadas a uma mesma saída. Na Fig. 2–3, considere que a célula da entrada ***i*** tenha sido selecionada para ser transmitida pela saída ***x***. Porém, a

entrada ***j*** também possui uma célula destinada à mesma saída ***x***. Nesse caso, a célula da entrada ***j*** destinada à saída ***x*** ficará impedida de ser transmitida, bem como a célula endereçada à saída ***y*** que espera no buffer de entrada ***j***, mesmo que a saída ***y*** esteja desocupada. A célula da entrada ***j*** destinada à saída ***x***, nesse caso, está bloqueando a transferência das células que esperam no buffer dessa entrada.

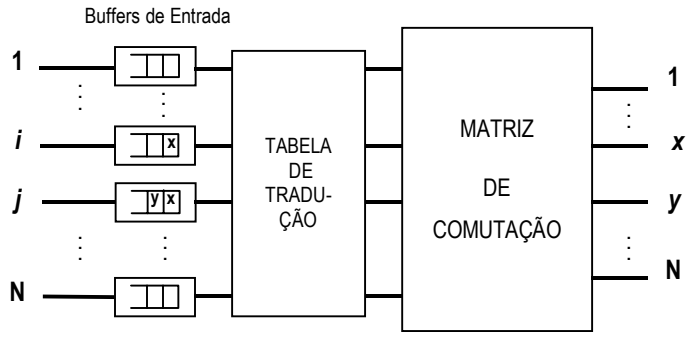


Figura 2-3: *Estrutura de Comutador com buffer na entrada*

O desempenho dessa estrutura é analisado em [20], considerando que as chegadas das células em cada entrada são independentes e identicamente distribuídas, que os processos de chegada de células em cada entrada são independentes das chegadas nas outras entradas, que todos os processos de chegada têm a mesma taxa de chegada e o destino das células é uniformemente distribuído para todas as saídas e que os buffers nas entradas são do tipo FIFO (*First-In-First-Out*). Essa análise demonstrou que a vazão teórica máxima da estrutura é de 58,6%.

A vazão da estrutura de comutação com buffers na entrada pode ser melhorada alterando a disciplina de atendimento dos buffers de entrada para FIRO (*First-In-Randon-Out),* ou adotando esquemas de pré encaminhamento de células, ou dividindo os *buffers* em cada entrada da estrutura. Em [22] e [25] é analisada a estrutura de comutação com *buffers* na entrada, usando a estratégia de distribuição dos *buffers* nas entradas. Em cada porta de entrada são colocados N *buffers* do tipo FIFO, um para cada porta de saída, conforme mostrado na Fig.2–4(a). O problema do bloqueio é completamente resolvido, uma vez que as células são previamente selecionadas e armazenadas nos *buffers* correspondentes a cada porta de destino. A análise feita em [22] mostra que a vazão máxima para um comutador que usa essa estrutura pode chegar a 100%. A desvantagem dessa configuração é que o número total de *buffers* tende a crescer na proporção N^2.

Outra possibilidade, e que também melhora a vazão da estrutura de comutação, é a adoção de uma solução intermediária entre a estrutura de comutação de *buffer* único

em cada entrada e a estrutura com N buffers em cada entrada. Em [26] é mostrado que a divisão da porta de entrada em X buffers (X≤N) em um comutador NxN pode reduzir o bloqueio na entrada do comutador. Para X=2 é possível obter a taxa média de utilização da porta de saída de 76% e para X=4, essa taxa é de 88%. A Fig. 2–4(b) ilustra essa estrutura com buffers de entrada divididos, para X=2.

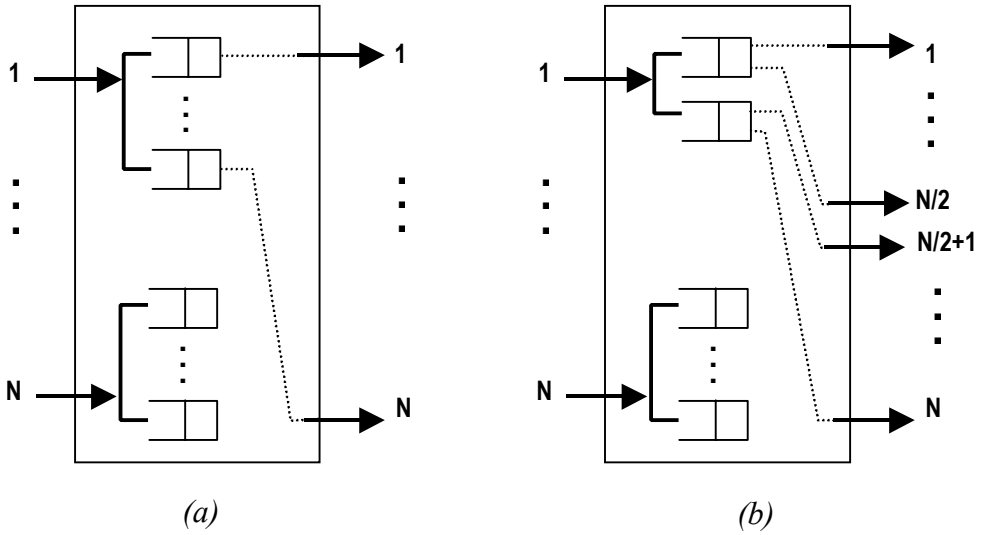


Figura 2-4: *Estrutura de Comutador com buffers de entrada divididos.*

Para melhorar a vazão em comutadores com buffers na entrada, vários algoritmos foram propostos. Alguns desses algoritmos serão discutidos.

Algoritmo PIM (Parallel Iterative Matching)

Este algoritmo, apresentado em [27], pode usar mais de uma iteração em cada janela de tempo e cada iteração é executada em três fases, que são:

1. Fase de Requisição (*Request Phase*): Cada entrada não combinada envia uma requisição à cada saída, para a qual ela tenha uma célula armazenada.
2. Fase de Concessão (*Grant Fase*): Caso uma saída não combinada tenha recebido mais de uma requisição, então ela escolhe aleatoriamente aquela que será atendida, e envia um sinal de concessão para entrada escolhida.
3. Fase de Aceitação (*Accept Phase*): Se uma entrada não combinada receber mais de uma concessão, então ela escolhe uma concessão aleatoriamente e envia um sinal de aceitação para a saída escolhida.

Questões sobre o número de iterações possíveis em uma janela de tempo também são abordadas em [27], e a conclusão da análise feita é que, com quatro iterações a vazão chega a 99,9% com 100% de carga, e que para um número de iterações maior que quatro, não há ganho significativo de desempenho.

Algoritmo SPIM (Simplified Parallel Iterative Matching)

O algoritmo SPIM [21], uma simplificação do algoritmo PIM, também pode usar mais de uma iteração em uma janela de tempo e possui três fases.

No caso do algoritmo SPIM, na fase de requisição (*Request Phase*) cada porta de entrada envia uma requisição apenas para uma porta de saída, em vez de enviar requisições para todas as portas. Como apenas uma requisição foi feita, a fase de aceitação (*Accept Phase*) pode ser eliminada, pois, no máximo uma saída será concedida para cada entrada. Na segunda fase, a fase de concessão (*Grant Fase*), se uma saída tem mais de uma requisição ela escolhe uma aleatoriamente e notifica a entrada. Na terceira fase, a saída notifica seu estado de reservação às entradas. Isto é feito rapidamente, usando apenas um bit. Se cada entrada conhece o estado de reservação das portas de saída, na próxima iteração elas enviam requisições apenas para as portas sem reserva, aumentando a rapidez no procedimento de reserva.

O algoritmo SPIM tem implementação mais simplificada que o algoritmo PIM e alcança vazão maior que 98% em quatro interações, similar ao conseguido pelo algoritmo PIM, enquanto reduz a falta de equidade na alocação da banda de passagem observada no algoritmo PIM.

Algoritmo SLIP-IRRM (Iterative Round-Robin Matching with SLIP)

O algoritmo SLIP-IRRM foi apresentado em [22], é similar ao PIM e também possui três fases, com diferentes disciplinas para as fases de pedido, concessão e aceitação de requisições:

1. Fase de Requisição (*Request Phase*): Cada entrada não combinada envia uma requisição a cada saída para a qual ela tenha uma célula armazenada.
2. Fase de Concessão (*Grant Fase*): Caso uma saída não combinada tenha recebido mais de uma requisição, então ela escolhe a requisição que estiver mais próxima daquela apontada pelo seu ponteiro de *round-robin*. O ponteiro de *round-robin* é incrementado apenas se essa saída receber o sinal de aceitação na terceira fase.
3. Fase de Aceitação (*Accept Phase*): Se uma entrada não combinada receber mais de uma concessão, então ela escolhe a concessão que estiver mais próxima à

posição apontada pelo ponteiro de Round-Robin de aceitação e esse ponteiro é incrementado.

Do mesmo modo que no algoritmo PIM, o SLIP-IRRM pode executar mais de uma iteração em cada janela de tempo.

Algoritmo IRRM-MC (Iterative Round-Robin Matching with Multiple Classes)

Este algoritmo, apresentado em [24], é uma variação do algoritmo SLIP-IRRM, e inclui facilidades para manter a qualidade de serviço exigida pelas diferentes classes de serviço das redes ATM. A estrutura do comutador, adaptada para o uso do algoritmo IRRM-MC é mostrado na Fig. 2–5.

Conforme é ilustrado na Fig.2–5, em cada porta de entrada, é colocado um conjunto de cinco buffers, um buffer para cada categoria de serviço ATM: CBR, rtVBR, nrtVBR, ABR e UBR. As células que chegam a uma porta de entrada são discriminadas de acordo com sua classe de serviço e armazenadas nos buffers correspondentes. Em uma primeira iteração, o algoritmo examina os buffers da classe de serviço CBR para combinar as entradas e as saídas. Se ainda houver entradas não combinadas, o algoritmo executa uma nova iteração, utilizando os buffers rtVBR. O processo é repetido até que todas entradas que tenham células esperando estejam combinadas.

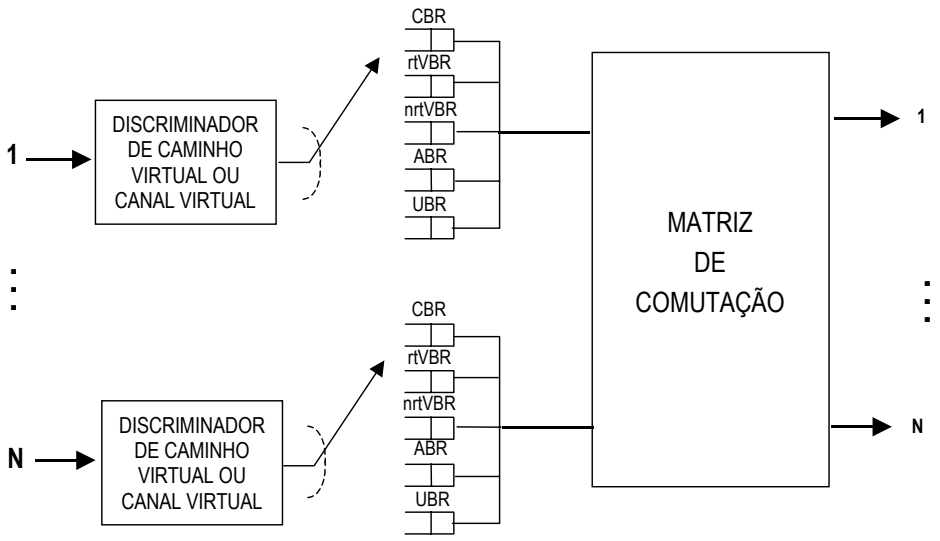


Figura 2-5: *Estrutura do Comutador com o Algoritmo IRRM-MC*

Algoritmo Faixa Ponderada

Apresentada em [28], a estrutura do comutador usada com o algoritmo de Faixa Ponderada é mostrada na Fig. 2–6, e é baseado no algoritmo IRRM-MC.

O algoritmo IRRM-MC utiliza um esquema de encaminhamento de células de acordo com as prioridades das classes de serviço, o que pode penalizar serviços com grande exigência de faixa de passagem, que porventura tenham sido colocados em classes de serviço menos prioritárias. No algoritmo de Faixa Ponderada, em cada entrada é colocado um conjunto de três buffers, um buffer para faixa estreita, um buffer para faixa média e um buffer para faixa larga. Um esquema de ponteiro do tipo round-robin é então utilizado, porém as visitas a cada buffer são feitas de forma ponderada. Ou seja, é atribuído um peso a cada buffer de uma dada entrada e este peso determina o número de visitas que cada buffer receberá do ponteiro de round-robin em um dado ciclo.

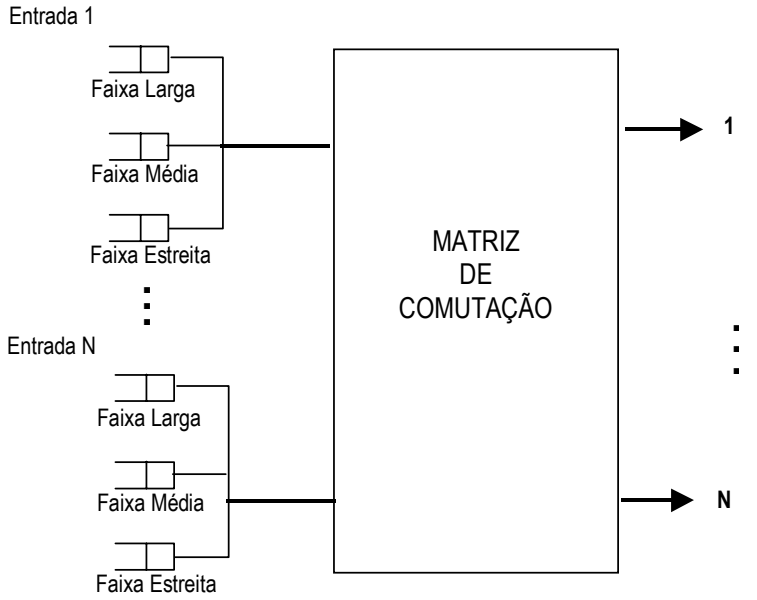


Figura 2-6: *Estrutura de Comutador usado com o algoritmo Faixa Ponderada.*

Comutador com buffers na entrada e escalonamento de saída.

Em [29], é apresentada uma proposta para solucionar o problema do bloqueio provocado pela célula HOL, usando um algoritmo de escalonamento de células, sem que seja necessário o aumento da velocidade interna da estrutura de comutação nem o aumento de complexidade do hardware. A estrutura proposta é mostrada na Fig.2–7 e

possui portas de entrada e de saída, uma matriz de comutação que provê roteamento espacial não bloqueante, além de um módulo centralizado de controle de contenção. Cada linha de entrada tem sua própria RAM e seu controle de fila de entrada. Usando informações do controle da fila de entrada, o módulo de controle centralizado de contenção agenda a transmissão das células de entrada. O comutador espacial de roteamento próprio recebe a célula da porta de entrada e transfere a célula para a porta de saída apropriada, usando o controle próprio de roteamento, baseado nas informações de roteamento físico.

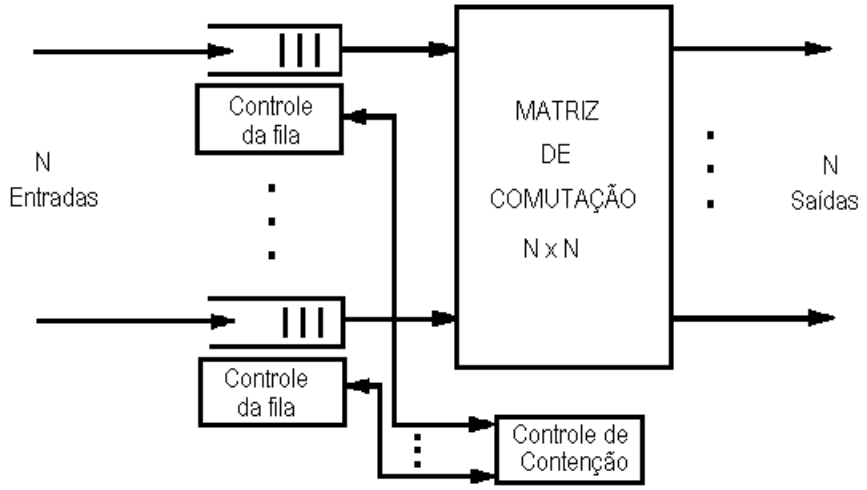


Figura 2-7: *Comutador com buffers na entrada e algoritmo de escalonamento de saída.*

Na Fig.2–8 é mostrado o diagrama de blocos simplificado de uma fila. O bloco RAM1 representa a memória de acesso aleatório que armazena as células. O bloco RAM2 também é uma memória de acesso aleatório e mantém as informações de instantes de transmissão atribuídas pelos escalonadores das portas de saída. O endereço T_A de RAM2 contém o endereço de RAM1 da célula agendada para envio no instante t_a. O buffer RAM2 é lido seqüencialmente, fornecendo os endereços apropriados para RAM1 nos instantes de transmissão agendados. O buffer RAM3 mantém a lista de todas as localizações de memória vagas em RAM1 e fornece o endereço de escrita de RAM1.

O módulo de controle de contenção consiste de um conjunto de escalonadores de saída independentes, cada qual encarregado de efetuar o escalonamento de uma única saída, simplificando a operação do comutador e aumentando sua velocidade. Conforme ilustrado na Fig. 2–9, o algoritmo de escalonamento tem duas fases: a fase de requisição e a fase de arbitragem. Na fase de requisição, as filas de entrada enviam as requisições aos escalonadores das portas de saída correspondentes. Suponha por exemplo que existam R_j requisições para a porta de saída *j*. Na fase de arbitragem, o

elemento escalonador da saída j, que mantém em um contador a próxima janela de tempo disponível (T_j) para a transmissão de uma célula, passa o instante de tempo t atribuído para a porta de entrada requisitante, ($T_j < t < T_j + R_j - 1$) e atualiza T_j para $T_j + R_j$. A implementação é simples: os instantes de envio são calculados através de um circuito somador que conta o número total de requisições, as quais são adicionadas do valor atual da "próxima disponibilidade" do circuito contador.

Uma vez que uma entrada tenha recebido uma atribuição de um escalonador de saída, a entrada checa sua tabela de envio. Se a tabela de envio não está reservada no instante t_a, ela torna-se reservada no instante t_a e a célula será enviada nesse instante. Se outra célula já tiver sido agendada para transmissão no instante t_a (por outra saída), existe um conflito de escalonamento e a célula retornará à fase de requisição na próxima janela de tempo, para que lhe seja atribuído um instante de transmissão diferente. Como o processo de escalonamento é executado com base na disciplina FIFO, a seqüência das células é mantida.

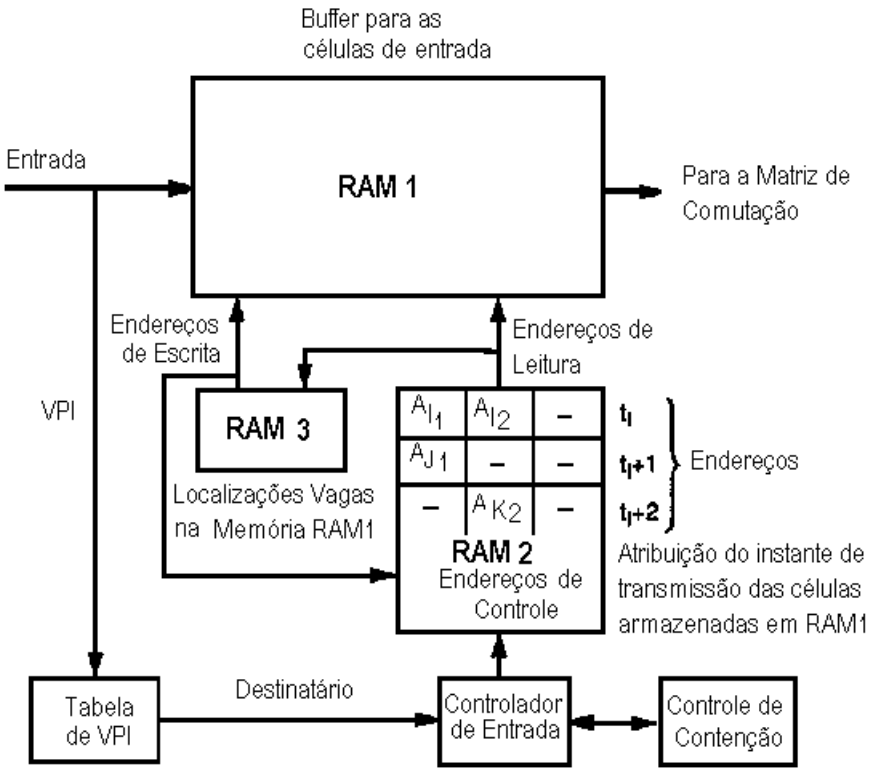


Figura 2-8: *Diagrama de Blocos de uma Fila de Entrada, em um comutador com buffers na entrada e escalonamento na saída.*

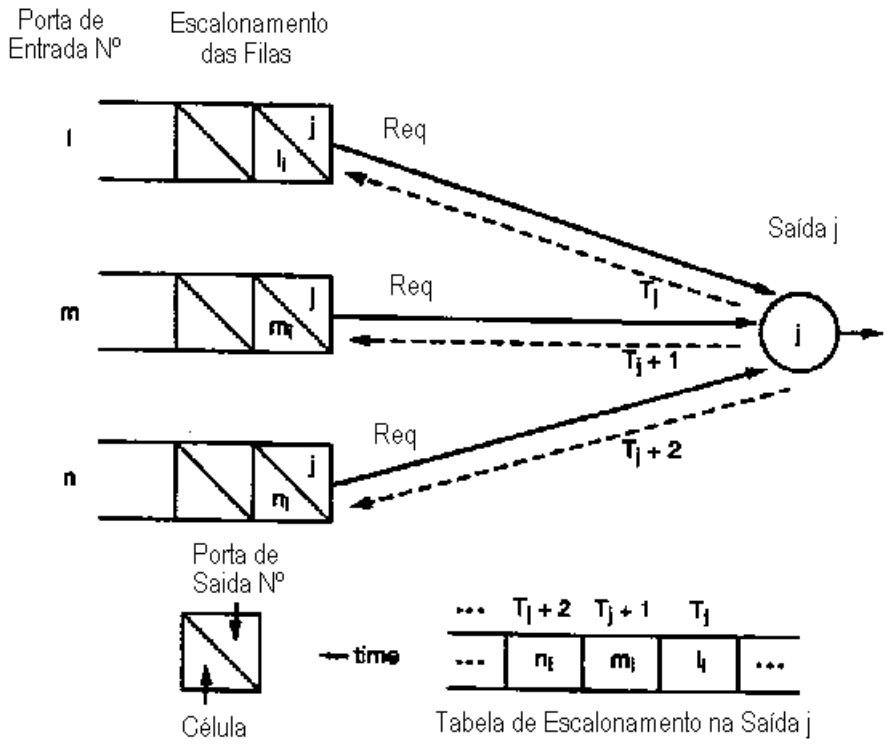


Figura 2-9: *Algoritmo de Escalonamento, implementado independentemente em cada saída.*

Observe que múltiplas saídas podem ser agendadas simultaneamente a uma única porta de entrada, porque não existe coordenação entre os vários escalonadores de saída. Para tráfego aleatório, o conflito de escalonamento entre escalonadores de saída independentes possibilita vazão máxima de aproximadamente 65%. A vazão é penalizada porque, quando um escalonador *j* atribui uma janela de tempo a uma entrada, ele assume que essa janela de tempo será usada para transmitir a célula. Se a entrada transmitir outra célula pela saída *k* reservada anteriormente, por exemplo, então a janela de tempo atribuída pelo escalonador j será perdida.

Para minimizar a degradação da vazão, provocada pelo uso de escalonadores de saída independentes, um agrupamento de portas de entrada poderia ser usado. Nesse caso, um grupo de *k* portas de entrada compartilham uma fila de transmissão comum ou cada fila de entrada opera *k* vezes mais rápido [30] que a velocidade das portas. Com o grupamento, *k* células podem ser enviadas simultaneamente. Com um grupo de tamanho *k=4*, a vazão máxima pode alcançar 82%.

Em [31] os algoritmos apresentados em [29] e [30] são modificados, aumentando-se substancialmente a vazão do comutador com buffers na entrada usando escalonadores independentes de portas de saída. O algoritmo de [31] torna os

escalonadores de saída recicláveis, ou seja, capazes de reassociar os tempos de transmissão das entradas, que não foram usadas em atribuições anteriores. O algoritmo é modificado de forma que, quando um escalonador de saída atribui um instante de transmissão a uma entrada, que não pôde ser usado porque ocorreu conflito de escalonamento, a entrada faz uma nova requisição para a mesma porta de saída e então retorna para o escalonador de saída a janela de tempo previamente designada. O autor se refere a esse processo como janela de tempo reciclada pela saída. O escalonador de tempo armazena a janela de tempo reciclada em um *latch* ou em uma memória FIFO, para usá-la em requisições futuras. A informação sobre a janela de tempo é apagada, se o tempo se esgota sem que ela tenha sido atribuída a outra entrada.

A proposta de [31], cuja estrutura está ilustrada na Fig. 2–10, adiciona hardware necessário para que os escalonadores de saída possam armazenar (e mais tarde reassociar) a janela de tempo reciclada, além de modificar a disciplina da fila de entrada. Para manter a seqüência das células em cada par de entrada/saída, a fila de entrada consiste de filas lógicas FIFO, correspondentes para cada porta de saída, que são representadas pelos blocos RAM2 e $FIFO_j$, (com j=1, 2, ... , N e onde N é o número de portas de saída) da Fig.2–10. O buffer $FIFO_j$ contém uma lista *first-in first-out* de endereços da RAM1 com as células destinadas à saída j. A RAM2 é uma memória de escrita aleatória e leitura seqüencial em que cada entrada contém a lista de portas de saída, caso uma mesma entrada tenha sido escalonada para transmitir células em janelas de tempo sucessivas. Na Fig. 2–10 por exemplo, OP1 e OP3 no endereço t_i indicam que a entrada (ou grupo) está agendada para transmitir células nas portas de saída 1 e 3 no instante t_i. O sinal de saída de RAM2, ou seja OP1, ativa a porta de saída da $FIFO_1$ e lê o endereço da próxima célula designada para a porta 1, e então, a retira da fila de entrada.

Conceitualmente, a modificação feita em [31] no algoritmo de [29] e [30] introduziu alguma coordenação entre as várias saídas independentes. Embora um escalonador de saída possa associar uma janela de tempo particular para um dado buffer de entrada sem saber se ele já foi associado a outra saída na mesma janela de tempo, o escalonador pode "aprender" algo com a ação dos outros escalonadores através das janelas de tempo recicláveis. A análise de desempenho feita em [31] mostrou que a vazão dessa estrutura está em torno de 92% e pode ser aumentada para 95%, caso o mecanismo de agrupamento das entradas seja usado.

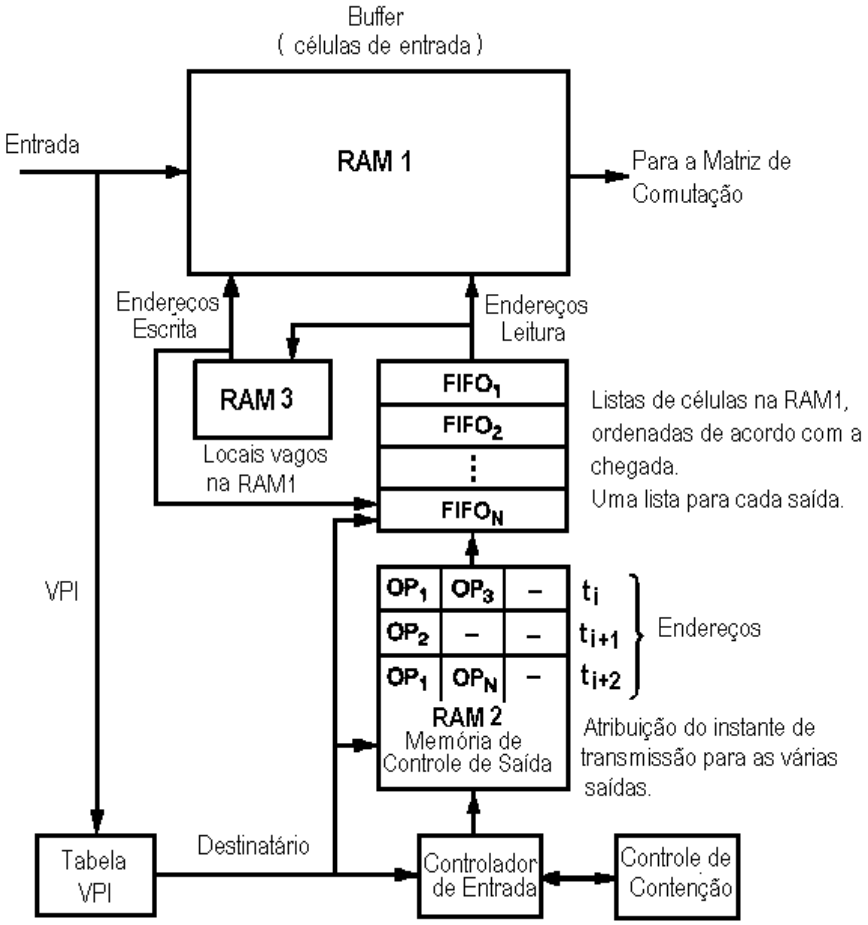


Figura 2-10: *Diagrama de Blocos de uma Fila de Entrada, em um comutador com buffers na entrada e escalonamento na saída usando Janelas de Tempo recicláveis e Enfileiramento de Saída Virtual (VOQ)*

2.3.2 Comutadores ATM com Memória Compartilhada

A estrutura de um comutador com memória compartilhada é mostrada na Fig.2–11. Nessa estrutura, as células que chegam são multiplexadas e armazenadas em um espaço de memória comum, compartilhado pelas portas de entrada e pelas portas de saída. Assim, essa estrutura apresenta a vantagem de otimizar o espaço de armazenamento disponível, porém, o fato de diferentes portas de entrada armazenarem células de forma seqüencial, torna necessário a implementação de um esquema de gerenciamento de memória mais complexo.

Em um comutador ATM de alta velocidade, a implementação de uma estrutura clássica de memória compartilhada é impraticável, pois as células que serão encaminhadas necessitam ter acesso à memória tanto para leitura como para escrita,

tornando o tempo de acesso à memória em fator limitante. Se a velocidade da porta é V bits por segundo, então a banda de passagem interna, necessária para a estrutura de memória compartilhada é 2*N*V em um comutador de N entradas e N saídas, pois, em uma janela de tempo é necessário a execução de N operações de escrita e N operações de leitura na memória compartilhada. Um controlador central processa seqüencialmente N pacotes de entrada e seleciona N pacotes de saída em uma janela de tempo. A limitação de velocidade de acesso à memória pode ser resolvido pelo uso de memórias organizadas em paralelo. Existem limitações relacionadas ao número de bancos de memória, que podem ser usadas em paralelo, devido ao comprimento fixo dos pacotes e existe um ponto em que o tempo de acesso à memória se torna um gargalo. Portanto, o tamanho de um comutador de memória compartilhada é limitado pela velocidade da memória disponível e pela velocidade de processamento conseguida [19].

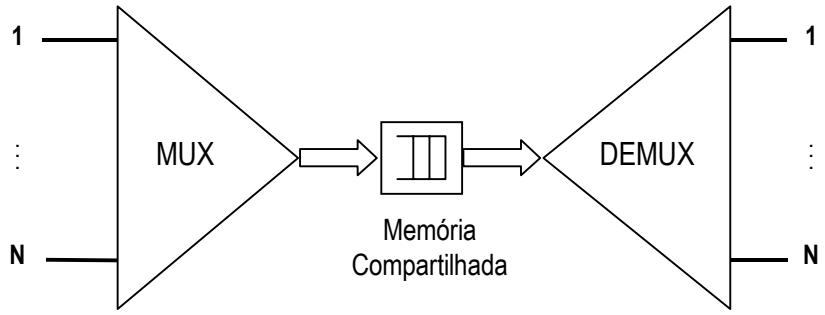


Figura 2-11: *Estrutura de Comutador ATM com Memória Compartilhada.*

Os comutadores de memória compartilhada podem ser construídos inteiramente por hardware, usando circuitos integrados de larga escala (LSI *Large Scale Intergrated*) para possibilitar a alta velocidade necessária aos comutadores ATM. Outro critério de projeto para comutadores com memória compartilhada é o tamanho da memória necessária, que não é apenas função do tamanho do comutador N, da carga ρ e do padrão de tráfego, mas também da forma em que a memória é compartilhada entre as várias filas de saída. A memória pode ser compartilhada de duas maneiras [19], pelo particionamento completo ou pelo compartilhamento completo.

Particionamento Completo: nesse caso, cada fila de saída tem um determinado montante de memória reservado a ela. Um pacote destinado a uma saída particular é perdido, caso a seção de memória reservada à fila de saída correspondente estiver cheia, mesmo que exista espaço de memória livre em outras filas de saída. Isto resulta em uma utilização ineficiente do espaço de memória para o caso de tráfego em rajadas, mesmo que a memória tenha sido distribuída de forma equilibrada entre as saídas.

Compartilhamento Completo: nesse caso, o montante de memória é compartilhado completamente entre as N saídas, o que proporciona maior eficiência na taxa de utilização da banda de passagem, devido ao compartilhamento estatístico. O compartilhamento completo necessita menor quantidade de memória que o particionamento completo, para a mesma taxa de perda de células. Entretanto, o espaço de memória pode ser completamente tomado pelo congestionamento provocado por rajadas de células pertencentes a poucas chamadas. Ou seja, células de poucas entradas poderão ocupar inteiramente a memória, excluindo as outras entradas. A falta de eqüidade presente para os padrões de tráfego em rajadas é a principal desvantagem deste esquema.

Os fatores compartilhamento e eqüidade variam com o comprimento da rajada do tráfego de entrada. Assim, o particionamento completo pode ser melhor para alguns ambientes de tráfego e o compartilhamento completo em outros ambientes.

O comutador Prelude [32] é um exemplo de comutação por memória compartilhada que emprega o particionamento completo do espaço de memória. Ele foi desenvolvido no *Centre National d'Etudes des Telecommunications* (CNET) na França entre 1982 e meados de 1987. Este comutador é uma versão assíncrona de um comutador de janelas temporais usado em redes de comutação de circuito tradicionais. Os pacotes são extraídos de um relógio de alinhamento de fase adaptado, de forma que os pacotes sejam deslocados em um byte no tempo, de uma linha de entrada para a próxima. Depois, os pacotes entram em um estágio de super-multiplexação, que consiste de um comutador por divisão de espaço rotativo que assume, ciclicamente, 16 diferentes padrões de comutação. O ciclo rotativo é sincronizado com os pacotes de entrada, de forma que o cabeçalho do pacote de entrada seja multiplexado na primeira saída e os bytes restantes possam ser seqüencialmente multiplexados nas linhas restantes. Dessa forma, os pacotes são paralelizados sobre as N linhas e alimentam os N bancos de memória. Os cabeçalhos são processados pelos controladores que determinam a porta de saída de cada pacote. O novo cabeçalho é colocado no primeiro banco de memória. O ponteiro do pacote é então colocado na fila da linha de saída correspondente. A demultiplexação e a serialização dos pacotes nas linhas de saída é feita de maneira similar pelo comutador de divisão espacial rotativo. O diagrama de blocos do comutador Prelude é mostrado na Fig. 2–12.

O comutador proposto em [33] e [34] é um exemplo de comutador de memória compartilhada que usa compartilhamento completo do espaço de memória. As filas são organizadas na forma de listas encadeadas. Isto é realizado através de seis tipos de circuitos integrados em larga escala, ou circuitos LSI. Os circuitos I/O LSIs executam a conversão série-paralelo e paralelo-série, enquanto que o *chip* de conversão do cabeçalho (HD CNV) processa o número do circuito virtual de cada pacote. Os *chips* de

comutação (SW) consistem de um multiplexador (MUX), demultiplexador (DMUX) e memória. Os *chips* de controle (CTRL) contém N registradores com endereços para escrita (WA), e N registradores com N endereços para leitura (RA), um par para cada lista encadeada. *Chips* de endereços vazios (IA BF) mantêm a lista de localizações vazias no buffer.

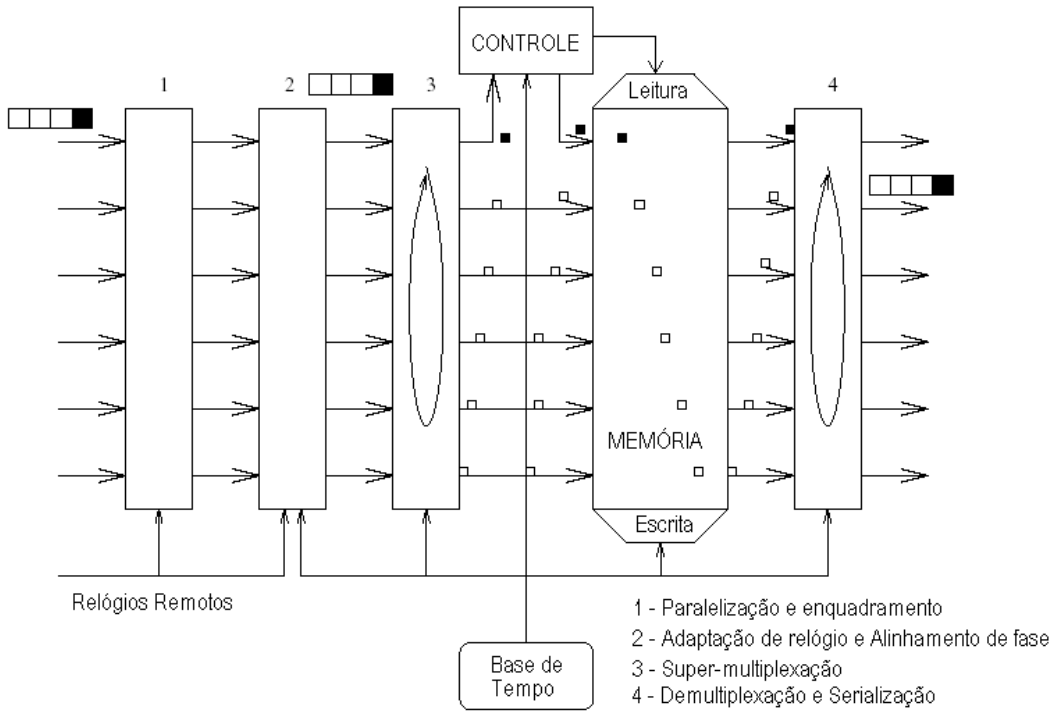


Figura 2-12: *Estrutura do comutador Prelude*

Para cada pacote que chega na entrada de um comutador, após a conversão do cabeçalho que determina em qual lista encadeada o pacote será colocado, o registrador WA apropriado é acessado para receber a localização de memória daquele pacote. O novo buffer vazio é identificado e fornecido pelos *chips* IA BF. Então o registrador WA e todos os ponteiros apropriados são atualizados. De modo similar, a cada janela de tempo, um buffer de cada lista encadeada é identificada usando os registradores RA, recuperada, e transmitida, simultaneamente atualizando os ponteiros e os conteúdos dos buffers de endereços vazios. As exigências de velocidade são satisfeitas usando vários *chips* SW na configuração de memória organizadas em paralelo. O diagrama de blocos do comutador proposto em [33] é mostrado na Fig.2–13.

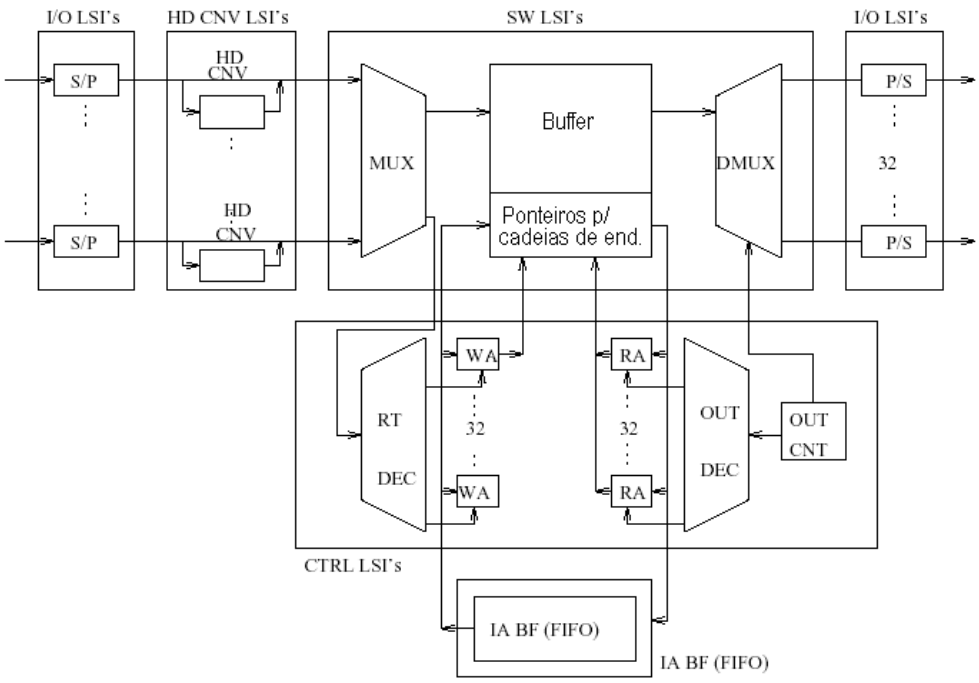


Figura 2-13: *Estrutura básica do comutador de memória compartilhada de Hitachi.*

2.3.3 Comutadores ATM com *buffers* na saída

Na Fig. 2–14 é mostrada a estrutura de um comutador com *buffers* na saída. As células que chegam às portas de entrada da estrutura são armazenadas diretamente nos buffers das respectivas portas de destino. Esta configuração permite uma vazão teórica máxima de 100%. Entretanto, o comutador deve operar a uma velocidade diferente das entradas e saídas e os elementos de memória devem ser suficientemente rápidos para armazenar nos buffers de saída, todas as células que chegam na portas de entrada, durante uma janela de tempo.

O comutador *knockout* [35], desenvolvido pela ATT Bell Laboratories em 1987 usa estrutura de comutação de *buffers* na saída. A estrutura de comutação do comutador *knockout* têm duas características básicas, que são: cada entrada tem um barramento separado e cada saída tem acesso a todos os pacotes que chegam em todas as entradas. Na Fig.2–15 está ilustrado estas duas características em que, cada uma das N entradas é ligada diretamente a um barramento separado e cada interface passiva de saída completa o conjunto dos N barramentos.

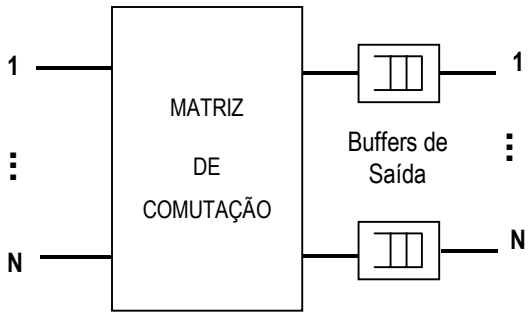


Figura 2-14: *Estrutura de Comutação com buffers na saída.*

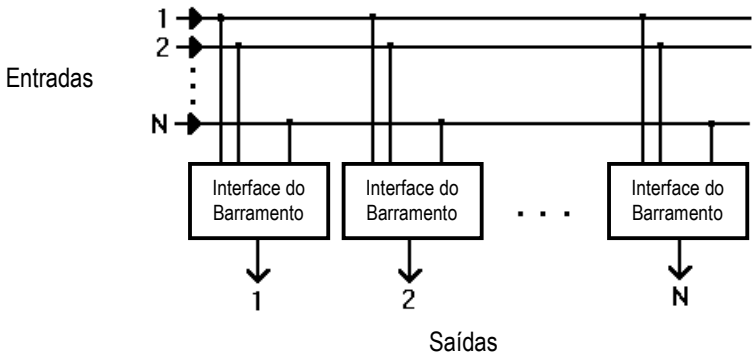


Figura 2-15: *Estrutura de Interconexão do Comutador Knockout*

A cada célula que chega, é feita a verificação do circuito virtual ao qual ela pertence e, utilizando-se filtros de pacotes, o caminho correto é habilitado. A célula é enviada a todas as portas, porém apenas a porta habilitada recebe a célula. É possível enviar, simultaneamente, uma célula para várias portas, bastando apenas habilitar as portas corretas.

Como não é feito escalonamento de células, é inevitável que as células que chegam nas várias entradas possam estar endereçadas à mesma porta de saída e a forma mais simples de se evitar colisões seria armazenar todas as células nas saídas correspondentes. Porém para um comutador de 1024 portas seria necessário 1024 *buffers* em cada saída. Na prática, a possibilidade de utilização simultânea dos 1024 *buffers* é mínima, então o número de buffers pode ser reduzido, introduzindo assim uma possibilidade de descarte de células. Quando a capacidade de controle for insuficiente para atender o número de células, o concentrador seleciona as células que serão armazenadas e as que serão descartadas. O concentrador é um circuito especializado para fazer a eliminação (*knockout*). A fila de saída é simulada por diversas filas. As

células selecionadas são enviadas para um deslocador (*shifter*) que as distribui entre as filas de saída. Na Fig. 2–16 está ilustrado a estrutura da interface de barramento associada a cada saída do comutador.

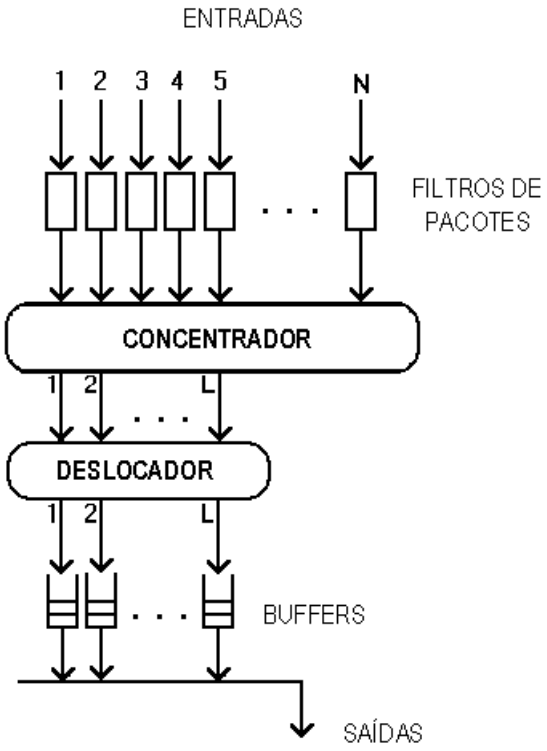


Figura 2-16: *Interface de Barramento do Comutador Knockout*

O custo do comutador pode ser negociado de acordo com a taxa de perda de células esperada.

Em [36] é apresentado uma proposta de comutador com buffer na saída baseado em três princípios: a) o princípio genérico do comutador *knockout* de agrupar as saídas para reduzir a necessidade de interconexões simultâneas; b) o enfileiramento na saída para possibilitar o melhor desempenho, melhorando a vazão e diminuindo os atrasos; e c) o uso de um algoritmo de roteamento de células eficiente para evitar o congestionamento e diminuir a complexidade de interconexão da estrutura adotada. A estrutura proposta favorece sua utilização em grandes comutadores, construídos a partir da interconexão de módulos de comutação menores, ou em ambientes de comutação com requisitos de crescimento da capacidade de comutação. Segundo o autor, a principal vantagem da estrutura proposta é sua compatibilidade para trabalhar com pacotes de tamanhos variáveis.

Um outro exemplo de comutador ATM com *buffer* na saída é apresentado em [37]. Este comutador possui facilidades para garantir banda de passagem diferenciada a fluxos individuais. O custo da redistribuição da banda de passagem e o desempenho dessa estrutura para escalonadores que usam a disciplina *round-robin* e para escalonadores baseados em informações contidas em *time-stamp* também são comparadas em [37].

2.3.4 Comutadores com buffers nos pontos de cruzamento.

A estrutura geral dos comutadores com *buffers* nos pontos de cruzamento, ou comutadores matriciais, está ilustrada na Fig. 2–17.

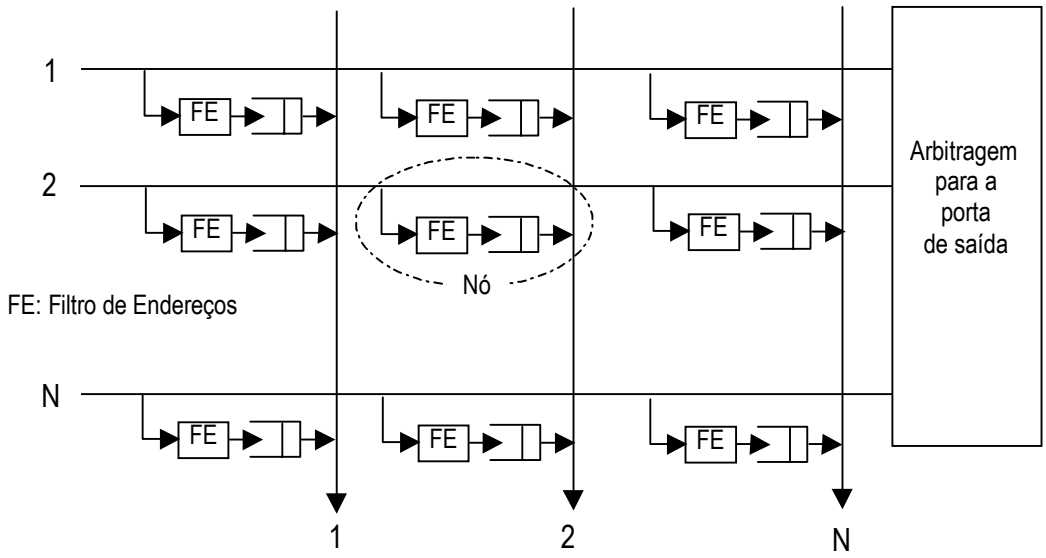


Figura 2-17: *Estrutura convencional do Comutador Matricial*

Cada nó de comutação da matriz possui um *buffer* para armazenar as células. Assim, as células que chegam pelas portas de entrada são diretamente enviadas para os *buffers* dos nós após serem selecionadas através de um filtro de endereços. A vazão do comutador matricial é equivalente ao obtido pelo comutador com *buffers* na saída. A desvantagem dessa estrutura é o tamanho do hardware, que é proporcional a N^2, em um comutador com N portas de entrada e N portas de saída. Um comutador que usa essa estrutura foi descrito em [38].

2.3.5 Comutadores com Estruturas Mistas.

Comutador Knockout com Buffers na Entrada.

Um comutador com *buffer* na entrada, baseado no Comutador *Knockout* é descrito em [39]. Conforme está ilustrado na Fig.2–18, a estrutura do Comutador *Knockout* com buffers na entrada é semelhante à estrutura do Comutador *Knockout* tradicional, na qual apenas foi introduzido um buffer em cada entrada, simplificando assim a complexidade do hardware envolvido. As células que chegam, esperam nos buffers de entrada até alcançarem a primeira posição do buffer. As células que estiverem na primeira posição do buffer de entrada e que forem destinadas à mesma porta de saída competem. Essa competição ocorre no concentrador, pois ele possui um número limitado de saídas. Somente as células vencedoras são transferidas aos respectivos buffers de saída. As outras células permanecem nos buffers de entrada até serem transmitidas, não sendo necessário o descarte dessas células nos concentradores.

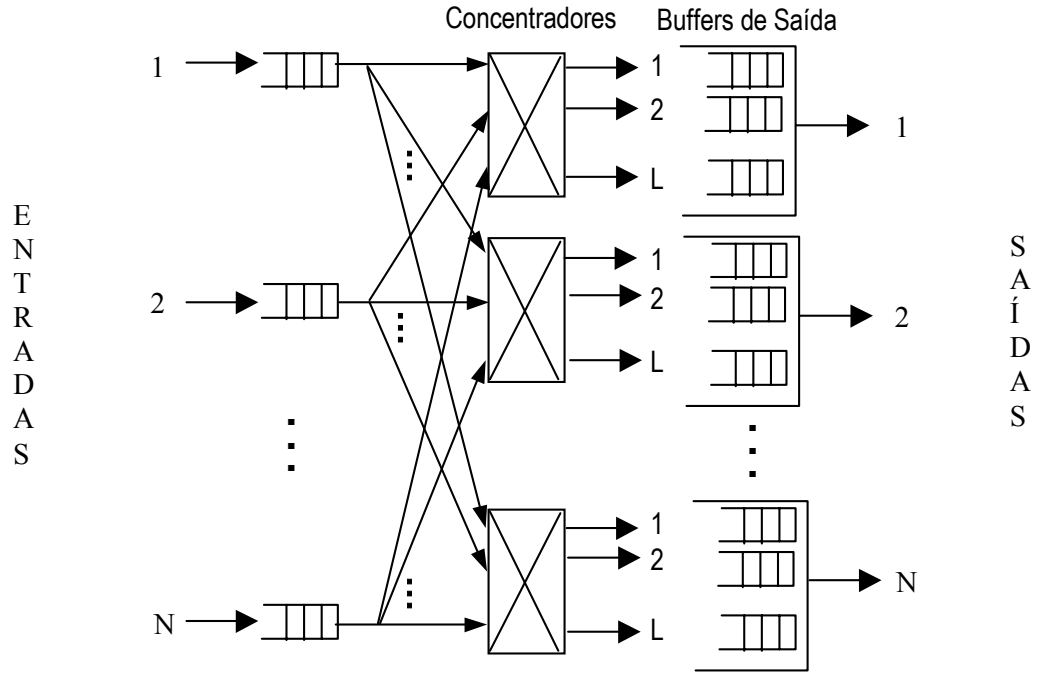


Figura 2-18: *Estrutura de um Comutador Knockout com buffers nas entradas*

O processo de seleção das células no concentrador é realizado através dos seguintes passos:

- É copiado o cabeçalho de cada célula a ser transferida e que contém o endereço do destino;
- A cópia do cabeçalho da célula é enviada para o concentrador correspondente, para participar da competição;
- Uma mensagem de confirmação é retornada pelo caminho aberto pela cópia do cabeçalho, para que a célula vencedora seja transferida. As células que não receberem a confirmação permanecem no buffer.

Comutadores com tentativas estatísticas de alta velocidade (HSR High-Speed Statistical Retry)

Em [40] é apresentada uma estrutura para comutação em alta velocidade, baseada em uma matriz de comutação com buffers nas entradas e nas saídas, que está ilustrada na Fig.2–19(a). Nessa estrutura os buffers são colocados nas portas de entrada e nas portas de saída do comutador e são interligados pela matriz de comutação.

A freqüência de transmissão interna é m vezes a freqüência de entrada do comutador, com ($1<m<N$, onde N é o número de portas do comutador). Assim, define-se o período interno do comutador (t) como $t=T/m$.

Cada *buffer* de entrada transmite, repetidamente, uma cópia de uma célula até receber um sinal de confirmação do *buffer* de saída. Essa confirmação indica o sucesso da transmissão da célula para o *buffer* de saída. Durante cada janela de tempo, apenas uma célula pode ser lida de cada buffer de entrada. Caso não ocorra um sinal de confirmação, o buffer de entrada retransmite uma cópia da célula na próxima janela de tempo.

Cada nó da matriz de comutação, Fig.2–19(b), consiste de um filtro de endereços (FE), um seletor (SEL), uma chave que é usada para enviar o sinal de confirmação (ACK) para os buffers de entrada, e um sinal de encaminhamento (RS) enviado pelos buffers de entrada. O sinal de encaminhamento é enviado para identificar a porta destino, ao mesmo tempo que a célula também é enviada. Este sinal é usado pelo filtro de endereços para determinar a porta de saída da célula. Se mais de uma célula for enviada a uma mesma saída, o Seletor utiliza o esquema predefinido de prioridades, para determinar o buffer de entrada que receberá a confirmação.

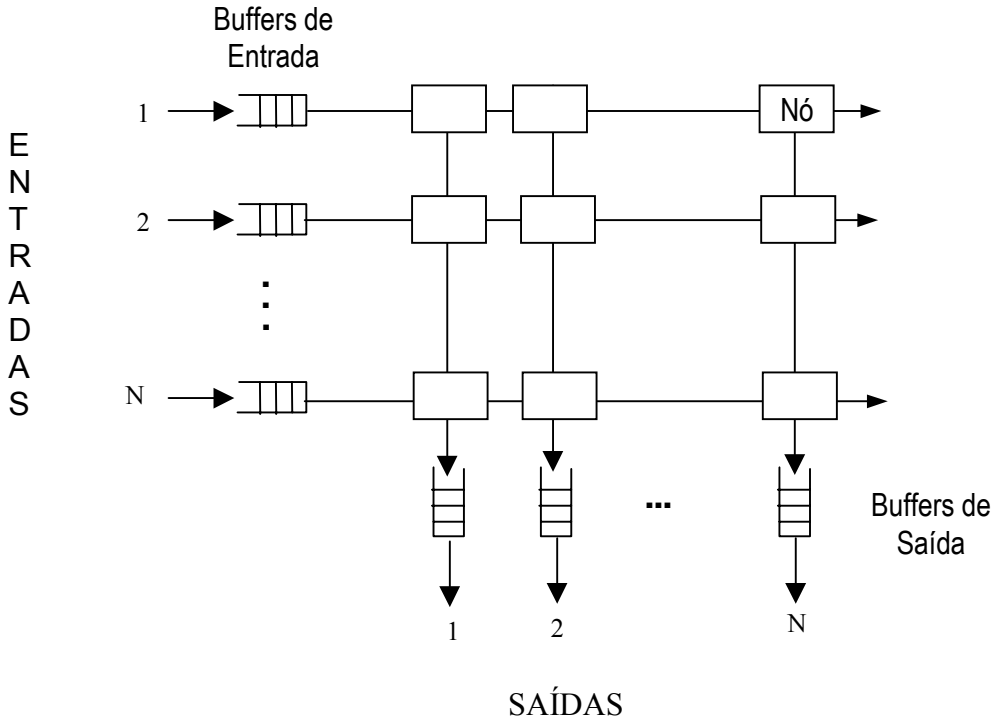


Figura 2-19 (a) Estrutura do Comutador HSR

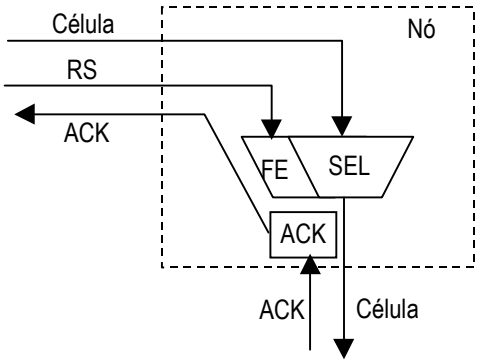


Figura 2-19*(b) Diagrama simplificado do nó da matriz de comutação do comutador HSR*

Comutador com buffers na entrada e nos pontos de cruzamento

Uma estrutura para um comutador matricial com buffers nas portas de entrada e nos pontos de cruzamento foi apresentada em [44] e está ilustrada na Fig.2–20. A velocidade interna da estrutura é a mesma velocidade das portas de entrada. O algoritmo utilizado para dividir a carga entre os buffers de entrada e dos nós de cruzamento é simples e está ilustrado na Fig. 2–21. A estrutura utiliza uma linha de

controle que conecta os buffers de entrada com os buffers dos nós da mesma linha da matriz.

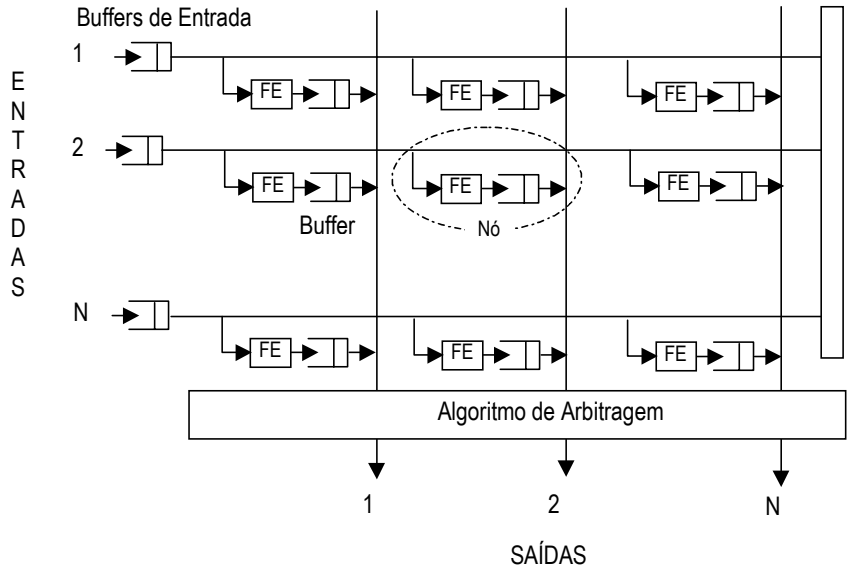
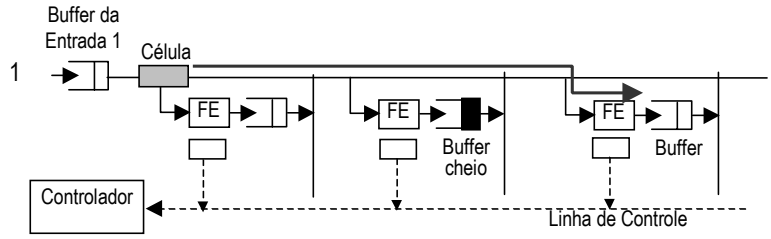


Figura 2-20: *Estrutura de comutador com buffers na entrada e nos pontos de cruzamento.*

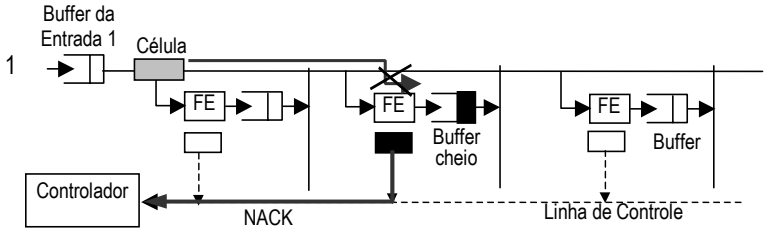
Todas as células que chegam ao comutador são armazenadas nos seus buffers das portas de entrada. Uma cópia do cabeçalho da célula armazenada no buffer da entrada é enviada para os nós da matriz. Um filtro de endereços (FE) é utilizado para selecionar a célula cujo endereço coincide com o endereço da porta de saída. É considerado sucesso de transmissão, se após um tempo de espera determinado, o buffer da entrada não receber um sinal de NACK (*Not Acknowledge*) da linha de controle. Em caso de sucesso de transmissão da célula ao respectivo buffer da matriz, a célula no buffer de entrada é removida. Um esquema de arbitragem é utilizado em cada coluna para efetuar a decisão de envio de células para os enlaces de saída em cada coluna da matriz. Na Fig. 2–21(a) está representado a transmissão de uma célula do buffer de entrada para o buffer do nó, da porta de saída correspondente.

Como está ilustrado na Fig. 2–21(b), pode acontecer que um buffer do ponto de cruzamento (ou nó) esteja ocupado e não possa receber as células que foram enviadas pelo buffer de entrada. Nesse caso, o buffer do nó envia pela linha de controle, um sinal de NACK para indicar buffer cheio e o buffer de entrada cancela a transmissão da cópia. A situação de bloqueio do nó pode ser implementada utilizando uma porta AND, que compara o sinal de buffer cheio com o endereço da porta de saída, para a qual a célula deverá ser encaminhada. O algoritmo utilizado para transmissão das células é muito simples e não necessita que seja feita qualquer negociação entre as entradas, ou que os tempos de transmissão das células sejam negociados antes da fase de

transmissão das células. Esta estrutura pode ser implementada usando memórias FIFO, uma função para cópias de células e um sistema de controle extremamente simples. O bloqueio de células, possível em caso de contenção extrema da saída, ocorre apenas nos buffers de entrada, enquanto que no comutador matricial convencional, esse bloqueio pode ocorrer nos nós.



(a) Transmissão de uma célula



(b) Bloqueio no buffer do ponto de cruzamento

Figura 2-21: *Transmissão de células entre buffers no comutador com buffers combinados na entrada e nos pontos de cruzamento.*

A vantagem da estrutura com buffers nas entradas e nos pontos de cruzamento é a diminuição do número de elementos de memória necessários para armazenar células. Para um comutador 32x32 e uma taxa de perda de células de 10^{-8}, essa estrutura requer um buffer nos nós da matriz de apenas 4 células e 32 células na entrada, enquanto que o comutador convencional requer 8 células nos nós da matriz para obter o mesmo desempenho. Ou seja, o tamanho do hardware pode ser reduzido quase à metade em relação ao tamanho do comutador convencional. Entretanto, a estrutura proposta ainda sofre de contenção provocada pela célula HOL.

No capítulo 3 serão apresentadas propostas para comutadores que combinam buffers nas entradas e nos pontos de cruzamento, mas diferentemente do proposto em [44], as estruturas que serão apresentadas apresentam facilidades para garantir a qualidade de serviço para as diversas classes de serviço previstas nas redes ATM. Considerações sobre o desempenho e tamanho do hardware necessário para cada proposta também serão discutidas no capítulo 3.

Comutador com buffers combinados na entrada e nos pontos de cruzamento que usam o algoritmo VOQ nos buffers de entrada.

Em [23] demonstrou-se que a utilização de filas FIFO separadas, uma para cada saída, em um comutador com buffers na entrada pode obter a vazão de 100% para processos de chegada de células independentes. Essas filas podem ser implementadas como filas de saída virtuais *(VOQ Virtual Output Queueing*), como foi apresentado em [31], por exemplo, e esse esquema elimina completamente o bloqueio provocado pela célula HOL. Quando o VOQ é usado é necessário a utilização de um algoritmo para determinar qual a célula que será transmitida. O problema com os algoritmos de escalonamento de células é o longo tempo necessário para determinar o casamento perfeito entre portas de entrada e portas de saída, que seja completamente livre de conflitos. Se o algoritmo não consegue evitar completamente o conflito, então a vazão é reduzida.

Em [43] é apresentado um comutador que usa uma estrutura combinada de buffers grandes nas entradas e buffers pequenos nos pontos de cruzamento, baseado em [44], porém o armazenamento de células nos buffers de entrada é feito de acordo com esquema VOQ [31], conforme mostrado na Fig.2–22. Essa estrutura usa dois algoritmos de escalonamento de células, que são também apresentados em [43], um algoritmo é usado para determinar a célula que será transmitida do buffer do ponto de cruzamento para a porta de saída e o outro algoritmo determina a célula que será transmitida do buffer da porta de entrada para o buffer do ponto de cruzamento.

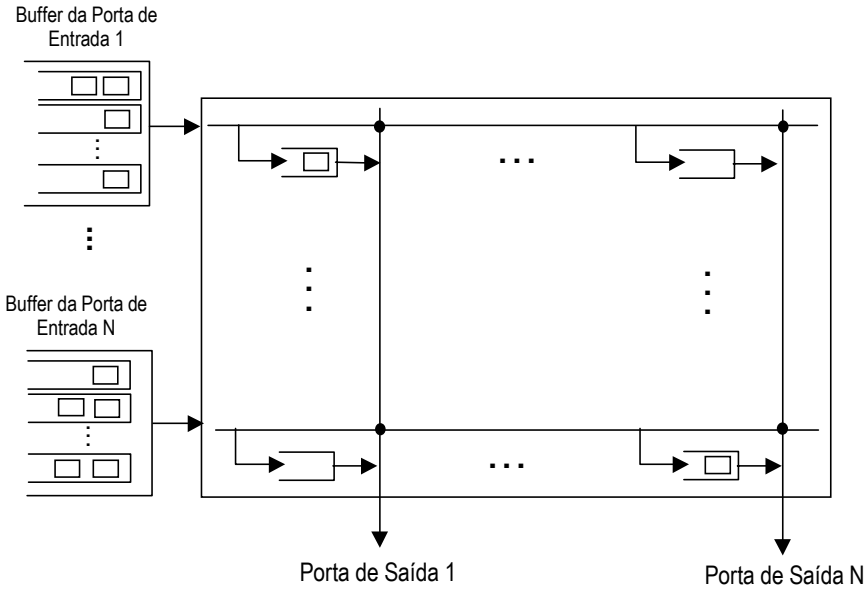


Figura 2-22: *Estrutura de comutador com buffers grandes nas portas de entrada e buffers pequenos nos pontos de cruzamento, usando escalonamento de células baseado em VOQ.*

O algoritmo de escalonamento usado para determinar a célula que será transmitida, do buffer do ponto de cruzamento para a porta de saída, está descrito na Fig. 2–23. O escalonador da porta de saída j seleciona o buffer da célula HOL que tem o maior tempo de atraso dentre todas as células que estão aguardando nas filas de entrada. Se mais de um buffer é selecionado, então a porta de entrada que tiver o menor identificador é escolhido. A célula selecionada é então transmitida pela porta de saída.

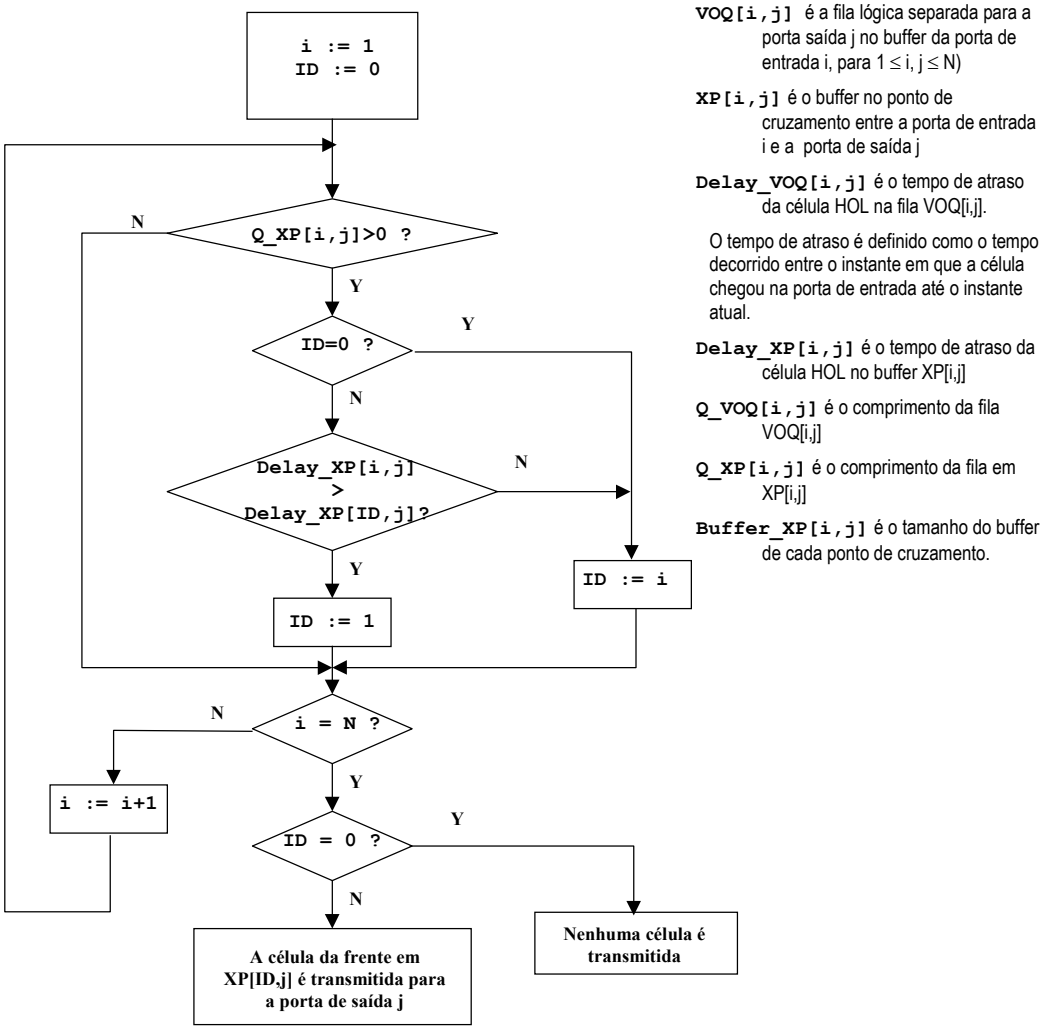
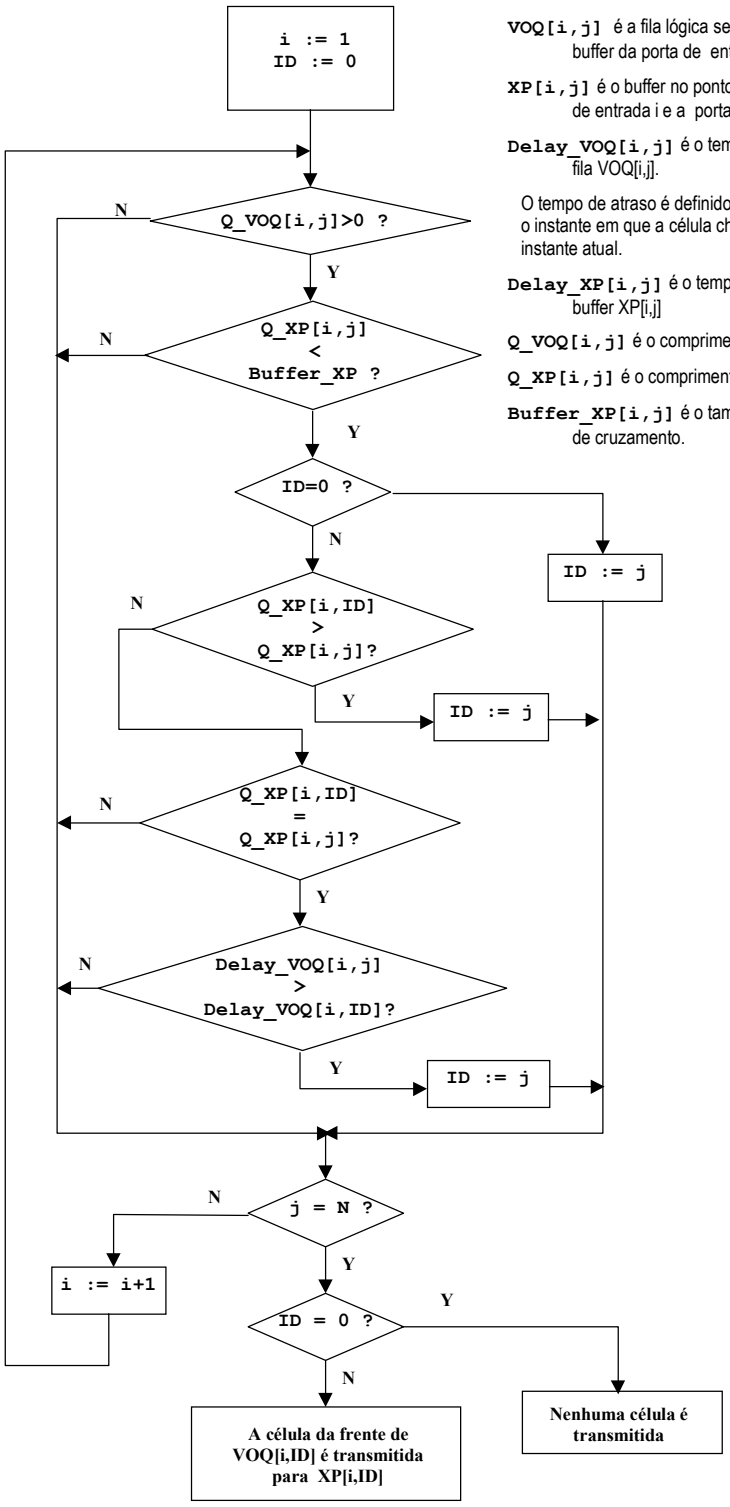


Figura 2-23: *Fluxograma do algoritmo de escalonamento que determina a célula que será transmitida de um buffer do ponto de cruzamento para a porta j.*

O algoritmo de escalonamento usado para determinar a célula que será transmitida, do buffer de entrada para o buffer do ponto de cruzamento, está descrito na Fig. 2–24.



VOQ[i,j] é a fila lógica separada para a porta saída j no buffer da porta de entrada i, para $1 \leq i, j \leq N$)

XP[i,j] é o buffer no ponto de cruzamento entre a porta de entrada i e a porta de saída j

Delay_VOQ[i,j] é o tempo de atraso da célula HOL na fila VOQ[i,j].

O tempo de atraso é definido como o tempo decorrido entre o instante em que a célula chegou na porta de entrada até o instante atual.

Delay_XP[i,j] é o tempo de atraso da célula HOL no buffer XP[i,j]

Q_VOQ[i,j] é o comprimento da fila VOQ[i,j]

Q_XP[i,j] é o comprimento da fila em XP[i,j]

Buffer_XP[i,j] é o tamanho do buffer de cada ponto de cruzamento.

Figura 2-24: *Fluxograma do algoritmo de escalonamento que determina a célula transmitida do buffer da porta de entrada i para o buffer do ponto de cruzamento.*

O escalonador da porta de entrada i seleciona a fila lógica separada que tem células e que tem a maior fila de espera. Se mais de uma fila lógica for selecionada, é escolhida aquela cuja célula tem o maior tempo de atraso. A célula selecionada é então transmitida para o buffer do ponto de cruzamento acoplado. Os resultados de simulação de [43] mostraram que: (1) a estrutura que usa buffers combinados nas entradas e nos pontos de cruzamento e estratégia VOQ apresentou melhor desempenho que a estrutura tradicional de buffers combinados (sem VOQ); (2) o tempo médio de atraso das células para cargas menores que 70% é menor que o apresentado pelo comutador com buffers na entrada; e (3) utiliza menos buffers que o comutador ATM com buffers apenas nos pontos de cruzamento.

O Capítulo 5 apresenta uma proposta para uma estrutura de comutador ATM com facilidade para garantir QoS, baseado na proposta de [43].

2.4 Comutadores de Pacotes com comprimento variável

Com a explosão do tráfego na Internet, os roteadores IP tornaram-se o grande gargalo da rede. Para aumentar o desempenho dos roteadores, muitos mecanismos novos foram propostos. Surgiram propostas para uso de mecanismos capazes de combinar a velocidade de um comutador da camada de enlace (camada 2) com a flexibilidade de um roteador de camada de rede (camada 3).

Entretanto, os comutadores de alta velocidade existentes foram desenvolvidos para transportar células ATM, que ao contrário dos pacotes IP, possuem comprimento fixo. Assim, encontrar algoritmos de escalonamento eficientes para comutar os pacotes IP, de comprimento variável, sobre os comutadores ATM é um dos desafios a serem vencidos para se obter comutadores IP de alta velocidade.

Em [46] é proposto um algoritmo chamado IP-PIM (*IP Parallel Iterative Matching*) para escalonamento de pacotes IP de comprimento variável em comutadores ATM. O desempenho do algoritmo proposto foi comparado ao desempenho do algoritmo PIM original e mostrou-se mais eficiente, principalmente para tráfego em rajadas. Nessa proposta, adota-se uma estrutura de comutação com buffers na entrada, em que cada entrada mantém uma fila separada de células destinadas a cada saída. Para selecionar uma das célula HOL que disputam a mesma saída, das diferentes filas de espera, é feita uma adaptação do algoritmo PIM, que é descrito na seção seguinte.

2.4.1 Algoritmo IP-PIM (IP Parallel Iterative Matching)

Quando um pacote IP é transmitido sobre comutadores ATM tradicionais, um pacote IP é segmentado em múltiplas células ATM e, depois, as células ATM são

transmitidas pela estrutura de comutação. Então, as células ATM pertencentes a um mesmo pacote IP são destinadas à mesma porta de saída. Motivado por essa observação, e também pela característica de rajadas do tráfego em redes IP, em [46] foi proposto uma modificação do algoritmo PIM, para executar o escalonamento de pacotes em vez de células. Cada pacote IP é segmentado em um número variável de células, que são chamadas "rajada de células". No algoritmo IP-PIM, cada rajada de células participa do escalonamento apenas quando a primeira célula da rajada está na posição HOL. A rajada, à qual a célula HOL pertence, é chamada "rajada HOL". Quando uma porta de entrada é associada a uma porta de saída pelo algoritmo PIM, esta associação é válida até que toda a "rajada HOL" tenha sido transmitida. Formalmente, o algoritmo de escalonamento IP-PIM fica:

1. São mantidas todas as associações feitas na última janela de tempo e que permanecem *válidas* na janela de tempo atual (ou seja, a última célula da rajada ainda não foi transmitida);
2. Cada entrada não associada envia uma requisição, com uma dada prioridade, a toda saída para a qual possui células de uma "rajada HOL" esperando nos buffers;
3. Se uma saída não associada recebe uma requisição, ela escolhe a de mais alta prioridade e envia uma concessão;
4. Se uma entrada recebe concessões, ela escolhe a de mais alta prioridade e notifica à saída esse aceite;
5. Os passos 2 a 4 são repetidos até que o máximo de associações possa ser realizado, ou até que um número especificado de iterações seja executado.

O passo 1 garante que todas as células de uma rajada (um pacote IP) sejam transmitidas pelo comutador ATM em "slots" contíguos e sem interrupção. Além disso, a probabilidade de que um grande número de "rajadas HOL" complete sua transmissão ao mesmo tempo é pequena. Como conseqüência, os passos 2 a 5 são executados apenas por um subconjunto de entradas e saídas e essa é a vantagem do escalonamento de pacotes IP em vez de escalonamento de células, uma vez que será necessário menos iterações para associar entradas e saídas.

O algoritmo IP-PIM degrada a taxa de perda de células de baixa prioridade, uma vez que células de um mesmo pacote são comutadas consecutivamente. Em [47] é proposto um algoritmo de escalonamento que dá prioridade a um buffer ativo para escalonar as células de um mesmo pacote de maneira eficiente, sem que o comprimento das filas aumentem. Entende-se por buffer ativo, o buffer que contém células de um determinado pacote em uma determinada janela de tempo.

2.4.2 Algoritmo de escalonamento com prioridade para o buffer ativo para pacotes IP de comprimento variável sobre comutadores ATM com buffers na entrada.

Esse algoritmo é usado em uma estrutura de comutação com NxN filas de entrada usando enfileiramento virtual de saída (VOQ). Denomina-se um estado ativo, o estado desde a chegada de uma célula HoP (*Header of Packet*) até a chegada de uma célula EoP (*End of Packet*). Denomina-se buffer ativo o buffer que está em um estado ativo. Os buffers ativos recebem maior prioridade. Se existir mais de um buffer ativo destinado a uma mesma saída, a célula que estiver na fila de maior comprimento é priorizada. A seguir, é dada prioridade aos buffers não ativos que possuem filas de maior comprimento. O algoritmo é aplicado em todos os buffers de entrada. Se essas condições são as mesmas, são atribuídas aleatoriamente três prioridades. Nos buffers ativos, a probabilidade de que as células cheguem na próxima janela de tempo é alta. Entretanto, como é dada prioridade ao buffer ativo que possui fila de maior comprimento, as células não serão perdidas. Como a probabilidade de uma célula HOL de um buffer ativo ser comutada é alta, a probabilidade das células de um mesmo pacote ser comutado consecutivamente também é alta. Assim, o atraso médio de comutação do pacote não será grande. Por outro lado, a probabilidade de chegada de uma célula aos buffers inativos é menor, porém, como os tamanhos das filas nos buffers ativos não crescerão muito, as células menos prioritárias terão maior chance de serem escalonadas. De acordo com os resultados de simulação apresentados em [47] a taxa de perda de células é menor em estruturas de comutação com buffers nas entradas e disciplina VOQ, quando se usa o algoritmo de escalonamento que dá prioridade ao buffer ativo, do que quando se utiliza o algoritmo IP-PIM, para o caso de pacotes de comprimento variável (pacotes IP).

No capítulo 5 é apresentada uma proposta para um comutador IP com escalonamento distribuído, baseado em prioridades. Essa estrutura de comutação, combina buffers nas entradas e buffers nos pontos de cruzamento, possui facilidades para garantir QoS para os fluxos IP de maior prioridade e é capaz de comutar pacotes IP (de tamanho variável), sem que seja necessário a segmentação do pacote.

2.4.3 Enfileiramento Imparcial usando *Round Robin* com déficit

Roteadores típicos procuram garantir a equidade no uso de seus recursos, promovendo acesso justo para o tráfego nos enlaces de entrada. O uso da disciplina FIFO pode produzir justiça com propriedades exponencialmente ruins. O algoritmo baseado em prioridades, atende aos requisitos de tempo de atraso limitados exigido para fluxos com maior prioridade, porém, quando pacotes com diferentes comprimentos são usados, essa disciplina provoca o uso desequilibrado de um enlace

de saída por um dos fluxos de maior prioridade, da mesma forma que a disciplina FIFO.

Quando há contenção de recursos, é importante que cada recurso seja alocado através de escalonamento imparcial. Entre usuários em contenção, é necessário a existência de algum mecanismo que possa garantir que a alocação "justa" seja estritamente seguida. Porém, não existem tais mecanismos na maioria das redes de computadores e elas são suscetíveis a fontes mal comportadas. Uma fonte "*mal intencionada,"* que transmite a uma taxa descontrolada, pode prender uma fração grande dos *buffers* de um roteador intermediário; isto pode resultar em perda de pacotes para outras fontes que transmitem a taxas mais moderadas. É necessário encontrar soluções que isolem os efeitos negativos provocados pelo provável mal comportamento dos usuários. Em [48] o problema da falta de equidade na alocação de recursos é discutido, as propostas apresentadas em [49]-[53] são analisadas e um algoritmo denominado Enfileiramento Imparcial Eficiente usando *Round Robin* com déficit é proposto.

Denomina-se enfileiramento imparcial (*fair queueing*) [49] à técnica que permite o compartilhamento justo de recursos, a cada fluxo que passa através de um dispositivo de rede. Os esquemas anteriores para enfileiramento imparcial que chegaram perto da imparcialidade perfeita têm implementação onerosa. Conforme foi discutido em [48], o trabalho requerido para processar um pacote nesses esquemas é ***O(log(n))***, onde ***n*** é o número de fluxos ativos, o que é oneroso em altas velocidades. Por outro lado, as aproximações mais baratas para o enfileiramento imparcial que têm sido reportadas na literatura não exibem um comportamento imparcial. O enfileiramento imparcial usando *Round Robin com déficit* (Deficit Round Robin), consegue imparcialidade quase perfeita em termos de vazão, o trabalho requerido para processar um pacote é de apenas *O(1)*, e é simples o suficiente para ser implementado por hardware. O esquema *Round Robin com déficit* é também aplicável em outros problemas de escalonamento onde o serviço não pode ser quebrado em unidades menores e em filas distribuídas.

No Capítulo 5, o algoritmo Enfileiramento Imparcial usando *Round Robin* com déficit será adaptado para ser usado em um comutador IP com escalonamento distribuído.

2.5 Conclusão

Neste capítulo foram discutidas diversas estruturas de comutação e algoritmos de escalonamento, presentes na literatura. Baseado em algumas dessas propostas, ou na combinação delas, serão apresentados nos Capítulos 4 e 5 propostas de novas estruturas e de novos algoritmos que podem ser usados em redes de alta velocidade, sejam elas de comutação de pacotes, ou de comutação de células.

Capítulo 3

3 Propostas de Estruturas de Comutadores ATM para Redes de Alta Velocidade

3.1 Introdução

Neste capítulo são apresentadas três estruturas de comutador para aplicações em redes de comunicação de alta velocidade que usam pacotes de comprimento fixo, como as redes ATM. As três estruturas são do tipo *crossbar*, e utilizam *buffers* nas entradas e nos pontos de cruzamentos para armazenar células. Os *buffers* são divididos em classes de serviço e o encaminhamento de células é feito de acordo com uma prioridade de serviço. São discutidos os esquemas de transferência de células e o algoritmo de encaminhamento de células. As três estruturas são comparadas e um modelo de análise para a estrutura que se mostrou mais eficiente para priorizar o tráfego com maior exigência de QoS é desenvolvido. Finalmente, é feita a análise de desempenho da estrutura modelada.

3.2 Estrutura 1: Comutador com *Buffers* na entrada e nos pontos de cruzamentos discriminados em classes de serviço.

O comutador proposto possui uma estrutura do tipo *crossbar* com N entradas e N saídas como mostrado na Fig. 3–1. Cada entrada possui um discriminador de células e um conjunto de cinco *buffers*, um para cada uma das classes de serviço CBR, rtVBR, nrtVBR, ABR e UBR. As células que chegam a uma porta de entrada são discriminadas quanto ao tipo de serviço e armazenadas no *buffer* apropriado. Em cada ponto de cruzamento é providenciado também um conjunto de cinco *buffers* mas neste caso cada *buffer* pode armazenar somente uma célula.

O processo de comutação é feito durante um slot de tempo (tempo de processamento de uma célula) e é dividido em duas fases. Na primeira fase as células são encaminhadas aos buffers dos pontos de cruzamentos. Cada entrada, sob a monitoração de uma unidade de controle e um filtro, encaminha uma célula a um buffer de cruzamento. Este encaminhamento obedece a uma disciplina que privilegia as classes de serviço de maior prioridade, e dentro de uma mesma classe de serviço,

obedece à disciplina FIFO. Na segunda fase é executado o algoritmo de resolução de encaminhamento, que escolhe a célula que será enviada por uma determinada saída, se existir mais de uma célula esperando nos *buffers* de cruzamento. Este algoritmo favorece as células de classes de serviços com maior prioridade e em casos da mesma classe de serviço é utilizado o esquema *round robin* para o encaminhamento.

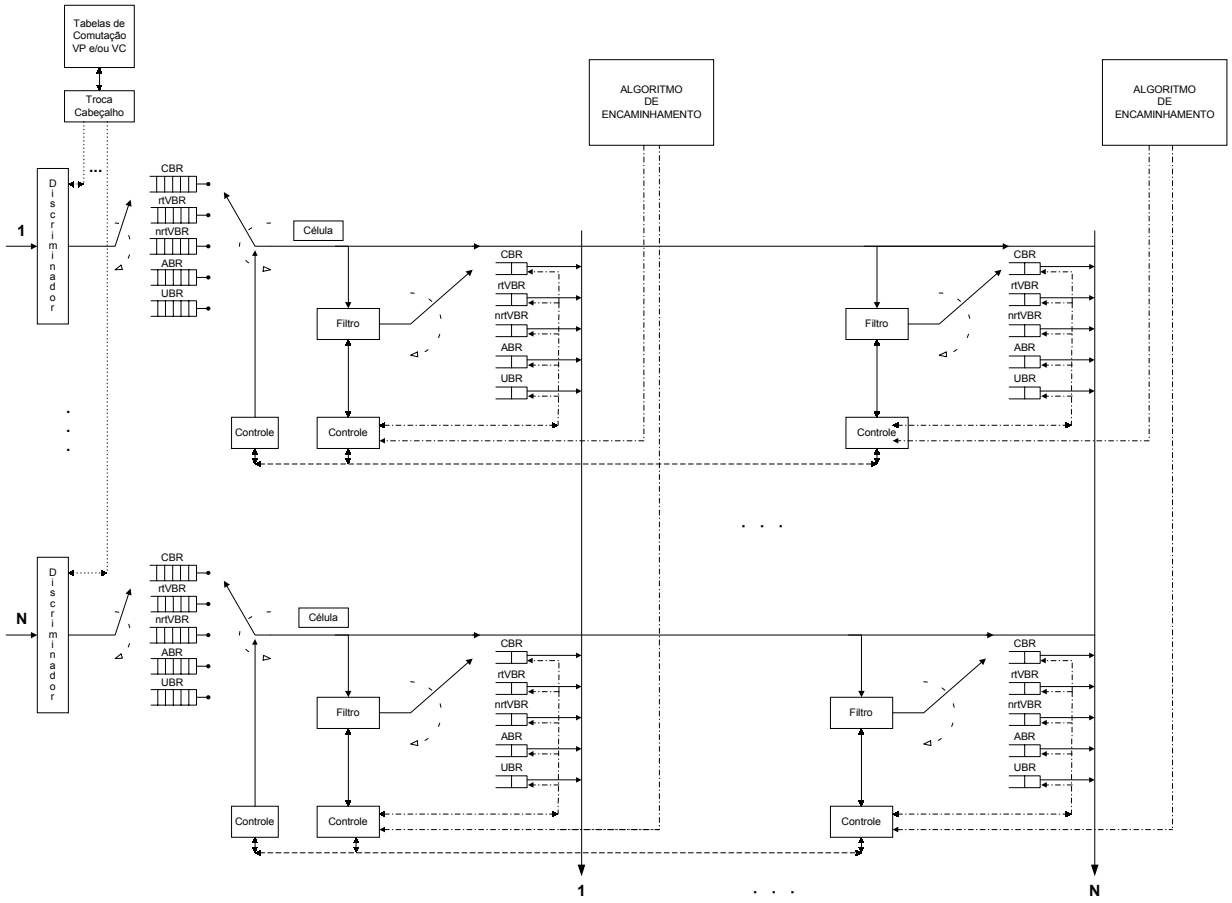


Figura 3-1: Estrutura 1 *Buffers* na entrada e nos pontos de cruzamentos, com armazenamento de uma célula em cada conjunto de *buffers* dos pontos de cruzamentos.

3.2.1 Esquema de transferência de células ao buffers dos pontos de cruzamentos

Na Fig. 3–2 está ilustrado o funcionamento do esquema que seleciona as células para encaminhamento aos *buffers* dos pontos de cruzamento.

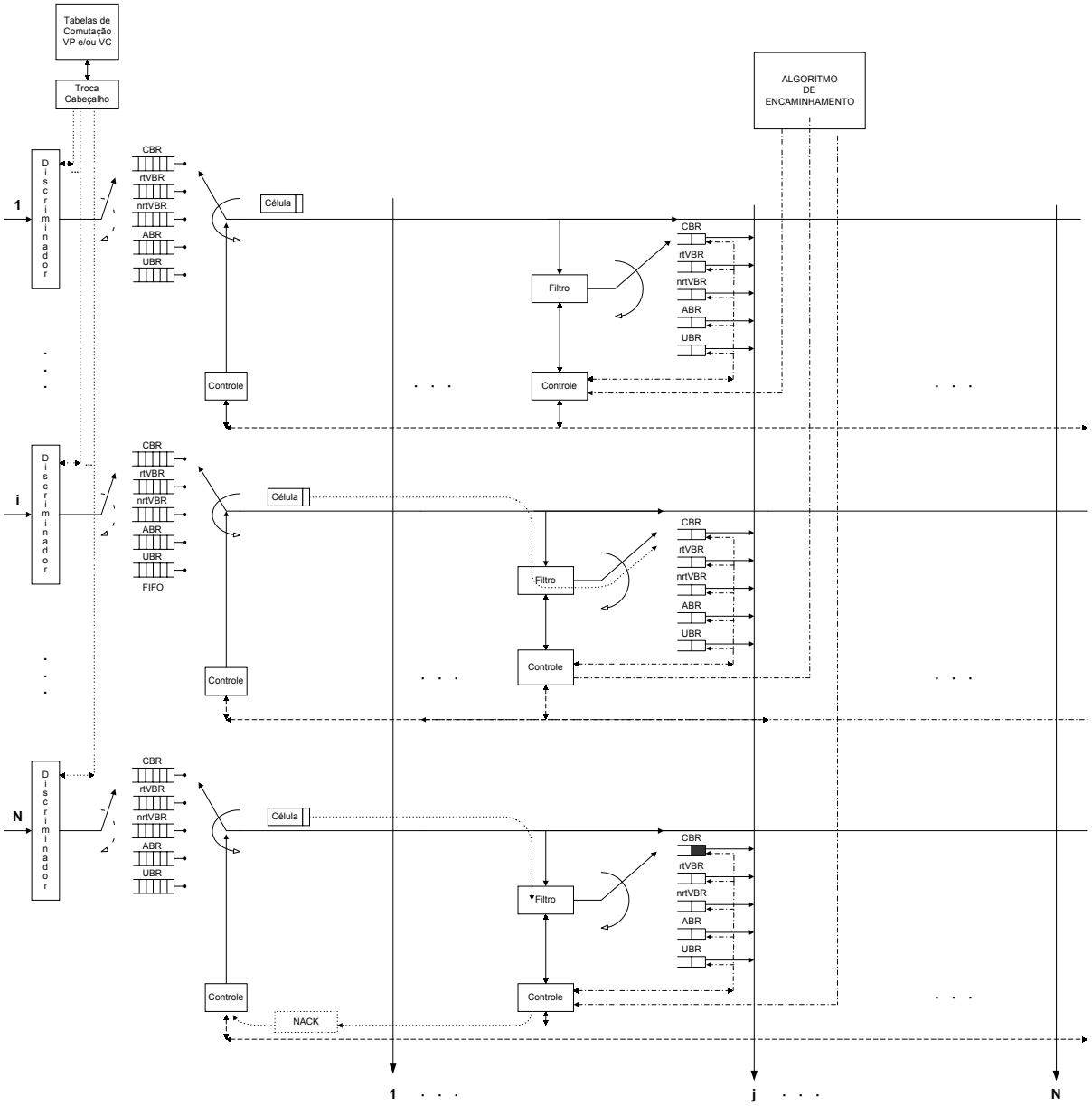


Figura 3-2: Esquema de encaminhamento de células aos buffers dos pontos de cruzamentos - Fase 1.

O conjunto de *buffers* de uma dada entrada, por exemplo da *entrada i* é examinado, começando pelo *buffer* da classe de serviço CBR. Se existir célula esperando, por exemplo na classe de serviço CBR, ela é transmitida pelo barramento. Uma cópia desta célula permanece no *buffer* de entrada. Os filtros selecionam as células que são direcionadas às respectivas saídas. Se o número da *porta j* é reconhecido e o *buffer* CBR do cruzamento *(i,j)* estiver vazio, a célula é armazenada no *buffer* da classe de serviço CBR do ponto de cruzamento *(i,j)*. Se o referido *buffer* estiver ocupado, a célula é descartada e um sinal de controle (*NACK* - um bit apenas) é enviado à entrada para que

a cópia seja preservada. Se o sinal de controle não é recebido pela *entrada i*, é admitido que a transferência de célula foi bem sucedida e a cópia da célula é apagada.

Se o *buffer* CBR da *entrada i* estiver vazio, o *buffer* rtVBR da *entrada i* é examinado. O procedimento é repetido até que uma célula seja armazenada em algum *buffer* dos pontos de cruzamentos ou até que o *buffer* UBR da *entrada i* seja examinado.

Este algoritmo é executado paralelamente em todas as entradas, ou seja, cada entrada encaminha uma célula para algum *buffer* dos pontos de cruzamentos. Os bits de cada célula podem ser encaminhados também em paralelo, possibilitando assim uma transferência em alta velocidade.

3.2.2 Algoritmo de encaminhamento das células

Na Fig.3−3 está ilustrado a segunda fase da comutação de célula da estrutura proposta.

Cada unidade de controle (UC) dos pontos de cruzamentos (1...N , j) verifica o estado do *buffer* CBR (1...N , j) e caso existam células esperando, um sinal de REQUEST (apenas um bit) é enviado para a Unidade de Controle de Requisição CBR (UCR-CBR) usando uma linha reservada (cada unidade de controle tem uma linha separada, reservada para propósitos de encaminhamento). Se existirem duas ou mais requisições a UCR-CBR seleciona uma célula de acordo com a disciplina *round robin* e envia um sinal de ACK (apenas um bit) para a $UC_{i,j}$ escolhida através da linha reservada, e a informação END para as outras UC, sinalizando que a célula que será transmitida no próximo slot já foi selecionada. (As UCR-CBR, UCR-rtVBR, UCR-nrtVBR, UCR-ABR e UCR-UBR possuem cada uma, linhas separadas para informações ACK e END para cada UC dos pontos de cruzamentos). Se a UCR-CBR não receber requisições, ela envia uma informação NEXT (apenas um bit) para as UCs e para a UCR-rtVBR. Agora as $UC_{1..N,j}$ examinam os *buffers* rtVBR e o mesmo procedimento é executado até que uma célula seja escolhida, ou até que os *buffers* UBR sejam examinados. Todas as portas de saída executam simultaneamente este algoritmo para selecionar a célula que será encaminhada.

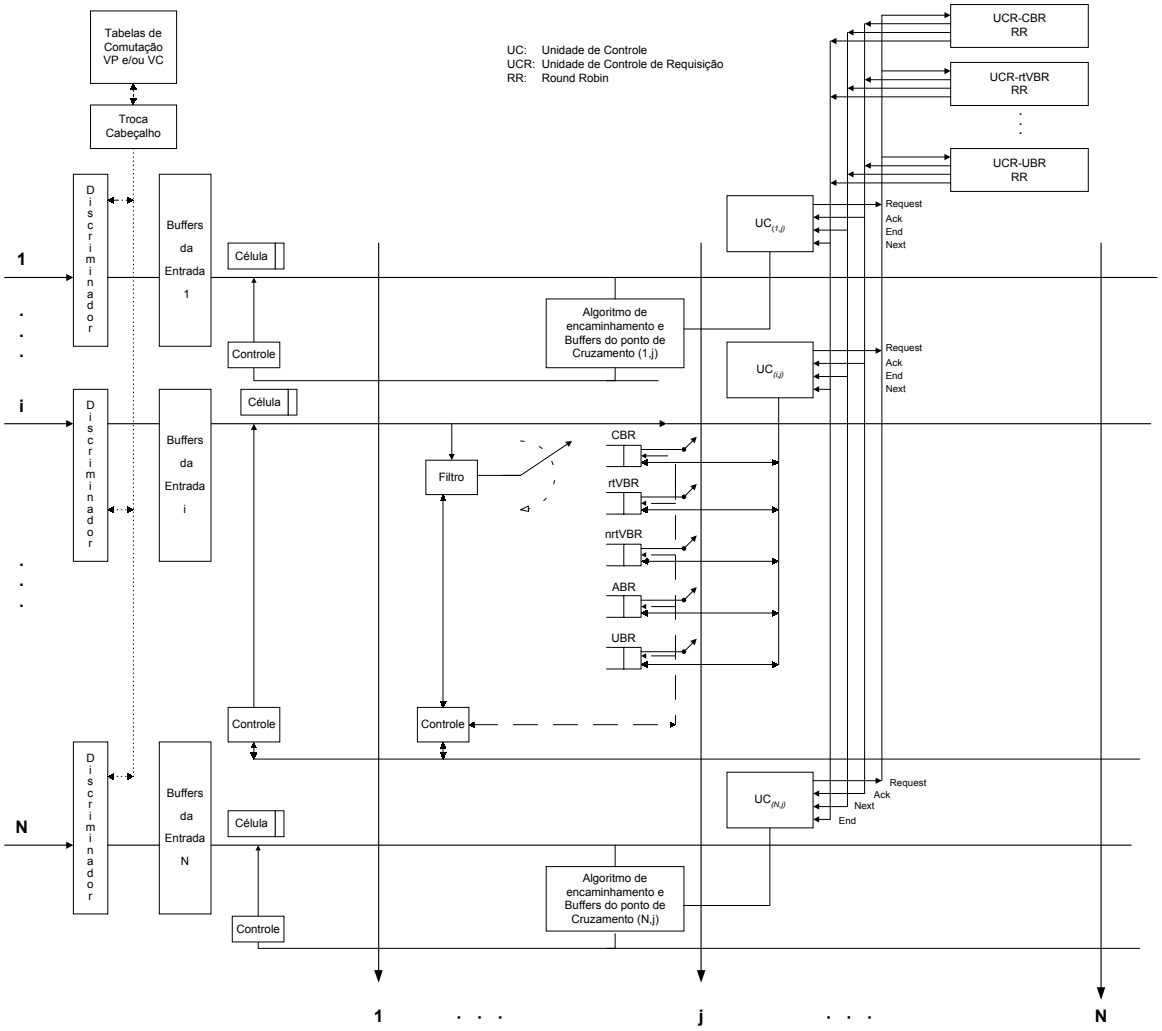


Figura 3-3: Algoritmo de encaminhamento de células - Fase 2.

3.2.3 Análise da vazão: Estudo de caso

A Fig. 3–4 mostra uma situação dos *buffers* de entrada de um comutador de três entradas e três saídas usado para avaliar o comportamento do esquema proposto. Cada entrada possui, então, um conjunto de cinco *buffers*, um para cada serviço CBR, rtVBR, nrtVBR, ABR, UBR. Os números representados nos *buffers* indicam o número da saída para a qual cada célula deve ser enviada, por exemplo: analisando o *Buffer* CBR da entrada E_1, da direita para a esquerda, verifica-se que o mesmo possui três células que devem ser encaminhadas para a Saída 2, duas células para a Saída 1, duas células para a Saída 3 e finalmente uma célula para a Saída 1.

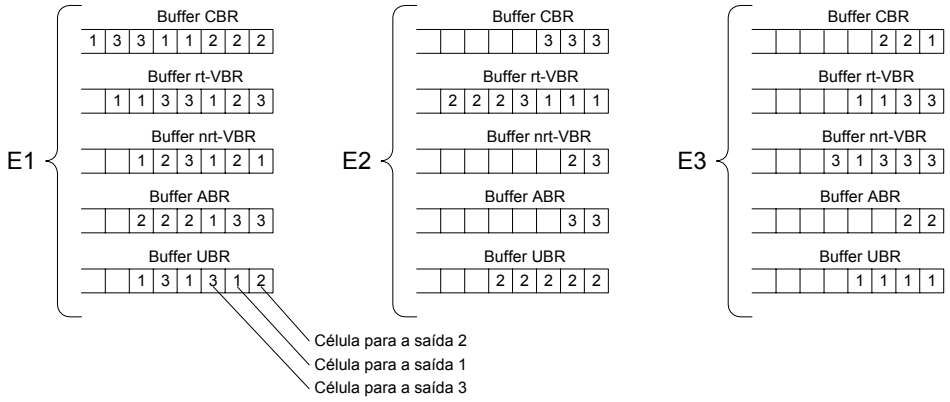


Figura 3-4: *Um cenário de tráfego nos buffers de entrada de um comutador usado para estudo de caso.*

A Tabela 3–1 apresenta as células encaminhadas a cada saída quando o algoritmo proposto é usado. Pela Tabela 3–1 verifica-se que saídas ficam ociosas, embora existam células endereçadas a elas armazenadas nos buffers de entrada. Ocorre o fenômeno conhecido como HOLB, onde a célula posicionada na saída do *buffer* impede que as demais sejam transmitidas. Por exemplo, embora existem células no *buffer* CBR da entrada E1 endereçadas à saída S1, e S1 esteja ociosa nos instantes de tempo T_2 e T_3 (Tabela 3–1), estas células não são enviadas enquanto as células endereçadas à saída S2, que estão a sua frente no *buffer*, não forem enviadas.

O esquema proposto na Estrutura 1 é baseado em atendimento de serviço do tipo FIFO em cada classe de serviço. Para evitar o problema de bloqueio (HOLB) na saída do *buffer* e melhorar o desempenho do comutador, pode-se encaminhar na fase 1 as células em paralelo.

Tabela 3-1: *Células da Fig. 3–4 transmitidas pelas saídas S1, S2 e S3 do comutador de acordo com o esquema proposto.*

	S1	**S2**	**S3**
T_1	CBR-E3	CBR-E1	CBR-E2
T_2	-	CBR-E3	CBR-E2
T_3	-	CBR-E1	CBR-E2
T_4	rt-VBR-E2	CBR-E3	-
T_5	rt-VBR-E2	CBR-E1	rt-VBR-E3
T_6	CBR-E1	-	rt-VBR-E3
T_7	CBR-E1	-	-
T_8	rt-VBR-E3	-	CBR-E1
T_9	rt-VBR-E2	-	CBR-E1
T_{10}	CBR-E1	-	rt-VBR-E2
T_{11}	rt-VBR-E3	rt-VBR-E2	rt-VBR-E1
T_{12}	-	rt-VBR-E1	nrt-VBR-E3
T_{13}	rt-VBR-E1	rt-VBR-E2	nrt-VBR-E3
T_{14}	-	rt-VBR-E2	rt-VBR-E1
T_{15}	-	-	rt-VBR-E1
T_{16}	rt-VBR-E1	-	nrt-VBR-E2
T_{17}	rt-VBR-E1	nrt-VBR-E2	nrt-VBR-E3
T_{18}	nrt-VBR-E1	-	ABR-E2
T_{19}	nrt-VBR-E3	nrt-VBR-E1	ABR-E2
T_{20}	nrt-VBR-E1	UBR-E2	nrt-VBR-E3
T_{21}	-	ABR-E3	nrt-VBR-E1
T_{22}	-	nrt-VBR-E1	-
T_{23}	nrt-VBR-E1	ABR-E3	-
T_{24}	UBR-E3	UBR-E2	ABR-E1
T_{25}	UBR-E3	UBR-E2	ABR-E1
T_{26}	ABR-E1	UBR-E2	-
T_{27}	UBR-E3	ABR-E1	-
T_{28}	UBR-E3	ABR-E1	-
T_{29}	-	ABR-E1	-
T_{30}	-	UBR-E1	-
T_{31}	UBR-E1	UBR-E2	-
T_{32}	-	-	UBR-E1
T_{33}	UBR-E1	-	-
T_{34}	-	-	UBR-E1
T_{35}	UBR-E1	-	-

3.3 Estrutura 2: Estruturas com atendimentos paralelos

Na Fig. 3–5 está mostrado o comutador da Fig. 3–1 modificado para melhorar o seu desempenho. Em cada entrada são introduzidos barramentos paralelos que possibilitam a transmissão de uma célula a cada *buffer* dos pontos de cruzamento para cada tipo de serviço.

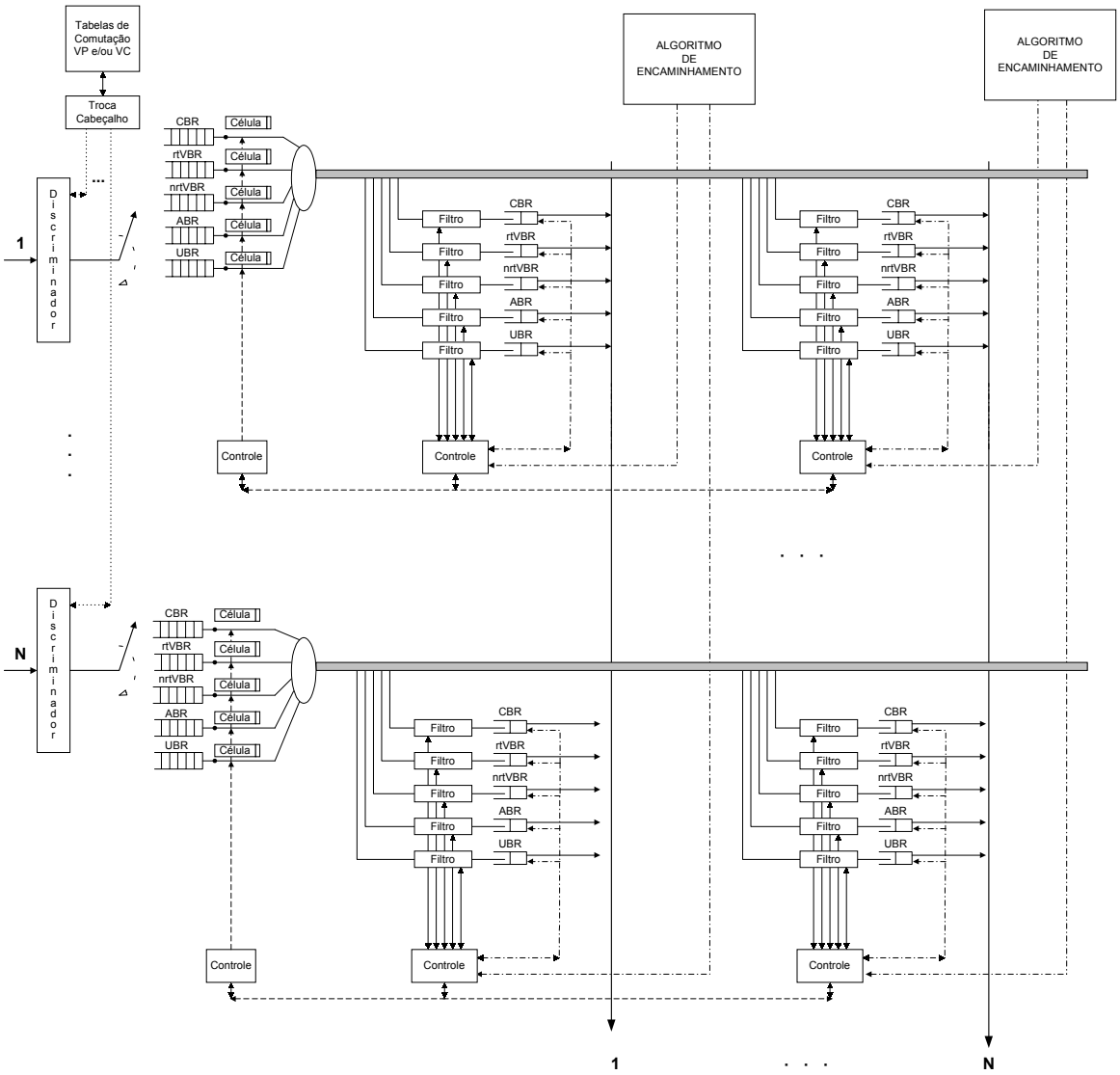


Figura 3-5: *Estrutura 2: Buffers na entrada e nos pontos de cruzamentos, com armazenamento de uma célula em cada buffer de cada tipo de serviço*

O esquema de transferência de células aos *buffers* dos pontos de cruzamentos é explicado a seguir.

O conjunto de *buffers* de uma dada entrada é examinado. Se existirem células esperando elas são transmitidas pelo barramento. Por exemplo, se existir uma célula na classe de serviço CBR da *entrada i*, ela é transmitida pelo barramento; paralelamente, se existir uma célula na classe de serviço rtVBR da *entrada i*, ela é transmitida pelo barramento e o mesmo se repete para cada uma das classes de serviço nrtVBR, ABR e UBR. Uma cópia destas células permanece no *buffer* de entrada. Os filtros selecionam as células que são direcionadas às respectivas saídas. Se o número da *porta j* é reconhecido e o buffer CBR do cruzamento (i,j) estiver vazio, a célula é armazenada no *buffer* da classe de serviço CBR do ponto de cruzamento (i,j). Se o referido *buffer* estiver ocupado, a célula é descartada e um sinal de controle (*NACK* - um bit apenas) é enviado à entrada para que a cópia seja preservada. Se o sinal de controle não é recebido pela *entrada i*, é admitido que a transferência de célula foi bem sucedida e a cópia da célula é apagada.

Somente essa fase 1 de transferência de células é diferente nesta estrutura. A fase 2 de encaminhamento de células é a mesma utilizada na estrutura anterior, ou seja é baseada em atendimento do tipo *round robin*.

A Tabela 3.2 mostra as células transmitidas pelas saídas S1, S2 e S3 para a mesma situação de ocupação de *buffers* de entrada mostrada na Fig. 3.4 e usando o esquema proposto na Estrutura 2.

Pela comparação das Tabelas 3.1 e 3.2 pode-se observar que a segunda estrutura proposta melhora a vazão do comutador, entretanto esta melhoria de desempenho não prioriza em algumas situações o tráfego com maior exigência de qualidade de serviço.

Na Fig. 3–6 é mostrada uma nova proposta, Estrutura 3, para o comutador. Nesta estrutura o comutador possui um barramento que liga cada entrada aos *buffers* de cruzamentos. Existe um caminho dedicado ligando cada um dos *buffers* de cada uma das classes de serviços de entrada aos *buffers* da classe de serviço equivalente a cada um dos pontos de cruzamentos.

Nesta estrutura até N células de uma mesma classe, em um enlace de entrada, podem ser encaminhados aos *buffers* de pontos de cruzamentos. O esquema de transferência de células opera do seguinte modo.

Tabela 3-2: *Células transmitidas pelas saídas S1, S2 e S3 para o Comutador da Estrutura 2.*

	S1	**S2**	**S3**
t_1	CBR-E3	CBR-E1	CBR-E2
t_2	rt-VBR-E2	CBR-E3	CBR-E2
t_3	rt-VBR-E2	CBR-E1	CBR-E2
t_4	rt-VBR-E2	CBR-E3	rt-VBR-E1
t_5	nrt-VBR-E1	CBR-E1	rt-VBR-E2
t_6	CBR-E1	rt-VBR-E1	rt-VBR-E3
t_7	CBR-E1	rt-VBR-E2	rt-VBR-E3
t_8	rt-VBR-E3	rt-VBR-E2	CBR-E1
t_9	rt-VBR-E1	rt-VBR-E2	CBR-E1
t_{10}	CBR-E1	nrt-VBR-E1	rt-VBR-E1
t_{11}	rt-VBR-E3	ABR-E3	rt-VBR-E1
t_{12}	rt-VBR-E1	ABR-E3	nrt-VBR-E1
t_{13}	rt-VBR-E1	nrt-VBR-E2	nrt-VBR-E2
t_{14}	nrt-VBR-E1	UBR-E1	nrt-VBR-E3
t_{15}	UBR-E1	UBR-E2	nrt-VBR-E1
t_{16}	UBR-E3	nrt-VBR-E1	nrt-VBR-E3
t_{17}	nrt-VBR-E3	UBR-E2	ABR-E1
t_{18}	nrt-VBR-E1	UBR-E2	nrt-VBR-E3
t_{19}	UBR-E1	UBR-E2	ABR-E2
t_{20}	UBR-E1	UBR-E2	ABR-E1
t_{21}	ABR-E1	-	ABR-E2
t_{22}	UBR-E3	ABR-E1	UBR-E1
t_{23}	UBR-E1	ABR-E1	-
t_{24}	-	ABR-E1	UBR-E1
t_{25}	UBR-E1	-	-

O conjunto de *buffers* CBR é examinado e determina-se para quais saídas existem células esperando. A seguir as células são transmitidas pelo barramento, uma para cada saída. Por exemplo, o *buffer* da classe de serviço CBR da *entrada i* é examinado e verifica-se que existem células esperando para as *saídas x*, *y* e *z*, assim três células são enviadas pelos caminhos dedicados, a primeira é armazenada no *buffer* de cruzamento do serviço CBR(*i*,*x*); a segunda no *buffer* CBR(*i*,*y*) e a terceira no *buffer* CBR(*i*,*z*).

Paralelamente, os buffers das outras classes de serviço são examinados e o mesmo esquema é executado simultaneamente em cada uma delas. As células de uma mesma classe de serviço endereçadas a uma mesma saída são transmitidas de acordo

com a disciplina FIFO, porém, dentro da mesma classe de serviço as células endereçadas às saídas mais ociosas podem ser encaminhadas primeiro. A fase 2 de encaminhamento de células é a mesma das Estruturas 1 e 2.

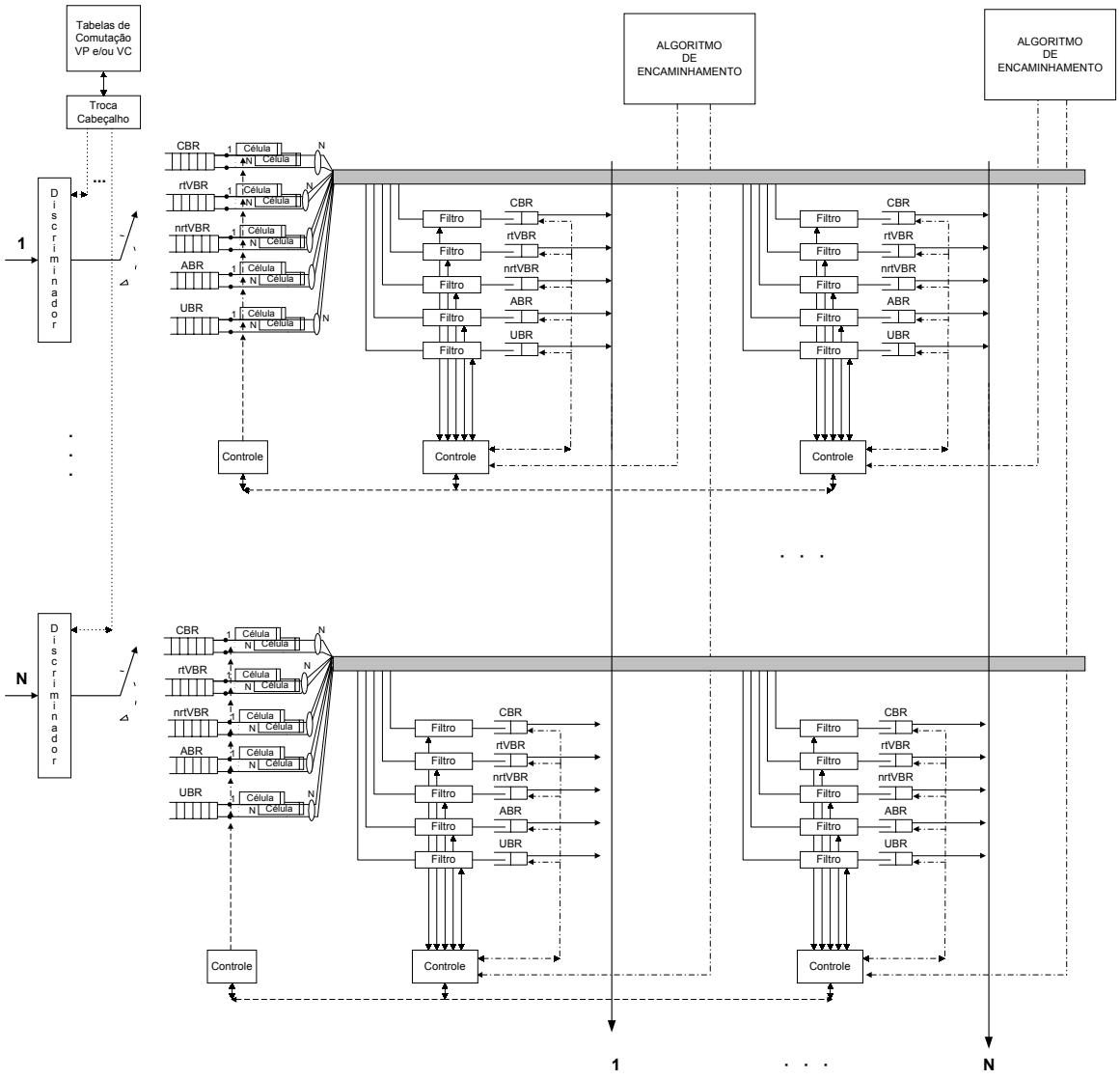


Figura 3-6: *Estrutura 3: Buffers na entrada e nos pontos de cruzamentos, com armazenamento de uma célula em cada buffer de cada tipo de serviço, e leitura paralela dos buffers de entrada.*

A Tabela 3–3 mostra as células transmitidas pelas saídas S1, S2 e S3 para o mesmo cenário proposto na Fig. 3.4 e usando o esquema proposto na Estrutura 3.

Comparando as Tabelas 3–2 e 3–3 observa-se que além da melhoria de desempenho a Estrutura 3 prioriza o tráfego com maior exigência de qualidade de serviço.

Tabela 3-3: *Células transmitidas pelas saídas S1, S2 e S3 para o Comutador da Estrutura 3.*

	S1	**S2**	**S3**
t_1	CBR-E1	CBR-E1	CBR-E1
t_2	CBR-E3	CBR-E3	CBR-E2
t_3	CBR-E1	CBR-E1	CBR-E1
t_4	CBR-E1	CBR-E3	CBR-E2
t_5	rt-VBR-E1	CBR-E1	CBR-E2
t_6	rt-VBR-E2	rt-VBR-E1	rt-VBR-E1
t_7	rt-VBR-E3	rt-VBR-E2	rt-VBR-E2
t_8	rt-VBR-E1	rt-VBR-E2	rt-VBR-E3
t_9	rt-VBR-E2	rt-VBR-E2	rt-VBR-E1
t_{10}	rt-VBR-E3	nrt-VBR-E1	rt-VBR-E3
t_{11}	rt-VBR-E1	nrt-VBR-E2	rt-VBR-E1
t_{12}	rt-VBR-E2	nrt-VBR-E1	nrt-VBR-E1
t_{13}	nrt-VBR-E1	ABR-E1	nrt-VBR-E2
t_{14}	nrt-VBR-E3	ABR-E3	nrt-VBR-E3
t_{15}	nrt-VBR-E1	ABR-E1	nrt-VBR-E3
t_{16}	nrt-VBR-E1	ABR-E3	nrt-VBR-E3
t_{17}	ABR-E1	ABR-E1	nrt-VBR-E3
t_{18}	UBR-E1	UBR-E1	ABR-E1
t_{19}	UBR-E3	UBR-E2	ABR-E2
t_{20}	UBR-E1	UBR-E2	ABR-E1
t_{21}	UBR-E3	UBR-E2	ABR-E2
t_{22}	UBR-E1	UBR-E2	UBR-E1
t_{23}	UBR-E3	UBR-E2	UBR-E1
t_{24}	UBR-E3	-	-

3.4 Comparação das estruturas.

3.4.1 Comparação de hardware

A Estrutura 1 possui uma linha paralela ligando os *buffers* da entrada aos pontos de cruzamentos e um filtro em cada ponto de cruzamento, totalizando um total de **N** linhas e $\mathbf{N^2}$ filtros onde **N** é o número de entradas ou saídas. Já a Estrutura 2 possui

uma linha paralela ligando os *buffers* de cada classe de serviço aos pontos de cruzamentos e um filtro para cada classe de serviço em cada cruzamento, sendo portanto necessário $\mathbf{M_{CS}xN}$ linhas e $\mathbf{M_{CS}xN^2}$ filtros no total, onde $\mathbf{M_{CS}}$ é o número de classes de serviço. Finalmente, a Estrutura 3 possui **N** linhas paralelas que ligam os *buffers* de entrada de cada classe de serviço aos pontos de cruzamentos e o mesmo número de filtros da Estrutura 2, totalizando $\mathbf{M_{CS}xN^2}$ linhas paralelas e $\mathbf{M_{CS}xN^2}$ filtros.

3.4.2 Gerenciamento de *buffer*

As Estruturas 1 e 2 utilizam o esquema FIFO para transmissão das células de uma mesma classe de serviço o que facilita o gerenciamento do *buffer*.

A Estrutura 3 não utiliza a disciplina FIFO para atendimento das células de uma mesma classe de serviço. O *buffer* neste caso deve ser uma memória RAM e o gerenciamento de memória é mais complexo, pois é necessário selecionar quais as células devem ser encaminhadas.

3.4.3 Comparação do número de *buffers* com relação a uma estrutura com *buffer* na entrada e na saída

A capacidade total de armazenamento de células C nas estruturas com *buffers* nas entradas e nos pontos de cruzamentos é dada por:

$$C = K_E \times N + M_{CS} \times N^2$$

onde,

$\mathbf{K_E}$ = Capacidade de armazenamento de células nos *buffers* de cada entrada

$$K_E = \sum_{i=1}^{M_{CS}} K_i \qquad \mathbf{K_i} = \text{Capacidade do buffer da classe de serviço i.}$$

N = Número de entradas ou de saídas

$\mathbf{M_{CS}}$ = Número de Classe de Serviços.

Por outro lado a capacidade total de armazenamento de células C nas estruturas com *buffers* nas entradas e nas saídas é dado por:

$$C = K_E \times N + K_S \times N$$

onde,

$\mathbf{K_E}$ = Capacidade de armazenamento de células nos buffers de cada entrada

$\mathbf{K_S}$ = Capacidade de armazenamento de células nos buffers de cada saída

N = Número de entradas ou de saídas

Para que a capacidade de armazenamento das duas estruturas sejam iguais é necessário que:

$$N = \frac{K_S}{M_{CS}}$$

Por exemplo, se o número de entradas ou saídas de um comutador (**N**) for igual a 32, e o número de classes de serviço ($\mathbf{M_{CS}}$) for 5, então para que as duas estruturas tenham a mesma capacidade de armazenamento interno de células a capacidade de armazenamento dos *buffers* de cada saída ($\mathbf{K_S}$) deverá ser igual a 160 células, considerando que a capacidade de armazenamento dos *buffers* de cada entrada ($\mathbf{K_E}$) são iguais nas duas estruturas.

A estrutura *crossbar* requer maior capacidade de armazenamento de células sempre que o número de entradas (**N**) for maior que a razão $\mathbf{K_S/M_{CS}}$; entretanto o algoritmo necessário para encaminhamento de células é mais simples.

3.4.4 Vantagens de cada estrutura

A Estrutura 3 permite transferência de até $\mathbf{M_{CS}xN^2}$ células para os pontos de cruzamentos em uma operação, conseguindo assim a melhor vazão.

A Estrutura 2 permite transferência de até $\mathbf{M_{CS}xN}$ células em uma operação conseguindo vazão total próxima à conseguida pela Estrutura 3, entretanto em algumas situações não prioriza as classes de serviço com maior prioridade.

A Estrutura 1 permite transferência de até **N** células em uma operação e consegue a menor vazão das três estruturas, porém é a mais simples.

A estrutura com buffers nas entradas e nas saídas permite encaminhar até $\mathbf{K_SxN}$ células aos buffers das saídas. Sempre que **N** for maior que $\mathbf{K_S/M_{CS}}$ as estruturas crossbar (Estruturas 1, 2 e 3) necessitam maior capacidade de armazenamento, entretanto, elas utilizam um algoritmo de encaminhamento de células mais simples.

3.5 Modelo de Análise de desempenho para o Comutador *Crossbar* com atendimentos paralelos e barramento dedicado (Estrutura 3)

Na Tabela 3–3 pode-se observar que cada uma das portas de saída do comutador com atendimentos paralelos e barramentos dedicados (Estrutura 3) opera como uma fila de prioridades, onde as classes de serviços de maior prioridade são atendidas primeiro, e dentro de uma mesma classe de serviço, as células de uma mesma entrada são atendidas de acordo com a disciplina FIFO.

Para a análise será considerado um modelo agregado de fila, com um buffer para cada classe de serviço, e um único servidor representando uma porta de saída como mostrado na Fig. 3–7. Neste modelo, a cada *slot*, o servidor verifica primeiro se existem células no *buffer* CBR que é o de mais alta prioridade; após transmitir todas as células CBR, ele passa a examinar o *buffer* rtVBR e assim por diante.

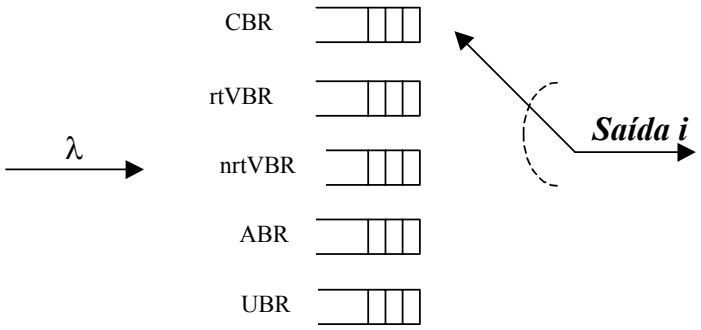


Figura 3-7: *Modelo agregado de fila em uma saída i*

São adotadas as seguintes suposições: O número de portas de entrada e de saída é **N** e a chegada das células na entrada obedece ao processo de Bernoulli idêntico e independente. A probabilidade de chegada de uma célula em um slot é *p*. A porcentagem de tráfego da classe de serviço *i* é S_i e $\sum_i S_i = 1$.

A probabilidade de uma célula em uma porta de entrada ser encaminhada para uma saída particular é **1/N**, então o tráfego em cada buffer agregado é $NpS_i\frac{1}{N}$. Assumiu-se também que existem ***r*** classes de serviços e a classe de serviço genérica *h* pode assumir os valores ***h = 1, 2, 3, ... r***

onde *r* é a classe de serviço de menor prioridade. As células de mesma prioridade são servidas de acordo com a disciplina FIFO.

Assumiu-se que os slots de entrada e de saída não são sincronizados, existe então um tempo residual de serviço quando chega a célula alvo.

Usando raciocínio similar ao desenvolvido em [54] para sistemas de filas com prioridades e disciplina de serviço não *preemptivo*, o tempo médio de espera $E\{W_h\}$ pode ser escrito como:

$$E\{W_h\} = E\{T_0\} + \sum_{k=1}^{h} E\{T_k\} + \sum_{k=1}^{h-1} E\{T'_k\} \qquad (3\text{–}1)$$

Onde,

- $\boldsymbol{E\{T_0\}}$ é o tempo médio (tempo residual) para terminar a transmissão de uma célula, quando chega a célula alvo;
- $\boldsymbol{E\{T_k\}}$ é o tempo médio para servir todas as células de classes de prioridade iguais e mais altas ***(h, h-1, ..., r)*** que esperam na fila quando chega a célula alvo;
- $\boldsymbol{E\{T'_k\}}$ é o tempo médio para servir todas as células de classes de prioridade mais altas ***(h-1, ..., r)*** que chegaram durante o período $\boldsymbol{E\{W_h\}}$ segundos e que serão servidas antes da célula alvo.

$\mathbf{E\{T_k\}}$ é dado por:

$$\boldsymbol{E\{T_k\} = E\{m_k\}.Tslot} \qquad (3\text{–}2)$$

Onde,

- $\boldsymbol{E\{m_k\}}$ é o número médio de células esperando no *buffer*, com prioridade maior à célula alvo ou prioridade igual mas que chegaram antes; e
- ***Tslot*** é o tempo necessário para transmitir uma célula.

Da fórmula de Little:

$$E\{m_k\} = \frac{S_k p}{T_{slot}} E\{W_k\} \qquad (3\text{–}3)$$

Onde $E\{W_k\}$ é o tempo médio de espera das células de prioridade *k*.

Então,

$$E\{T_k\} = S_k \cdot p \cdot E\{W_k\} \qquad \text{e} \qquad E\{T'_k\} = S_k \cdot p \cdot E\{W_k\} \qquad (3\text{–}4)$$

Portanto a Eq.3–1 pode ser resolvida para $E\{W_1\}$, depois para $E\{W_2\}$ e genericamente para $E\{W_h\}$ obtendo-se:

$$E\{W_h\} = \frac{E\{T_0\}}{(1 - p\sigma_{h-1})(1 - p\sigma_h)} \tag{3.5}$$

Onde,

$$\sigma_h = \sum_{k=1}^{h} S_k \ , \ \sigma_0 = 0 \ , \ \sigma_r = 1$$

Para slots de tamanho fixo, $E\{T_0\}$ é dado por:

$$E\{T_0\} = \sum_{k=1}^{r} \frac{N-1}{N} \frac{p}{2} S_k T_{slot} = \frac{(N-1)p}{2N} T_{slot} \tag{3.6}$$

Então,

$$E\{W_h\} = \frac{(N-1)p}{2N(1 - p\sigma_{h-1})(1 - p\sigma_h)} T_{slot} \tag{3.7}$$

Em geral as células são servidas em slots seguintes ao que chegaram. Assim, pode-se escrever:

$$E\{W_h\} = T_{slot}\left[1 + \frac{(N-1)p}{2N(1 - p\sigma_{h-1})(1 - p\sigma_h)}\right] \tag{3.8}$$

3.5.1 Análise de Desempenho

O desempenho do comutador, considerando o tempo de atraso devido à espera das células nos *buffers,* obtido pela Eq.3–8, é analisado pela observação das Fig.3–8, Fig.3–9 e Fig.3–10. Nestas figuras são mostradas as diferentes situações de distribuição de carga, entre as classes de serviço, para um comutador com 16 entradas e 16 saídas.

Na Fig. 3–8 considerou-se porcentagens de carga de S_1=40%, S_2=20%, S_3=20%, S_4=10% e S_5=10% para CBR, rtVBR, nrtVBR, ABR e UBR respectivamente. Observa-se pela Fig. 3–8(a) que, para cargas abaixo de 90%, somente o serviço UBR tem um tempo médio de espera longo. Todas as outras classes de serviço possuem tempo de espera razoáveis. Para os serviços de prioridades mais altas, CBR e rtVBR, (Fig. 3–8 (b)) o tempo de espera é menor que o tempo de 3 slots para qualquer situação de carga.

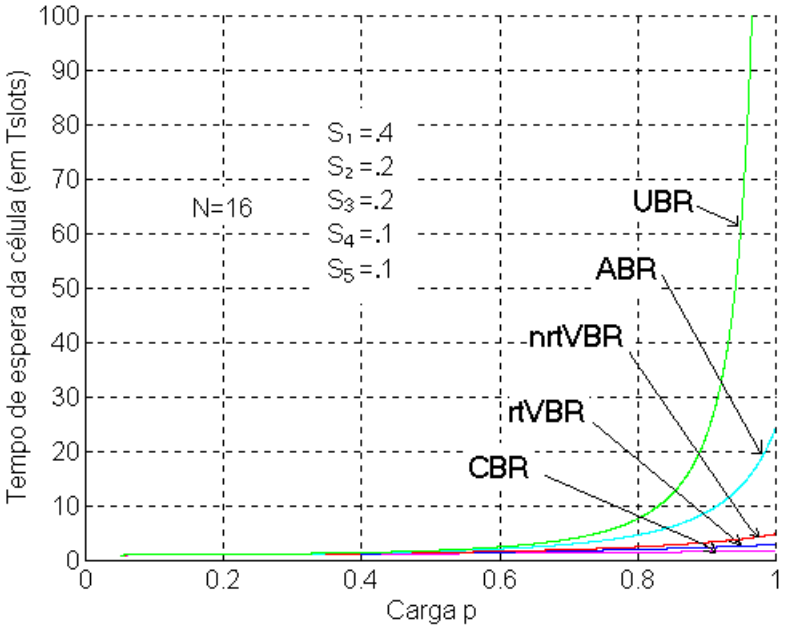


Figura 3.8 (a)

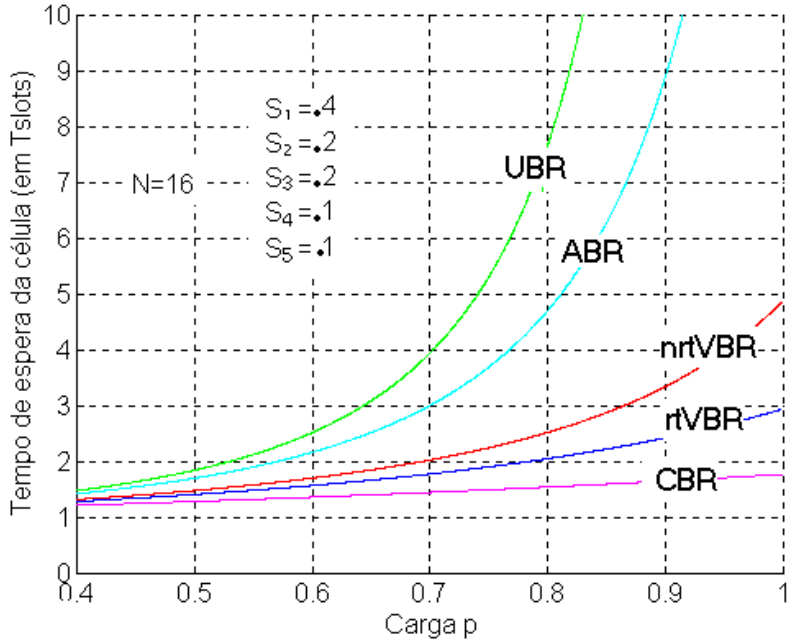


Figura 3.8 (b)

Figura 3-8: *Desempenho de um comutador 16x16 com 80% do tráfego nas classes de serviço com maior exigência de qualidade de serviço (CBR, rtVBR e nrtVBR).*

No exemplo mostrado pela Fig. 3–9 (a) e (b) considerou-se que o tráfego da rede é igual para todas as classes de serviço. Neste caso os tempos de atraso das células de maior prioridade permanecem pequenos enquanto que os atrasos para as células de menor prioridade diminuem.

Já, na Fig. 3–10 (a) e (b) considerou-se que 60% da carga na rede é de tráfego de baixa prioridade, ABR e UBR. Nesse caso, os tempos de atrasos para as classes CBR, nrtVBR, rtVBR e ABR são baixos, e o tráfego UBR, a classe de menor prioridade, tem atraso considerável apenas para situações de carga acima de 95%.

Considerando a análise de desempenho feita, o comutador ATM com buffers nas entradas e nos pontos de cruzamentos, com atendimento paralelo e barramentos dedicados é adequado e atende facilmente a QoS de cada classe de serviço.

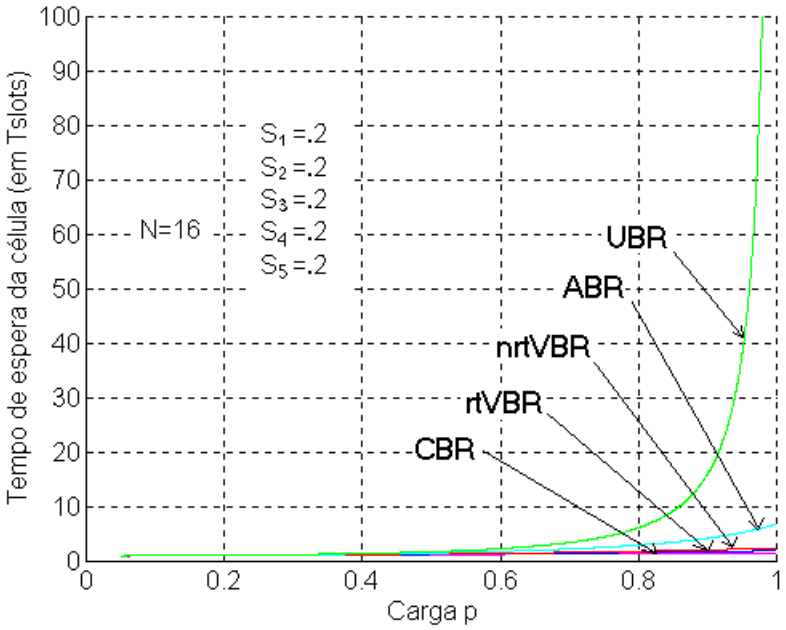


Figura 3.9 (a)

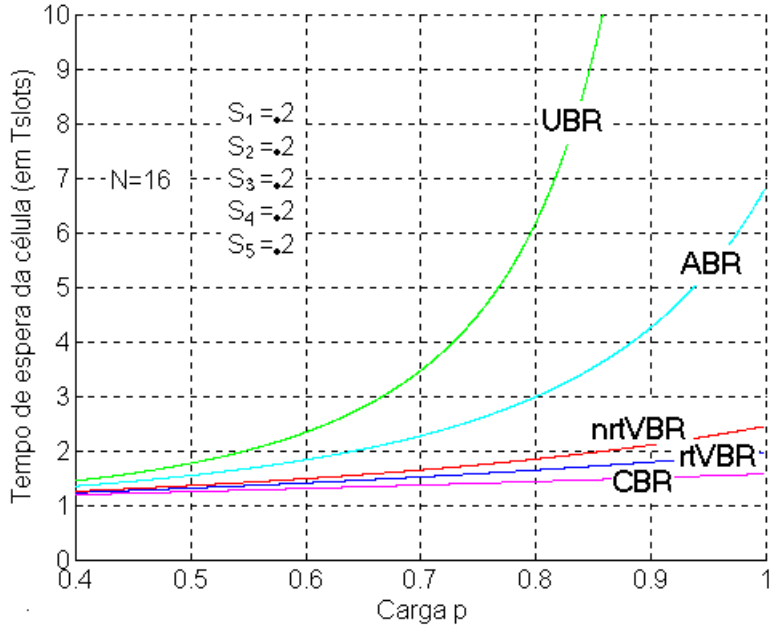


Figura 3.9 (b)

Figura 3-9: *Desempenho de um comutador 16x16 com tráfego da rede distribuído igualmente entre as classes de serviço.*

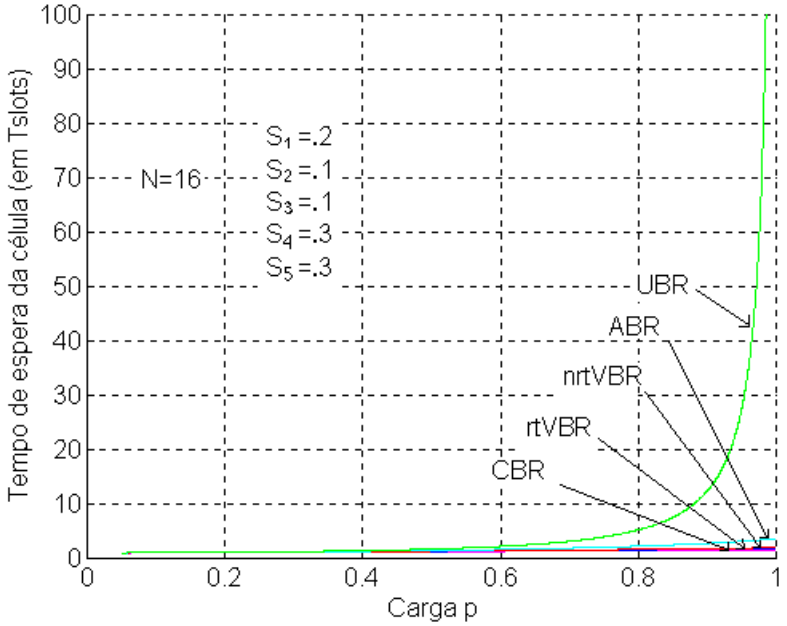


Figura 3.10 (a)

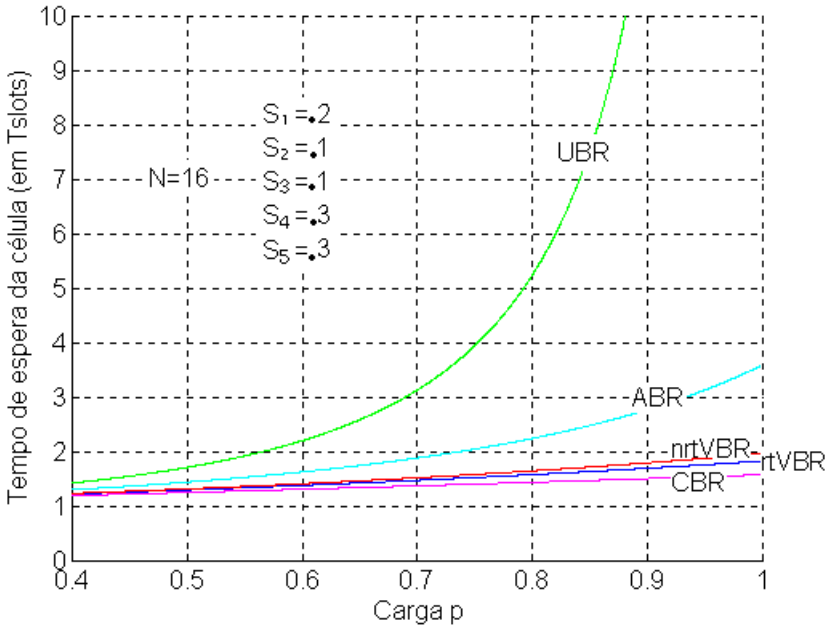


Figura 3.10 (b)

Figura 3-10: *Desempenho de um comutador 16x16 com 60% do tráfego da rede nas classes de serviço de mais baixa prioridade.*

3.6 Conclusão

Neste capítulo foram apresentadas três estruturas para comutadores ATM de alta velocidade. Os comutadores possuem um conjunto de cinco *buffers* em cada ponto de cruzamento, um *buffer* para cada tipo de serviço. A principal diferença entre as três estruturas propostas é a quantidade de caminhos paralelos no barramento que ligam os *buffers* de entrada aos *buffers* dos pontos de cruzamentos. O esquema de encaminhamento favorece as células de serviços prioritários.

Um estudo de caso foi feito para comparar o desempenho das três estruturas e verificou-se que a Estrutura 3 apresenta melhor desempenho e prioriza efetivamente o tráfego com maior prioridade. A Estrutura 1 apresenta o menor desempenho ao custo de baixa complexidade. A Estrutura 2 apresenta desempenho compatível com a Estrutura 3, ao custo de complexidade intermediária, porém não dá prioridade efetiva ao tráfego com maior exigência de qualidade de serviço.

Na proposta feita para a Estrutura 3 considerou-se um barramento dedicado a transmitir as células de uma entrada a cada *buffer* de ponto de cruzamento, para cada tipo de serviço. Simulações podem ser feitas a fim de especificar a quantidade de barramentos paralelos, assumindo o compromisso entre desempenho e complexidade

do comutador. Os serviços menos prioritários poderiam transferir uma única célula em vez de transmitir células em paralelo aos *buffers* dos pontos de cruzamentos relacionados a cada saída.

Apresentou-se também um modelo para análise de desempenho para a Estrutura 3. Os resultados da análise mostraram que a estrutura proposta apresentou desempenho adequado para garantir a QoS exigida pelas classes de serviços previstas no ATM. Em condições de 80% de tráfego na rede pertencentes às classes de serviço mais prioritárias (40% de CBR, 20% de rtVBR e de 20% nrtVBR), para cargas de 90%, o tráfego possui atraso máximo de $3xT_{slots}$ (T_{slots} = tempo para transmissão de uma célula) para células CBR e rtVBR; e atraso menor que $5xT_{slots}$ para células nrtVBR. Os atrasos das células ABR e UBR são consideráveis apenas para situações de carga acima de 80%. Já, em condições de 60% do tráfego de rede pertencentes às classes de serviço de menor prioridade, observou-se que os serviços CBR, VBR e ABR tem atrasos pequenos (menores que $3xT_{slots}$) para qualquer situação de carga da rede, e que o serviço UBR tem atraso considerável apenas para carga superior a 95%.

Capítulo 4

4 Comutador ATM com escalonamento de células distribuído

4.1 Introdução

Um comutador ATM com estrutura *crossbar* usando *buffers* grandes nas portas de entrada e *buffers* pequenos em cada ponto de cruzamento foi recentemente proposto [43]. A proposta em [43] e as propostas feitas no capítulo anterior têm em comum uma estrutura *crossbar* com *buffers* pequenos nos pontos de cruzamento e *buffers* grandes nas linhas de entrada e ambas foram baseadas em [44]. Em [43] a estrutura de comutação combina um esquema de enfileiramento de saída virtual (VOQ) nas portas de entrada e buffers pequenos nos pontos de cruzamento, usa um algoritmo de escalonamento de células distribuído em duas fases. Na primeira fase, para cada fila de saída virtual, o escalonador seleciona uma célula para ser transmitida ao buffer do ponto de cruzamento. Na segunda fase, para cada porta de saída, outro escalonador seleciona a célula que será transmitida pela porta de saída. A estrutura proposta em [43], parece muito interessante para aplicações em *backbone* pois o processamento do escalonador pode ser distribuído nas filas de entrada e nas filas dos pontos de cruzamento evitando a principal desvantagem das estruturas com armazenamento na entrada.

Neste capítulo, a estrutura de comutação com buffers nas entradas e nos pontos de cruzamento é usada, porém, diferentemente de [43], as células são discriminadas nos *buffers* de entrada em classes de serviço e, usando uma modificação da técnica de enfileiramento de saída virtual é proposta uma nova estrutura de comutação. Um modelo de análise para essa estrutura é desenvolvido e os resultados da análise são comparados com os resultados obtidos através de um modelo de simulação, desenvolvido em pacote de software *AweSim* na versão 1.4 [45] .

4.2 Estrutura do comutador ATM com escalonamento de células distribuído

A nova estrutura é apresentada na Fig. 4–1. A cada porta de entrada um conjunto de N *buffers* é providenciado. Em cada conjunto há um *buffer* para cada classe

de serviço. Cada um dos N *buffers* de cada porta de entrada consiste de filas lógicas separadas para as classes de serviço. A cada porta de entrada, após a leitura e atualização dos cabeçalhos das células, cada célula recebe um rótulo (*time stamp*) usado com o propósito de escalonamento. A célula é então discriminada de acordo com a porta de saída e a classe de serviço à qual pertence e é armazenada em uma das filas de classe de serviço virtuais.

Para cada *buffer* de entrada é providenciado um escalonador (*EE*) para selecionar a célula que será transmitida usando um algoritmo de escalonamento apropriado. Esta seleção é apenas local, ou seja, a célula selecionada é a próxima célula desse *buffer* a ser copiada no buffer do ponto de cruzamento. Uma vez que um escalonador de entrada é colocado em cada *buffer* de entrada em todos os enlaces de entrada e o algoritmo de escalonamento pode ser executado em paralelo, uma alta taxa de processamento agregado pode ser conseguida pelos escalonadores.

Uma cópia da célula enviada permanece no buffer de entrada e só é descartada quando o EE correspondente receber a confirmação de envio desta célula pela porta de saída; caso contrário, no próximo ciclo esta célula participa novamente do algoritmo de escalonamento de células.

Cada *buffer* de entrada possui uma linha separada conectando-o ao *buffer* do ponto de cruzamento correspondente (*PX*) e a um *buffer* auxiliar (*BA*) como é mostrado na Fig. 4–1. O *buffer* do ponto de cruzamento tem capacidade para acomodar apenas uma célula e o *buffer* auxiliar pode acomodar as informações locais (*IL*) de cada *buffer* de entrada. O rótulo *IL* (informações de *time stamp* e da classe de serviço) é transmitido para o *buffer* auxiliar (*BA*) usado pelo escalonador do ponto de cruzamento (*EX*) colocado em cada enlace de saída. Usando a *IL*, o *EX* executa o algoritmo de escalonamento apropriado para selecionar a próxima célula a ser transmitida; autoriza o *buffer* do ponto de cruzamento correspondente a transmiti-la através da linha selecionada; e envia um sinal de reconhecimento ao *EE* correspondente à célula selecionada, para que a sua cópia seja descartada no buffer de entrada. Uma vez que cada porta de saída possui um escalonador de ponto de cruzamento e cada um pode rodar independentemente, uma alta capacidade total de escalonamento pode ser conseguida.

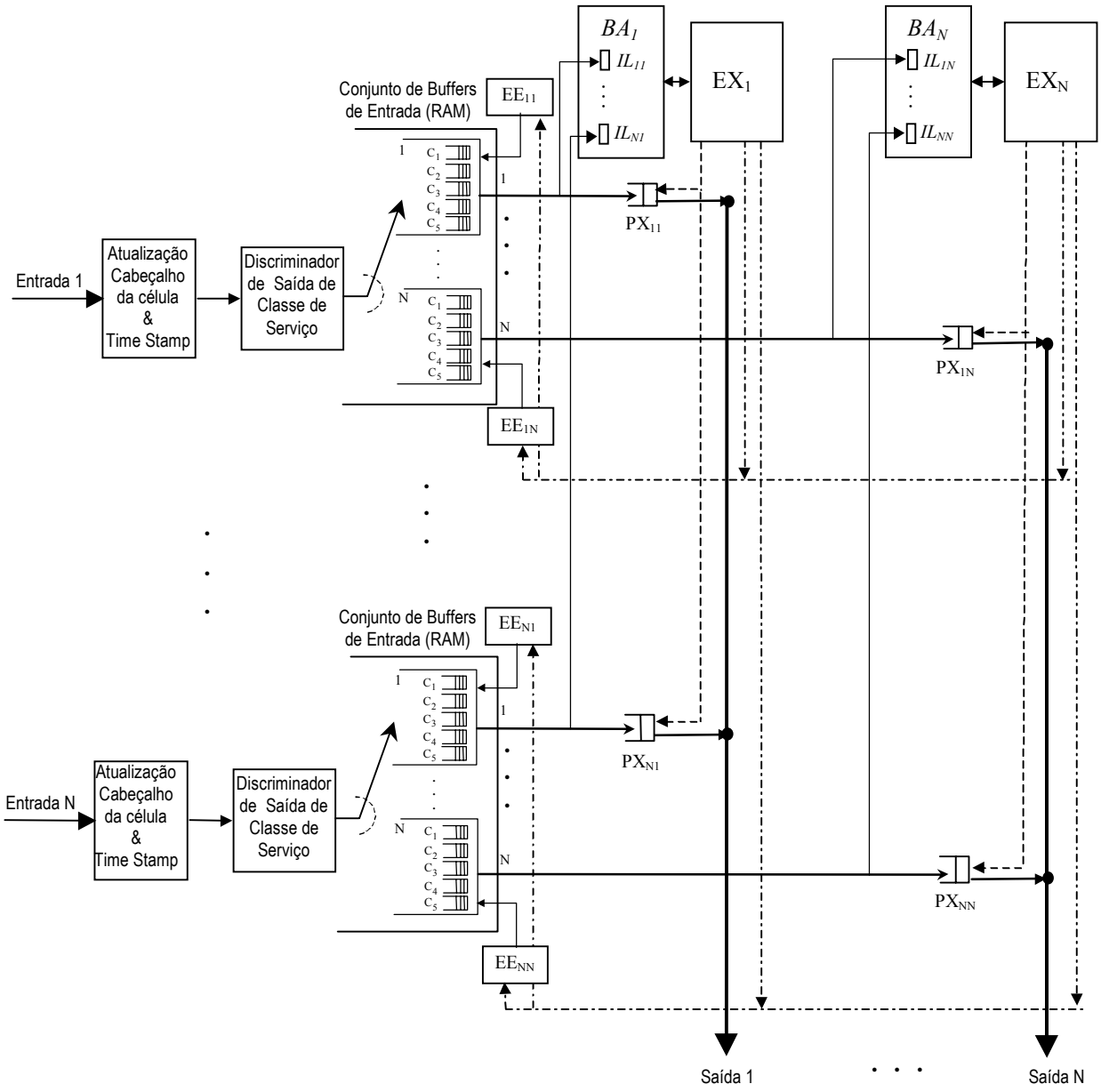


Figura 4-1: *Estrutura do comutador ATM com buffers grandes nas portas de entrada e buffers pequenos nos pontos de cruzamento, usando escalonamento de células distribuído.*

4.3 Modelo para análise de desempenho para o Comutador ATM com escalonamento de células distribuído

O tipo do algoritmo de escalonamento a ser adotado afeta o desempenho da estrutura de comutação proposta. Como um exemplo para a análise de desempenho, será usado um algoritmo de escalonamento não preemptivo por prioridade. O escalonador de entrada sempre escolhe a célula com mais alta prioridade, e se existir

mais que uma célula com a mesma prioridade, a célula com o tempo de atraso mais longo é escolhida.

O algoritmo usado para selecionar a célula que será transmitida dos buffers de entrada para os buffers dos pontos de cruzamento é descrito, considerando que as classes de serviço C_1, C_2, C_3, C_4 e C_5 estão em ordem decrescente de prioridade. O escalonador da entrada *i* e saída *j* (EE_{ij}) examina primeiro se a classe de serviço C_1 tem células a transmitir. Se alguma célula estiver esperando nesta fila, a célula que tiver o maior tempo de espera é copiada para o buffer do ponto de cruzamento *ij* (PX_{ij}). Ao mesmo tempo, IL_{ij} é transmitido para o buffer auxiliar (BA_j). Se não existirem células esperando na fila lógica C_1, então a fila lógica C_2 será examinada e assim por diante, até que a fila lógica C_5 tenha sido examinada. Este procedimento é executado em paralelo em cada escalonador de entrada.

O segundo algoritmo é usado para determinar a célula que será transmitida dos buffers dos pontos de cruzamento ($PX_{i,j}$, *i=1..N)* pela porta de saída *j*. O escalonador na porta de saída *j* (EX_j) usa o IL_{ij} da célula selecionada, obedecendo ao critério de prioridade descendente. Se mais que uma célula de mesma prioridade estiver esperando, aquela que tiver o maior tempo de espera é escolhida para ser transmitida no próximo *time slot*. Como o algoritmo acima é usado a cada *time slot*, a classe de serviço C_5 pode não ser atendida por vários ciclos. Mas, o algoritmo proposto é muito eficiente para atender às restrições dos serviços, assim a qualidade de serviço (*QoS*) pode ser satisfeita. Todos os escalonadores das saídas executam simultaneamente o procedimento descrito e portanto, um comutador simples e com velocidade extremamente alta pode ser implementado.

As células de cada classe de serviço são distribuídas entre todos os buffers de saída e são servidas de acordo com a ordem de prioridade não preemptiva. Assim, o modelo matemático desenvolvido na Seção 3.5 do capítulo anterior, pode ser usado para o Comutador ATM com escalonamento de células distribuído. Adaptando-se os parâmetros necessários, o tempo médio de espera das células é dado pela Eq. 4.1

$$E\{W_h\} = T_{slot}\left[1 + \frac{(N-1)p}{2N(1-p\sigma_{h-1})(1-p\sigma_h)}\right] \qquad (4.1)$$

onde,

- ***Tslot*** é o tempo necessário para transmitir uma célula.
- **N** = Número de entradas = Número de saídas da estrutura de Comutação
- **p** é a probabilidade de chegar uma célula em um slot
- $\sigma_h = \sum_{k=1}^{h} S_k$, $\sigma_0 = 0$, $\sigma_r = 1$

4.4 Análise de Desempenho do Comutador ATM com escalonamento de células distribuído

Para validar o modelo teórico apresentado, foi desenvolvido um modelo de simulação utilizando a ferramenta de software AweSIM. O AweSIM [45] é simulador baseado em eventos discretos e usa a linguagem de simulação Visual SLAM (*Visual Simulation Language for Alternative Modeling*) para a plataforma Windows 95.

Para a análise e simulação, foi considerado que as classes de serviço C1, C2, C3, C4 e C5 do modelo são respectivamente as classes de serviço do ATM: CBR, rtVBR, nrtVBR, ABR e UBR. Na Fig. 4–2 (a) e (b) são mostradas as curvas obtidas por simulação e pelo modelo teórico. Foram computados os tempos médios que cada célula esperou para ser transmitida na saída S1, considerando uma estrutura de três entradas e três saídas e o tráfego de entrada igualmente distribuído entre as classes de serviço.

Na Figura 4–2 (b) é destacado o trecho em que o tempo de atraso de células é no máximo de 10 vezes o tempo de transmissão de uma célula (T_{slot}) para melhor comparação do modelo simulado e teórico. Pode-se observar que os resultados dos modelos teóricos e simulados são equivalentes para carga na rede até 50%. A partir daí a aderência das curvas simuladas e teóricas vai diminuindo à medida que aumenta a carga na rede, e essa aderência é menor para a classe de serviço menos prioritária (UBR). Para as classes de serviço CBR e VBR, as mais prioritárias, as variações entre o modelo teórico e simulado é sempre menor que a metade de T_{slot}. Para a classe de serviço ABR a aderência das curvas simulada e teórica diminui, quando a carga está em torno de 80% atingindo um desvio máximo entre curvas de aproximadamente um T_{slot}. Para carga em torno de 90% a curva simulada do tráfego ABR apresenta variação, provavelmente devido a erros de arredondamento. Conforme pode ser observado na Fig.4–2(b), essa instabilidade está dentro de um desvio de 3%.

A simulação apresenta resultados ligeiramente melhores que o modelo teórico para os tráfegos UBR. e ABR. Para os tráfegos CBR e VBR, ao contrário, os resultados simulados são ligeiramente piores que os valores teóricos. Para tráfego acima de 85%, entretanto, o comutador já está operando na região instável. Nessa região, não é possível ter resultados confiáveis, pois um pequeno aumento na carga pode levar a um atraso considerável.

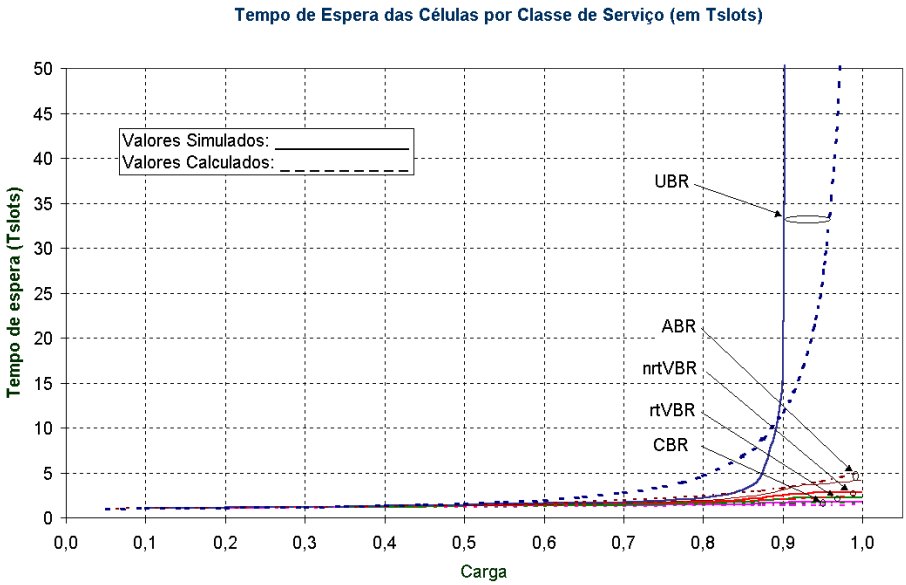


Figura 4.2:(a)

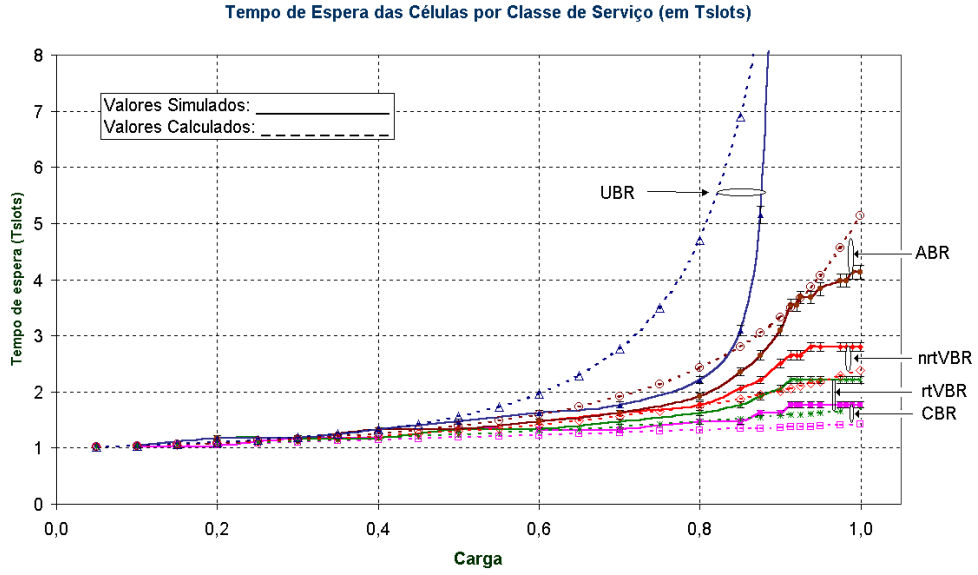


Figura 4.2 (b)

Figura 4-2 (a) e (b) *Comparação dos valores simulados e teóricos do Tempo de espera da célula em função da carga na rede. – Comutador com 3 entradas e 3 saídas, carga distribuída igualmente entre as classes de serviços.*

Na Fig. 4–3 é mostrada a utilização média de um enlace de saída, com a variação da carga nas entradas do comutador. Pode-se observar que a utilização média do enlace é proporcional a variação da carga no comutador. Observa-se ainda pela Fig. 4.–3 que a utilização do enlace não atinge 100%. Isso ocorre porque o modelo de simulação considerou que os enlaces de entrada e saída não estão sincronizados. Assim, a célula que chega a um enlace de entrada em um determinado *time slot* é encaminhada apenas no *time slot* seguinte, mesmo que o enlace de saída esteja vazio no instante de chegada da célula no enlace de entrada.

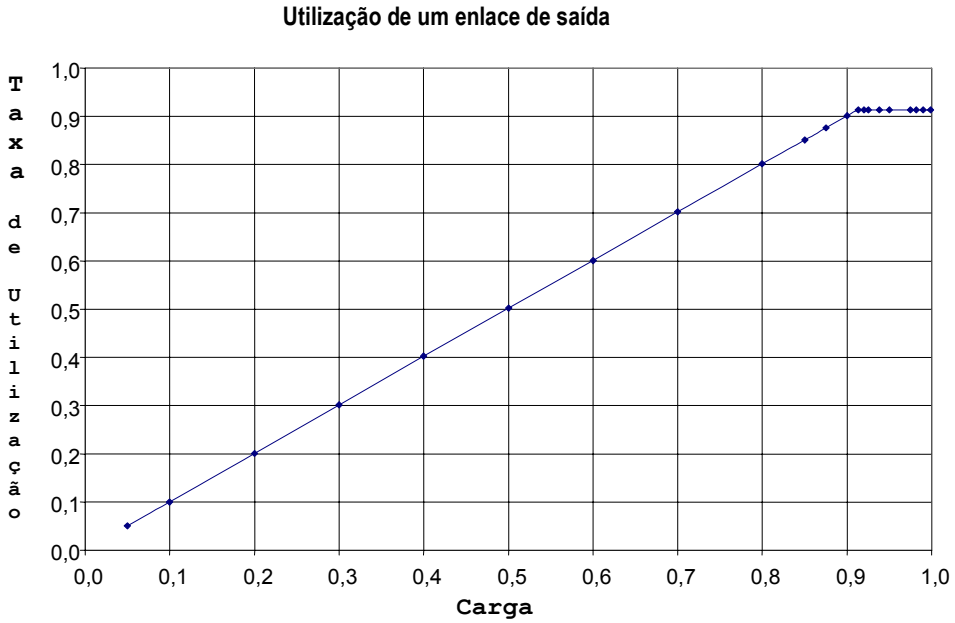


Figura 4-3: *Utilização do enlace de saída S1 com a variação da carga nas linhas de entrada da Estrutura do comutador ATM proposta usando escalonamento de células distribuído obtido através de simulação.*

Na Fig. 4–4 é apresentada a vazão da estrutura do comutador obtido pelo modelo simulado. As curvas mostram a razão entre as células que foram transmitidas a uma das saídas (incluindo a célula que está sendo servida) e as células que chegaram às linhas de entrada da estrutura com destino a essa saída, durante a simulação, em diferentes situações de carga no comutador. Pode-se observar pela curva obtida, que a estrutura obtém desempenho de 100% em qualquer situação de carga para as classes de serviço mais prioritárias (CBR e VBR). As classes de serviço menos prioritárias sofrem um decréscimo de vazão quando a carga no comutador é maior que 90%. A vazão média se mantém sempre superior a 90%. Essa queda de vazão ocorre porque o serviço UBR tem vazão próxima de 60% quando a carga da rede está em torno de 100%. A queda da vazão do tráfego UBR ocorre devido ao fenômeno conhecido como inanição (*starvation*). Ou seja, quando a rede está operando com carga em torno de 100%, a vazão de células UBR diminui, pois enquanto células UBR esperam pelo atendimento, outras

células mais prioritárias chegam à estrutura de comutação. Assim, o tempo de espera das células menos prioritárias pode aumentar indefinidamente provocando a inanição dessas células.

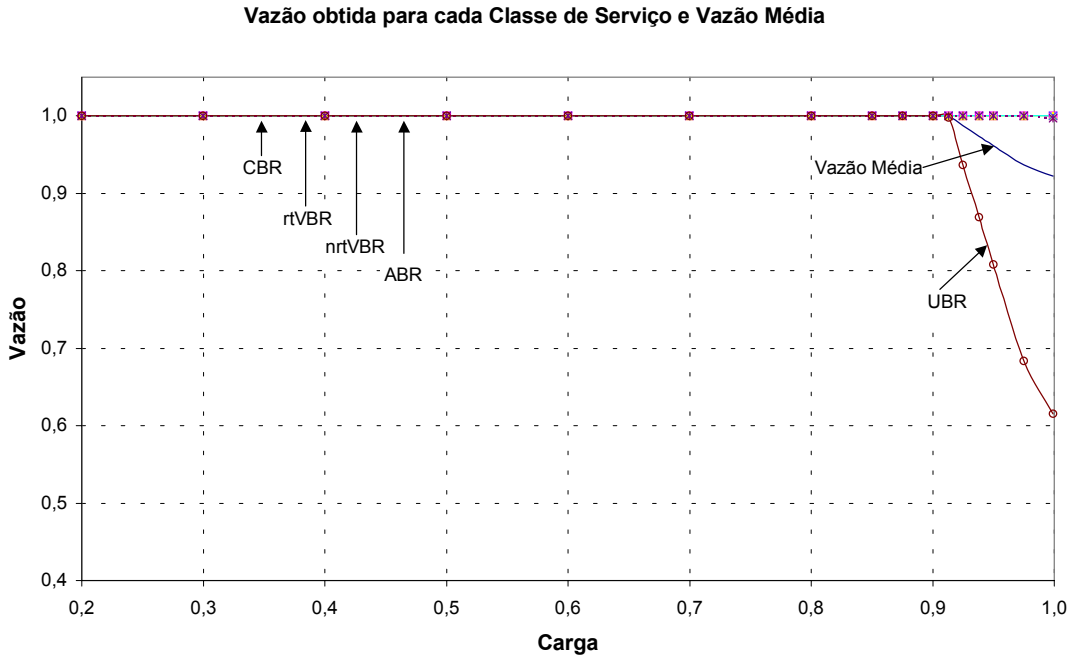


Figura 4-4: *Vazão do comutador ATM com a estrutura proposta, usando escalonamento distribuído de células. Curva obtida através de simulação.*

Na Fig. 4–5 é mostrada a quantidade de células esperando nos buffers para cada tipo de serviço, em função da variação da carga nas linhas de entrada do comutador. Pela Fig. 4–5 pode-se observar que, quando o tráfego é igualmente distribuído entre as classes de serviços, não são necessários *buffers* grandes para os serviços ABR, VBR e ABR. Apenas o serviço UBR necessita *buffers* grandes para que não haja degradação do serviço, e mesmo assim quando a estrutura de comutação estiver trabalhando com cargas superiores a 90%.

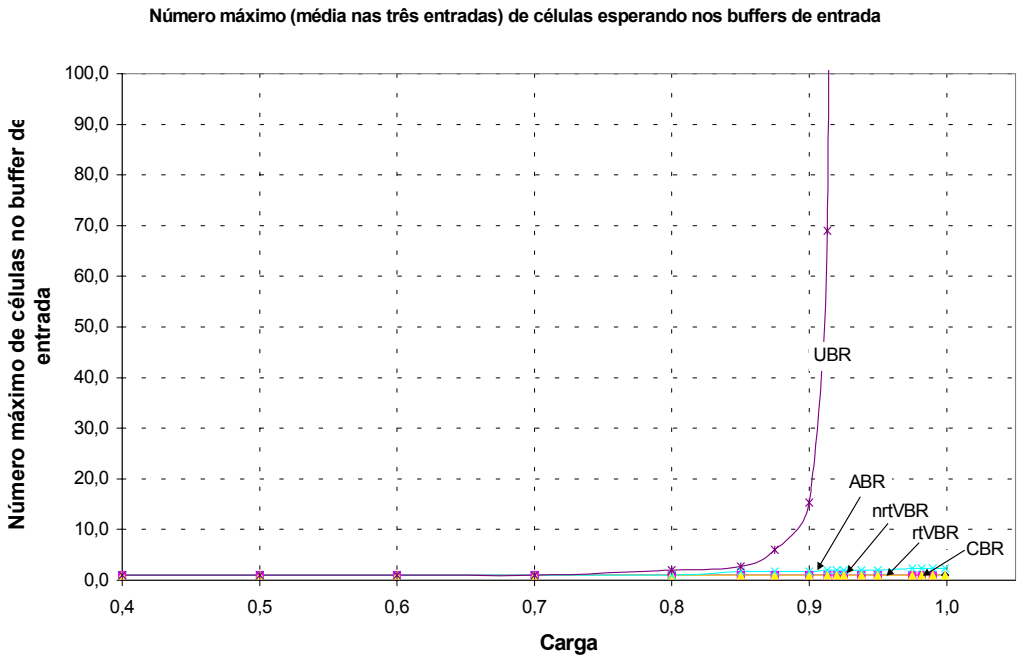


Figura 4-5: *Número de células esperando nos buffers de entrada para cada Classe de Serviço em função da variação da carga nas linhas de entrada do comutador.*

4.5 Conclusão

Neste capítulo foi apresentada uma estrutura de comutação ATM com buffers grandes nas entradas e buffers pequenos nos pontos de cruzamento, usando escalonamento de células distribuído. Pela discriminação das células de entrada de acordo com a classe de serviço e usando uma modificação do algoritmo de enfileiramento de saída virtual (VOQ *Virtual Output Queued*) o algoritmo de escalonamento de células pode ser distribuído nos buffers de entrada e nos buffers dos pontos de cruzamento. A estrutura proposta tem vazão teórica de 100%.

Um modelo matemático foi apresentado e, baseado nesse modelo e em resultados de simulação foi feita a análise de desempenho da estrutura proposta. Os resultados obtidos com a simulação mostraram que o modelo matemático é adequado e pode ser usado para analisar a estrutura proposta.

Capítulo 5

5 *IP Switching* (Comutação de Tráfego IP)

5.1 Introdução

O crescimento explosivo do tráfego na Internet está exigindo roteadores de alta capacidade. Estes roteadores combinam a alta velocidade de comutação da Camada 2 com as características de roteamento da Camada 3. Esta técnica de comutação é conhecida como comutação de tráfego *IP* ou *IP Switching*. Muitos mecanismos para *IP Switching* foram propostos na literatura [46], [47], [55]-[57], porém, a maioria dessas propostas é baseada nas estruturas de comutação ATM que são especializadas em comutação de pacotes de comprimento fixo. Para serem processados pelos comutadores, os pacotes IP são segmentados e depois de comutados são novamente remontados. Algumas propostas, como por exemplo as apresentadas em [46] e [47], adaptam os algoritmos de escalonamento de células, melhorando o desempenho da comutação de pacotes de tamanho variável. Essas propostas levam em consideração que um pacote gera uma seqüência de células com os mesmos requisitos de comutação, entretanto a segmentação/remontagem continuam sendo necessárias.

A maioria das estruturas que trabalha com pacotes variáveis usa escalonadores de fluxo localizados nos enlaces de saída [36], o que poderia conduzir ao problema de escalabilidade de fluxos. O processamento necessário para analisar os pacotes IP que chegam aos enlaces de saída cresce com o aumento do número de fluxos.

Neste capítulo são feitas duas propostas para a estrutura de *IP Switching* com escalonamento distribuído, capazes de comutar pacotes de comprimento variável. A primeira proposta usa algoritmos baseados na priorização das classes de fluxo e a segunda usa algoritmos baseados em *WFQ (Weighted Fair Queuing*) que asseguram a equidade no uso dos recursos pelas diferentes classes de fluxo.

5.2 *IP Switching* com escalonamento distribuído baseado em prioridade.

Os fluxos a serem escalonados são distribuídos entre os *buffers* de entrada e os buffers dos pontos de cruzamento, em uma estrutura de comutação (*switch fabric*) do

tipo *crossbar*. Cada escalonador de entrada administra apenas o grupo de fluxos de um enlace.

5.2.1 Estrutura do *IP Switching*

Na Fig. 5–1 é apresentada a estrutura proposta, que é uma adaptação da estrutura do Comutador ATM com escalonamento distribuído, apresentada no Capítulo 4, para operar com pacotes de tamanhos variáveis.

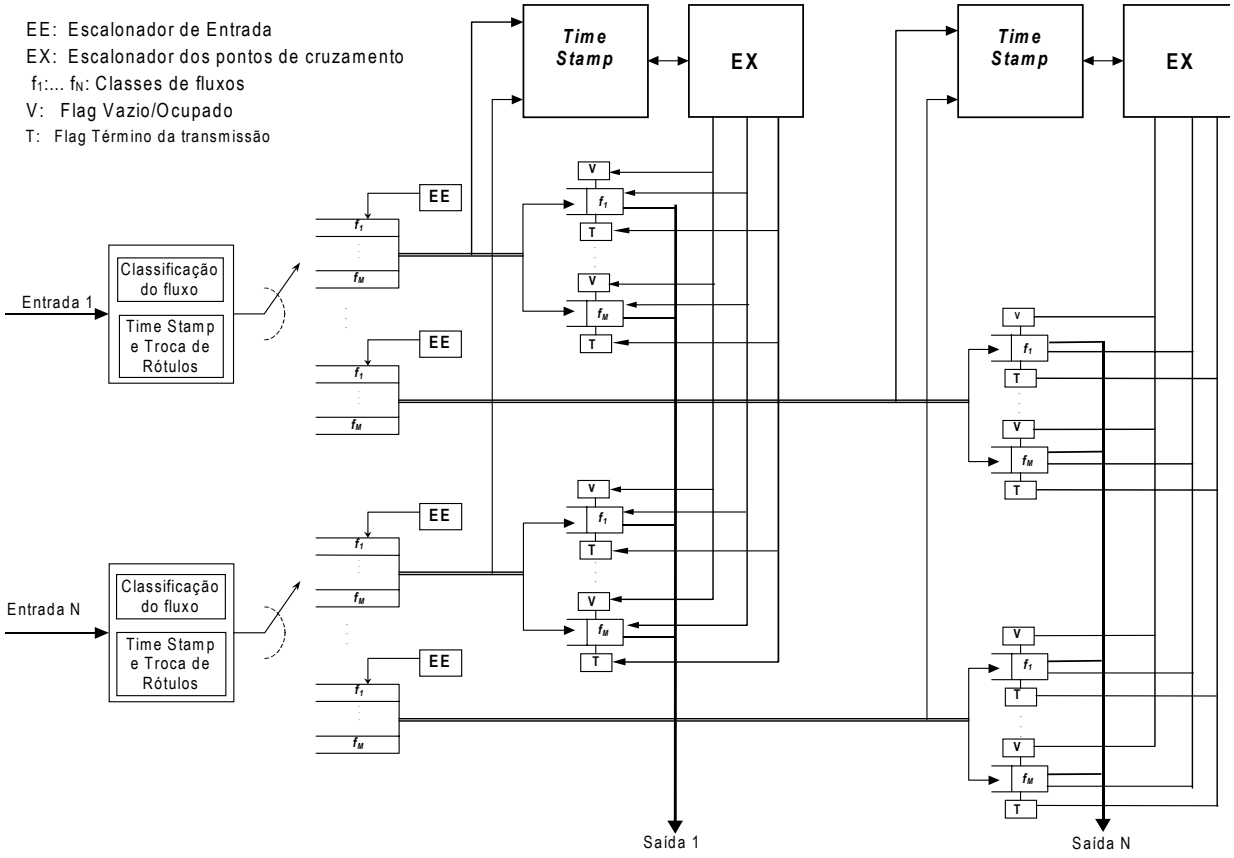


Figura 5-1: *Estrutura do IP Switching com escalonamento distribuído*

De forma semelhante ao comutador ATM com escalonamento distribuído, a cada enlace de entrada um pacote é classificado de acordo com a classe de fluxo ao qual pertence, recebe um selo de tempo (*time stamp*) para a finalidade de escalonamento e para o caso de MPLS (*MultiProtocol Label Switching*) é feita também a troca de rótulos (*labels*).

Um pacote classificado em um tipo de fluxo é armazenado em um dos N buffers de entrada (memórias RAM), um para cada enlace de saída. Em cada buffer de entrada, é providenciado um escalonador de entrada (EE) para selecionar o pacote que será

transmitido, usando um algoritmo de escalonamento apropriado. Esta seleção é apenas local e significa que o pacote selecionado é o próximo a ser transmitido naquele buffer.

Como um escalonador de entrada é colocado em cada buffer de entrada, em todas as entradas, pelo fato do algoritmo de escalonamento ser executado em paralelo, é possível conseguir um processamento agregado de alta capacidade.

Cada buffer de entrada contém M linhas separadas que o conecta ao conjunto de M buffers (um buffer para cada classe de fluxo) colocados no ponto de cruzamento correspondente, como é mostrado na Fig. 5−1. Cada buffer do conjunto de buffers dos pontos de cruzamento pode acomodar apenas um pacote de cada classe de fluxo.

Cada buffer do ponto de cruzamento possui dois *flags*, V e T. O *flag* V indica se o buffer está vazio ou ocupado enquanto que o *flag* T sinaliza o término da transmissão de um pacote. Quando um pacote é transmitido de um buffer de entrada ao buffer do ponto de cruzamento, ele leva a informação do selo de tempo que é usado pelo escalonador do ponto de cruzamento (EX) colocado em cada enlace de saída

Assim que um pacote é transmitido ao enlace de saída, o escalonador do ponto de cruzamento confere se os buffers dos pontos de cruzamento possuem pacotes para serem transmitidos examinando o flag V. Usando o tempo estampado no selo, o EX executa o algoritmo de escalonamento apropriado para selecionar o próximo pacote a ser transmitido e autoriza o buffer do ponto de cruzamento correspondente a transmitir pela saída correspondente.

Uma vez que cada saída tem um escalonador de ponto de cruzamento e cada um pode ser executado independentemente, uma alta capacidade de processamento global pode ser obtida.

A estrutura de comutador IP proposta pode manipular pacotes de qualquer tamanho, pois o final da transmissão de um pacote no buffer do ponto de cruzamento é conferido bit a bit.

5.2.2 Modelo de análise para o comutador IP, com escalonamento baseado em prioridade das classes de fluxo

O tipo de algoritmo de escalonamento usado afeta o desempenho do comutador IP proposto. Para analisar o comportamento do comutador IP com escalonamento distribuído será usado nessa seção, escalonador baseado em esquema de prioridade não preemptivo para as classes de fluxo. Para isso, quando o pacote é classificado na entrada, de acordo com sua classe de fluxo, o selo que cada pacote recebe, além da

informação do instante de chegada, deve conter também a classe de fluxo à qual pertence o pacote. Esse selo será usado apenas localmente, com propósitos de escalonamento.

Em cada entrada, o escalonador sempre escolhe o pacote de mais alta prioridade, e em caso de empate, escolhe o pacote com o maior tempo de espera e o transmite para o buffer do ponto de cruzamento correspondente à classe de fluxo do pacote escolhido. O escalonador do ponto de cruzamento escolhe entre os MxN buffers dos pontos de cruzamento aquele que contém o pacote de maior prioridade, e o autoriza a transmitir esse pacote. Em caso de empate, o escalonador escolhe entre os buffers que contêm pacotes de mesma prioridade, aquele cujo pacote tenha chegado primeiro.

Assim, o modelo de filas de prioridade não preemptiva pode ser usado para análise de desempenho da estrutura proposta para o comutador IP.

Para a análise as seguintes suposições são feitas. O número de entradas e de saídas da estrutura de comutação é N. Em cada enlace de entrada, a chegada de pacotes têm distribuição de *Poisson* com taxa média λ. A porcentagem de tráfego do fluxo ***i*** é S_i e $\sum_i S_i = 1$.

A probabilidade de um pacote em uma porta de entrada ser roteado para uma saída particular é $1/N$, então o tráfego do fluxo ***i*** em cada buffer agregado é $N\lambda S_i \frac{1}{N}$. Assumiu-se também que existem ***r*** fluxos diferentes e um fluxo genérico ***p*** pode assumir os valores: ***p= 1, 2, 3, ... r*** , onde *r* é o fluxo de menor prioridade. Os pacotes de mesma prioridade são servidos de acordo com a disciplina FIFO.

Para sistemas de filas que adotam disciplina de serviço de acordo com a prioridade e não-preemptivo, o cálculo do tempo médio de espera por pacote $E\{W_p\}$ é conhecido e pode ser determinado por:

$$E\{W_p\} = \frac{E\{T_0\}}{(1-\sigma_{p-1})(1-\sigma_p)} \qquad (5.1)$$

onde,

- $\sigma_p = \sum_{k=1}^{p} \rho_k \quad , \quad \sigma_0 = 0 \quad , \sigma_r < 1$
- $\rho_k = \left(\lambda S_k / \mu_k \right)$
- $\lambda \boldsymbol{S_k}$ é a taxa média de chegada de pacotes do *fluxo k*

- $1/\mu_k$ é o comprimento médio do pacote do *fluxo k*.
- $\rho_T = \lambda \sum_{k=1}^{r} S_k/\mu_k$ é a carga total por enlace
- $E\{T_0\} = \frac{1}{2}\sum_{k=1}^{r} \lambda\, S_k E\{R_k^2\}$
- $E\{R_k^2\}$ é o segundo momento da distribuição do comprimento dos pacotes. Para pacotes com comprimentos de acordo com a distribuição exponencial e média $1/\mu_k$, $E\{R_k^2\} = \left(1/\mu_k\right)^2$.

5.2.3 Análise de desempenho do comutador IP com escalonamento distribuído e escalonadores baseados em prioridade.

Para um exemplo numérico da análise de desempenho da estrutura do comutador IP com escalonamento distribuído, serão considerados cinco classes de fluxos f_1, f_2, f_3, f_4 e f_5, em ordem decrescente de prioridade. Considerou-se que os pacotes têm distribuição de comprimento exponencial com médias iguais à 800, 2400, 4000, 5600 e 8000 respectivamente aos fluxos f_1, f_2, f_3, f_4 e f_5 e que, os fluxos f_1 e f_2 representam cada um deles 30% do tráfego total, o fluxo f_3 representa 20% do tráfego e os fluxos f_4 e f_5 representam, cada um deles, 10% do tráfego total. A capacidade do enlace de saída é de 150 Mbps.

Na Fig. 5–2 estão traçadas as curvas para o tempo gasto pelos pacotes na estrutura de comutação. Essas curvas foram obtidas usando-se os parâmetros considerados acima, substituídos na Eq. 5–1, adicionando-se a essa equação o tempo de serviço de cada pacote, para diferentes situações de carga no comutador (ρ_T). O tempo de serviço de cada pacote é função de seu tamanho e pode ser calculado por $1/\mu_k$.

Pode-se observar, pela Fig. 5–2, que apenas o fluxo f_5 possui tempo de espera longo para carga no comutador acima de 90%. Abaixo de 80% de carga total no comutador, os pacotes das classes de fluxo com maior prioridade (f_1, f_2 e f_3) têm seu tempo no sistema inferior a 0,1 *ms*, os pacotes do fluxo f_4 têm o tempo no sistema inferior a 0,15 *ms* e os pacotes do fluxo f_5, pacotes de menor prioridade, têm seu tempo no sistema inferior a 0,36 *ms*.

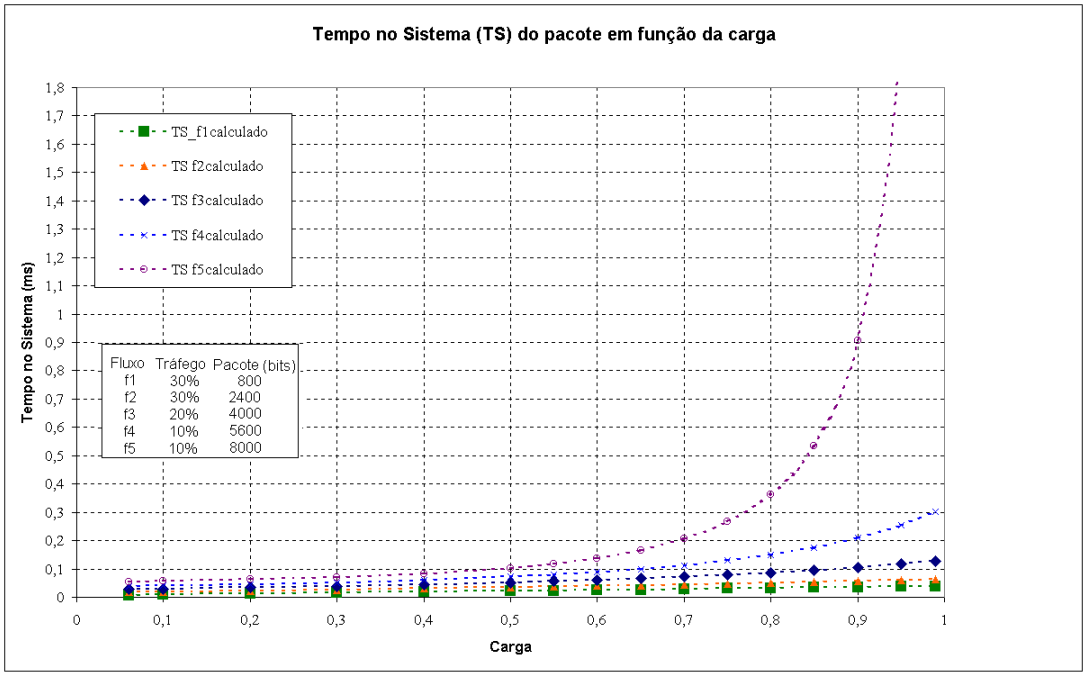


Figura 5-2: *Curvas do Tempo Médio no Sistema versus a Carga, para a estrutura IP Switching com escalonamento distribuído baseado na prioridade das classes de fluxo, obtido através do modelo matemático.*

Na Fig. 5–3 são apresentadas curvas para comparação dos resultados obtidos pelo modelo matemático usado, com os resultados obtidos através de simulação de eventos discretos usando a ferramenta *AweSim*. Os resultados demonstram que o modelo matemático usado é adequado para analisar o desempenho do comutador de IP, pois, as diferenças entre os resultados obtidos pela simulação e pelo emprego da expressão matemática são desprezíveis.

A taxa de utilização de um enlace de saída, de acordo com a variação da carga de entrada no comutador é mostrada na Fig.5–4. Pode-se observar que a Taxa de Utilização média do enlace de saída cresce proporcionalmente com o crescimento da carga. A utilização máxima do enlace, ou seja 100%, ocorre para valores de carga também em torno de 100%.

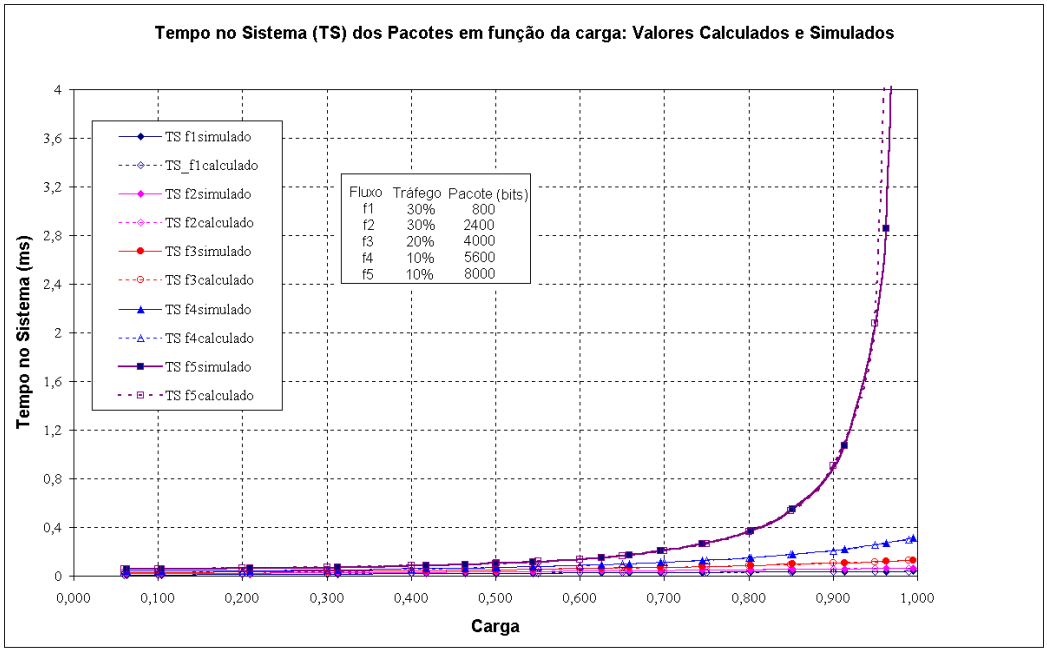


Figura 5-3: *Comparação dos Tempos Médios (dos pacotes) no Sistema versus a Carga, obtidos pelo uso do modelo matemático e pelo modelo de simulação, para a estrutura IP Switching com escalonamento distribuído, baseado na prioridade das classes de fluxo.*

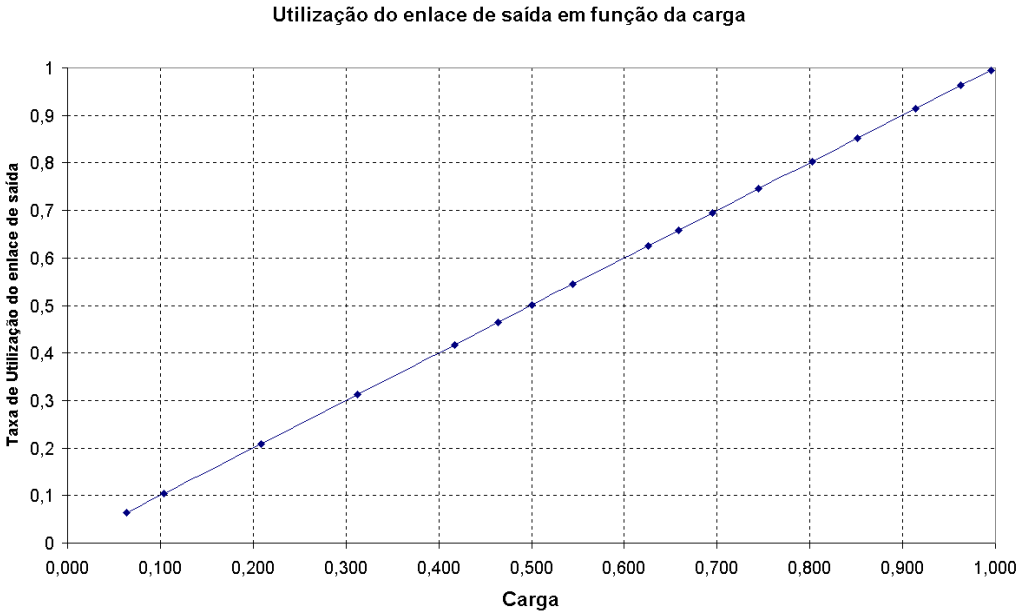


Figura 5-4: *Utilização de um enlace de saída com a variação da carga nas linhas de entrada para a estrutura IP Switching, com escalonamento de pacotes distribuído, baseado em prioridade. Resultados obtidos através de simulação.*

A vazão de pacotes na estrutura *IP Switching* proposta é mostrada na Fig. 5–5. A vazão foi obtida pela razão entre os pacotes que chegaram nas linhas de entrada da estrutura, endereçados a uma determinada saída, e os pacotes que foram transmitidos

por essa linha de saída durante a simulação. Pode-se observar que a vazão obtida por todos as classes de fluxo é muito próxima a 100%, para condições de buffer infinito nas linhas de entrada.

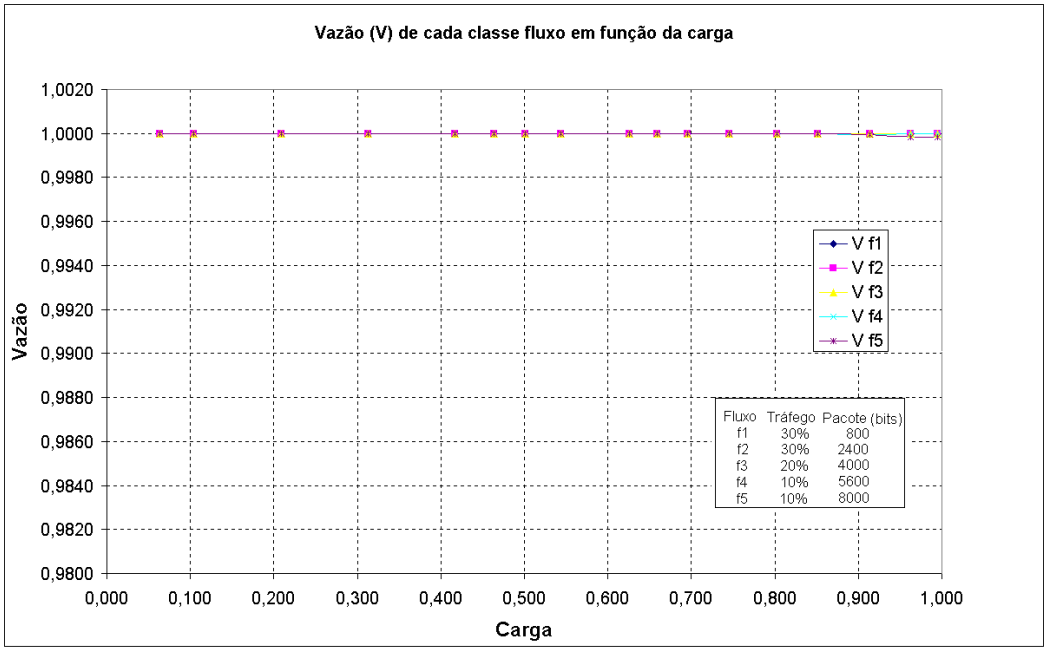


Figura 5-5: *Vazão da estrutura IP Switching usando escalonamento distribuído baseado em prioridade. Curva obtida através de simulação.*

Nas Fig. 5–6(a) e Fig. 5–6(b) são mostradas as curvas da ocupação dos buffers de entrada, em kbits, com os pacotes que estão esperando nos buffers das linhas de entrada para cada uma das classes de fluxo em função da carga, para que não haja perda de pacotes. Estas curvas foram obtidas através de simulação, para distribuição da carga de entrada entre as classes de fluxo f_1, f_2, f_3, f_4 e f_5 respectivamente iguais a 30%, 30%, 20%, 10% e 10%. Pode-se observar que, para a distribuição de carga usada, a capacidade dos buffers aumenta com a diminuição da prioridade da classe de fluxo. Assim, como pode ser observado na Fig.5–6(a) as classes de fluxo f_1, f_2, f_3, e f_4, necessitam buffers com capacidade de armazenamento respectivamente iguais a 5kbits, 16 kbits, 27 kbits e 46 kbits, para que não haja perda de pacotes. Conforme pode ser melhor observado na Fig. 5–6(b), a classe de serviço f_5 necessita buffer com capacidade igual à 750 kbits, consideravelmente maior que o exigido pelas outras classes de serviço, para que não haja perda de pacotes e consequentemente degradação da qualidade de serviço.

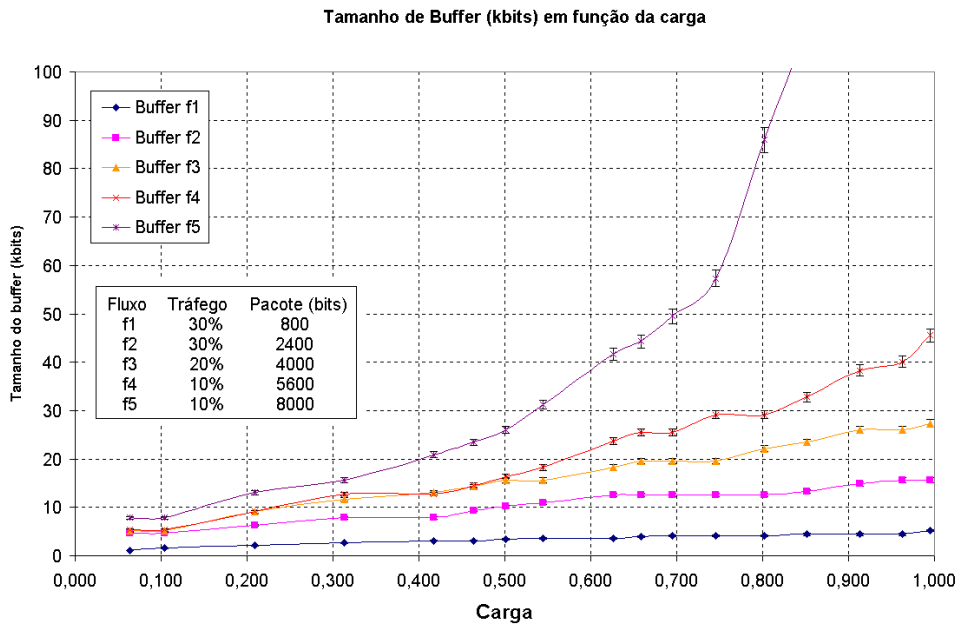


Figura 5.6(a):

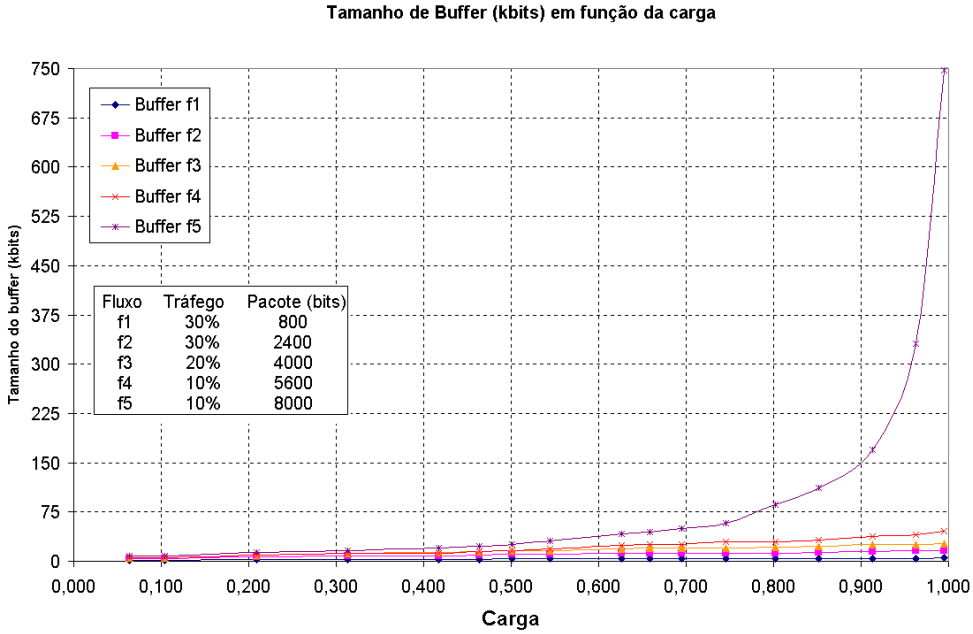


Figura 5.6 (b)

Figura 5-6: *Capacidade dos buffers de entrada em cada classe de fluxo em kbits, em função da variação da carga nas linhas de entrada da estrutura de comutação IP Switching, para distribuição de carga de 30%, 30%, 20%, 10% e 10% respectivamente para as classes de fluxo f_1, f_2, f_3, f_4 e f_5.*

5.3 Comutador de IP com escalonamento distribuído baseado em Enfileiramento imparcial usando Round Robin com déficit.

Roteadores típicos procuram garantir a equidade no uso de seus recursos, promovendo acesso justo para o tráfego nos enlaces de entrada. O uso da disciplina FIFO pode produzir justiça com propriedades exponencialmente ruins [48]. Várias propostas foram feitas para assegurar a equidade no uso dos recursos [48], [49], [51]-[53] e conforme foi discutido em [48], quando um algoritmo garante o uso eqüitativo de recursos, sua complexidade é O(log(M)), onde **M** é o número de fluxos ativos, o que é oneroso para redes de altas velocidades.

Nagle [50] propôs um algoritmo de escalonamento para as redes de datagrama, que usa a discriminação dos fluxos e emprega a disciplina de serviço *round-robin* para os fluxos em todos os enlaces de saída. O algoritmo proposto por Nagle identifica os fluxos através dos endereços fonte-destino, usa filas de saída separadas para cada fluxo e aplica a disciplina de serviço *round-robin* para atender as filas, o que impede que uma fonte aumente sua parte da largura de banda arbitrariamente. Quando uma fonte envia pacotes muito depressa, isto somente aumenta o tamanho de sua própria fila. Os pacotes de uma fonte mal comportada serão perdidos repetidamente. Entretanto, este esquema ignora o tamanho dos pacotes. Se o comprimento médio dos pacotes fosse igual para todos os fluxos, cada fluxo teria acesso a uma parte igual da largura de banda do enlace de saída. Porém, se uma fonte envia pacotes longos enquanto outra envia pacotes curtos, a fonte que utiliza pacotes grandes teria acesso a uma maior quantidade da banda de passagem. No esquema *round-robin,* para o pior caso, um fluxo pode obter ***Max/Min*** vezes a largura de banda de outro fluxo, onde ***Max*** é o comprimento máximo de pacote e ***Min*** é o comprimento mínimo de pacote [48].

O algoritmo baseado em prioridades apresentado na Seção 5.2, atende aos requisitos de tempo de atraso limitado, exigido pelos fluxos de maior prioridade, porém, quando pacotes com diferentes comprimentos são usados, essa disciplina pode provocar o uso desequilibrado de um enlace de saída por um dos fluxos de maior prioridade, da mesma forma que a disciplina *round-robin* proposta por Nagle. Além disso, algoritmos baseados em prioridade podem ocasionar a inanição (*starvation*) dos fluxos de menor prioridade.

Shreedhar e Varghese [48] apresentaram um esquema, ao qual chamaram *round-robin com déficit* (DRR – *Deficit Round Robin*), que possui a mesma vantagem de tempo de atendimento constante da disciplina r*ound-robin tradicional,* porém, sem a desvantagem da falta de equidade provocada pelas prováveis diferenças de comprimento dos pacote usados por fluxos diferentes. O esquema proposto em [48] usa enfileiramento

estocástico para associar cada fluxo a uma fila. Uma cota de serviço é atribuída a cada fila e o serviço *round robin* é usado, respeitando a cota e o saldo acumulado de cada fila. A diferença entre os mecanismo *round-robin com déficit* e *round-robin tradicional* é que, se uma fila não puder enviar um pacote no ciclo anterior porque o comprimento de seu pacote era muito grande, o *saldo* remanescente do ciclo anterior é somado à cota do próximo ciclo. Assim o déficit provocado pela rejeição em um ciclo anterior é compensado no próximo ciclo. O esquema proposto em [48] supõe que o escalonadores de pacotes estão localizados nos enlaces de saída, o que poderia conduzir ao problema de escalabilidade de fluxo.

Uma adaptação da proposta apresentada em [48] será feita para ser usada em uma estrutura de *IP Switching* com escalonamento distribuído. A Fig. 5.8 apresenta um exemplo para esta nova estrutura. No exemplo é representada uma estrutura com três entradas e três saídas.

A diferença entre a estrutura apresentada na Seção 5.2 e nesta seção é basicamente o funcionamento dos escalonadores dos *buffers* de entrada (***EE***) e os escalonadores dos pontos de cruzamento (***EX***).

Da mesma forma que na estrutura apresentada anteriormente, um pacote é classificado, em cada enlace de entrada, de acordo com a classe de fluxo ao qual pertence. Para o caso do MPLS (*MultiProtocol Label Switching*) é feita também a troca de rótulos. Um pacote classificado por um tipo de fluxo é armazenado em um dos N buffers de entrada. Cada buffer de entrada possui um escalonador de entrada (***EE***) que é usado para selecionar o pacote que será transmitido usando o algoritmo de escalonamento, que nesse caso é o *round-robin com déficit*. Cada buffer de entrada é conectado ao conjunto de buffers colocado no ponto de cruzamento correspondente, ou seja, um buffer é colocado para cada classe de fluxo em cada ponto de cruzamento, como mostrado em Fig.5–8. Também na nova estrutura, cada buffer do ponto de cruzamento possui dois *flags*, ***V*** e ***T***. O *flag* ***V*** indica se o buffer está vazio ou ocupado enquanto que o *flag* ***T*** sinaliza o término da transmissão dos pacotes que estão armazenados no buffer do ponto de cruzamento. Assim que os pacotes armazenados no buffer do ponto de cruzamento autorizado são transmitidos ao enlace de saída, o escalonador ***EX*** examina o flag ***V*** para conferir se os buffers dos pontos de cruzamento possuem pacotes para serem transmitidos, executa o algoritmo *round-robin tradicional* para selecionar o próximo pacote a ser transmitido e autoriza o buffer do ponto de cruzamento correspondente a transmitir pela linha autorizada.

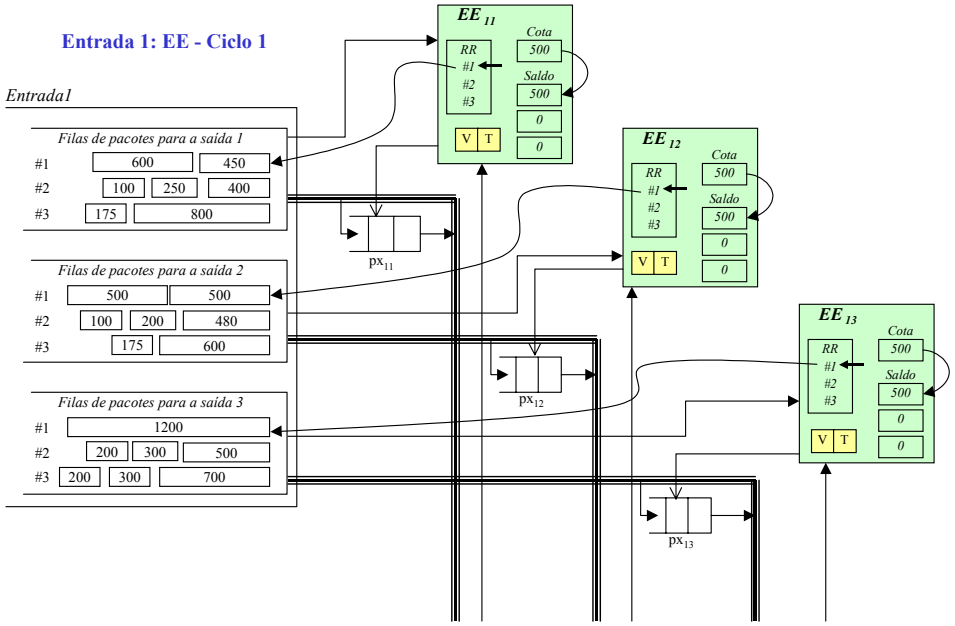


Figura 5.8 (a): Entrada 1 – Ciclo 1

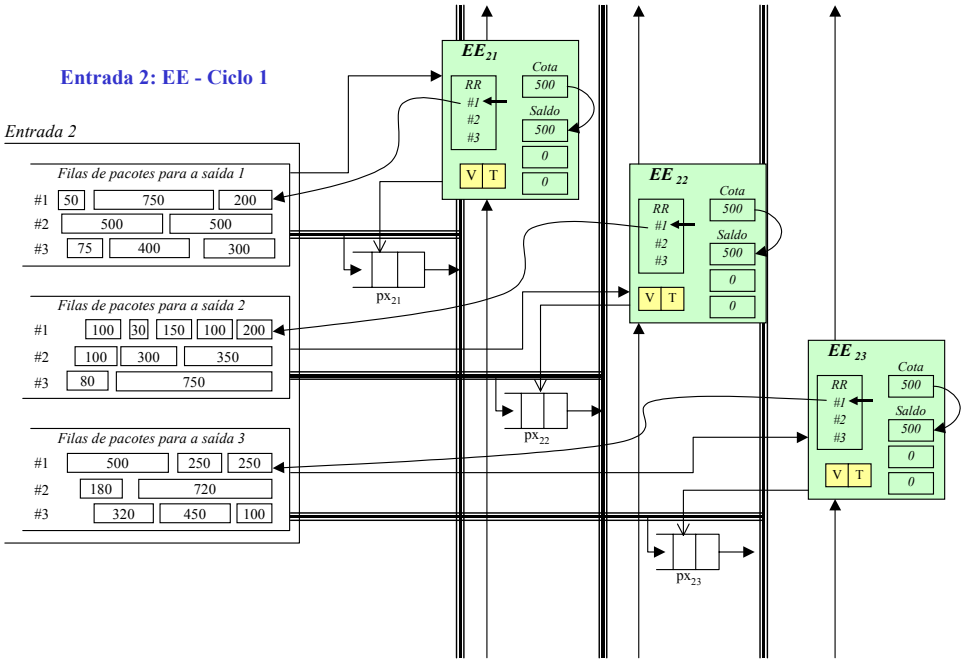


Figura 5.8 (b): Entrada 2 – Ciclo 1

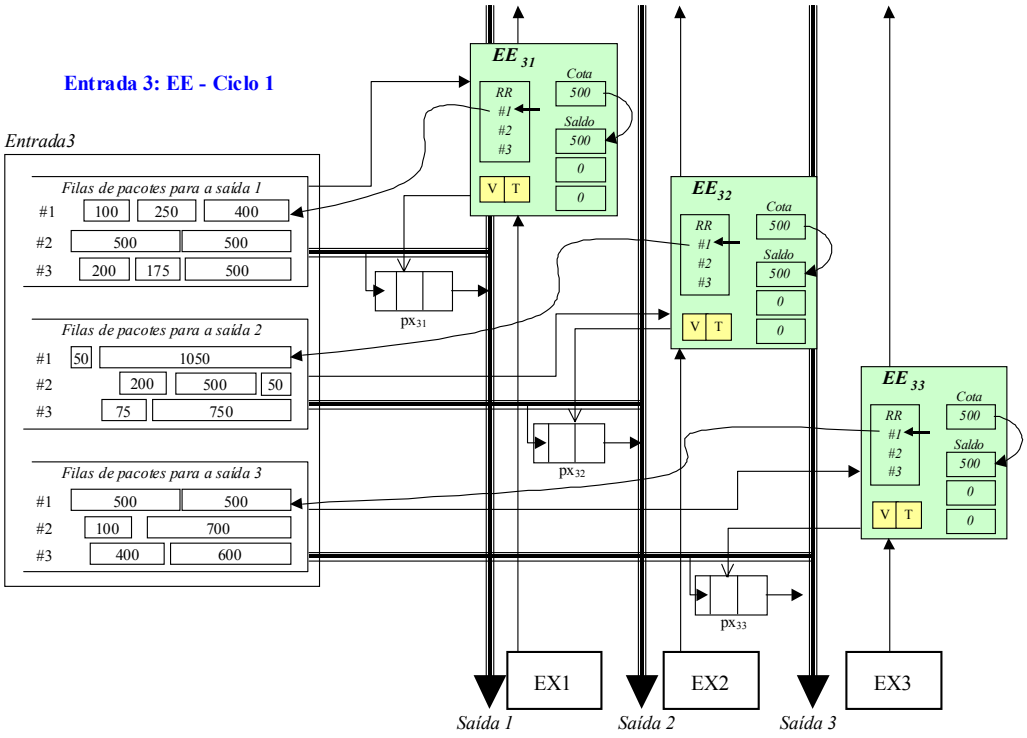


Figura 5.8: Entrada 3 – Ciclo 1

Figura 5-8: *IP Switching com escalonamento distribuído usando algoritmo round-robin com déficit nos escalonadores de entrada (EE_{jk}) e round-robin tradicional nos escalonadores dos pontos de cruzamento (EX_j).*

Estrutura com três entradas e três saídas e Cotas iguais a 500 bytes para cada classe de fluxo. No primeiro ciclo do algoritmo round-robin com déficit, todas as variáveis Saldo são inicializadas com o valor zero (0). O ponteiro do RR aponta para o topo da lista ativa. Em cada escalonador EE a primeira fila é atendida, o valor 500 atribuído à Cota é somado ao valor do Saldo. O valor remanescente, após o serviço da fila, permanece na variável Saldo. A Fig.5–8(a) mostra a Entrada 1, a Fig.5–8(b) mostra a Entrada 2 e a Fig.5–8(c) mostra a Entrada 3 detalhando as filas de entrada, os pontos de cruzamento e os escalonadores de cada buffer de entrada.

5.3.1 Escalonadores dos buffers de entrada: *Algoritmo round-robin com déficit.*

O algoritmo *round-robin com déficit* adaptado à estrutura com escalonamento distribuída será explicado com o auxílio das Fig. 5–8 e Fig. 5–9. Nessas figuras são mostrados os pacotes, que depois de terem sido discriminados de acordo com a saída correspondente e com a classe de fluxo ao qual pertencem, são armazenados em uma das MxN filas de cada entrada, onde M é o número de classes de fluxo e N é o número de saídas da estrutura. A Fig. 5–8(a) mostra as filas de pacotes da Entrada 1, e os escalonadores dos buffers para essa entrada. Cada classe de fluxo é identificado por um número precedido pelo símbolo "#". Os retângulos com números inscritos que

aparecem nas filas das Fig. 5–8 e Fig.5–9 simbolizam os pacotes, e o número indica o comprimento do pacote em bytes. Cada buffer de entrada possui um escalonador, que executa o algoritmo *round-robin com déficit* para selecionar o fluxo que transmitirá seus pacotes para o buffer do ponto de cruzamento. Cada escalonador mantém o controle do algoritmo round-robin para o atendimento dos fluxos, a cota autorizada para cada fluxo, o saldo acumulado para cada fluxo, e o estado dos *flags* ***V*** e ***T*** associado ao buffer controlado por ele. Como o algoritmo trabalha em ciclos, a medida de tempo usada para analisar o algoritmo é o número de ciclos. A quantidade de bytes atribuída ao *fluxo i*, ou f_i, é chamada $Cota_i$. A $Cota_i$ adicionada ao saldo acumulado em ciclos anteriores determina o tamanho da janela de transmissão de f_i em um determinado ciclo.

Os pacotes que entram através de fluxos diferentes e são endereçados a saídas diferentes são armazenados em filas diferentes. Sendo $Bytes_{i,x}$ o número de bytes enviados para ***fila i*** no ***ciclo x***, é permitido a cada ***fila i*** enviar pacotes no primeiro ciclo sujeito à restrição $Byte_{i,1} \leq Cota_i$. Se não houver mais nenhum pacote na ***fila i*** após a fila ter sido servida, a variável de estado chamada $Saldo_i$ é reajustada para 0. Caso contrário, a quantia restante ($Cota_i - Bytes_{i,x}$) é armazenada na variável de estado $Saldo_i$. Nos ciclos seguintes, a quantidade de largura de banda utilizável por este fluxo é o $Saldo_i$ do ciclo anterior somado a $Cota_i$. O *Pseudo-código* para este algoritmo é mostrado na Fig. 5–10.

Para evitar que o escalonador tenha que examinar filas vazias, é mantida uma de lista auxiliar, chamada ***ListaAtiva***, que contém os índices das filas com pelo menos um pacote. Sempre que um pacote chega a uma *fila i* previamente vazia, o índice ***i*** é acrescentado ao final da ***ListaAtiva.*** Sempre que o índice ***i*** estiver no topo da ***ListaAtiva***, o fluxo f_i é servido até que o limite $Cota_i + Saldo_i$ em bytes seja atingido. Se ao término desta oportunidade de serviço, a ***fila i*** ainda tem pacotes para enviar, o índice ***i*** é movido para o final da ***ListaAtiva***, caso contrário, o $Saldo_i$ é reajustado com o valor zero e o índice i é removido da ***ListaAtiva***.

No caso mais simples, atribuí-se um mesmo valor de ***Cota*** para todos os fluxos e nesse caso, o algoritmo *round-robin com déficit* coincide com o algoritmo enfileiramento imparcial ponderado (*WFQ - Weighted Fair Queuing*) [48]. Entretanto, para o *round-robin com déficit* é possível atribuir a cada ***fluxo*** *i* uma largura de banda diferenciada, e isso pode ser ajustado atribuindo-se diferentes valores de ***Cota*** a cada classe de fluxo. Ou seja, se $Cota_i = 2Cota_j$, o ***fluxo i*** terá largura de banda disponível duas vezes maior que o ***fluxo j***, quando f_i e f_j estiverem ativos.

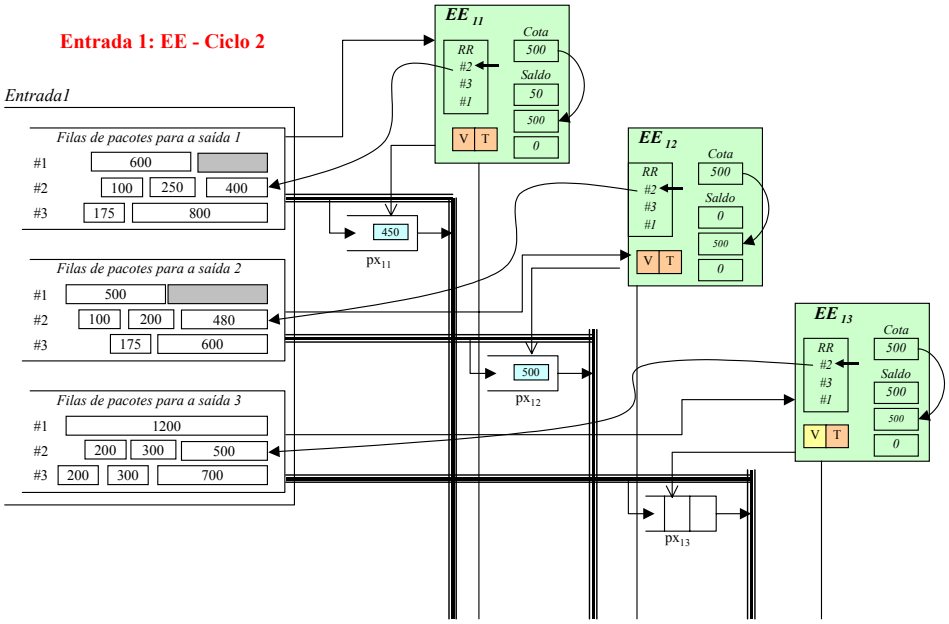


Figura 5.9 (a): Entrada 1 – Ciclo 2

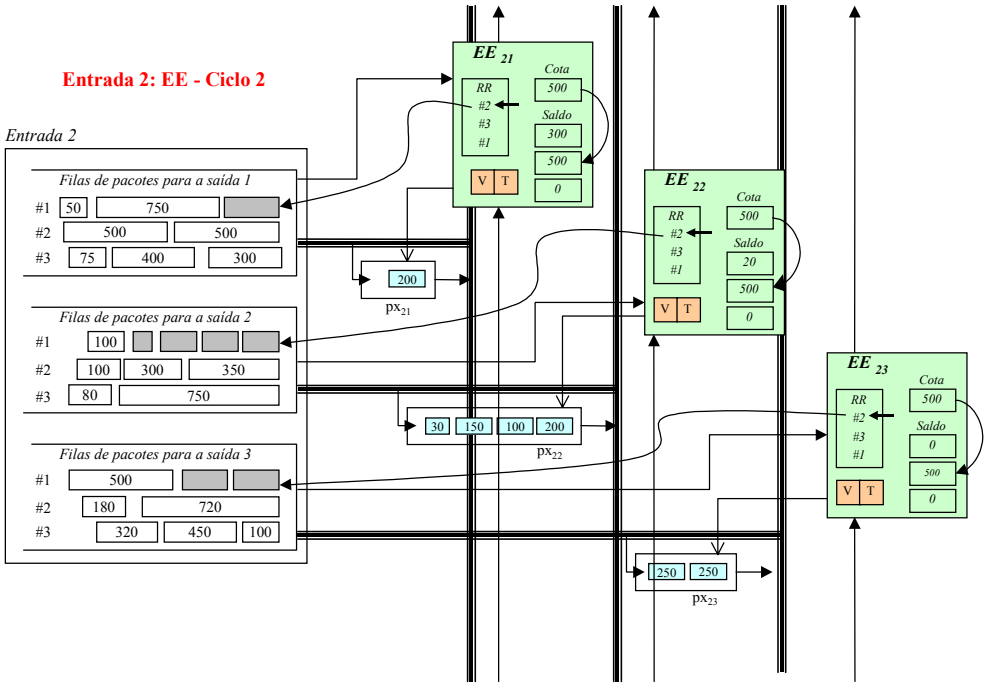


Figura 5.9 (b): Entrada 2 – Ciclo 2

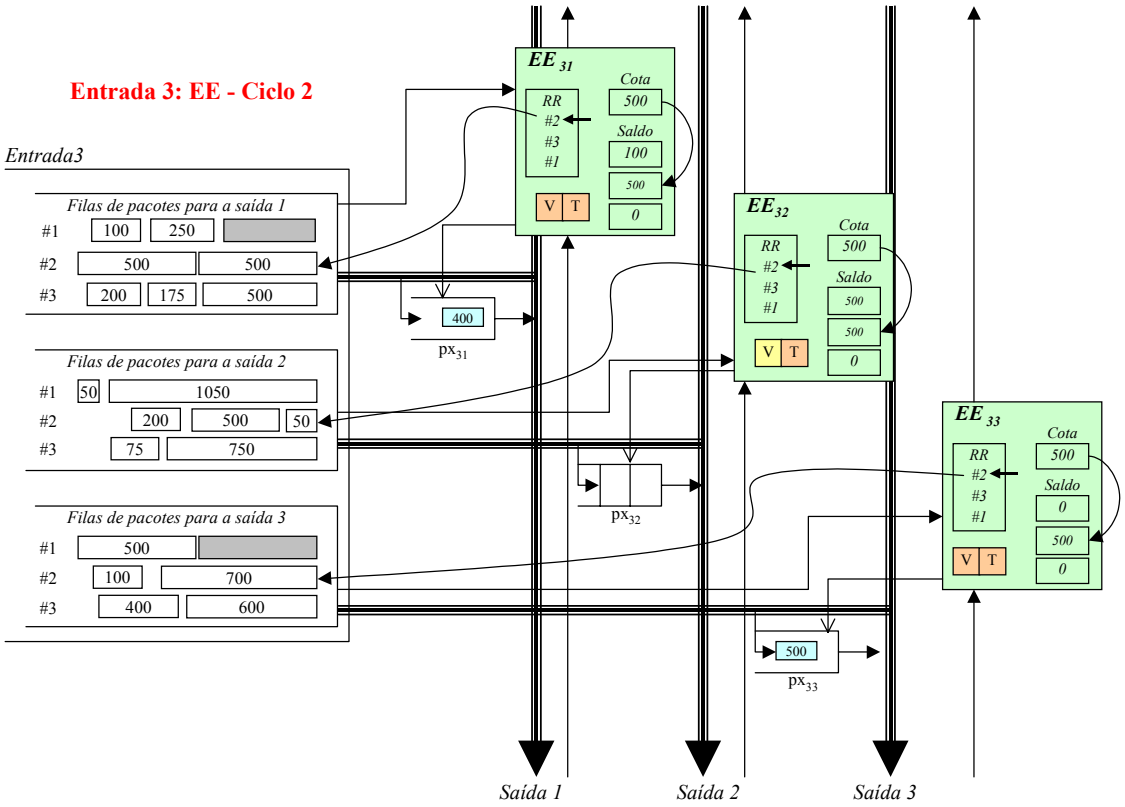


Figura 5.9 (c) Entrada 3 – Ciclo 2

Figura 5-9: *IP Switching com escalonamento distribuído usando algoritmo round-robin com déficit da Fig.5–8, no Ciclo 2 dos EEs.*

Em cada entrada, cada escalonador de cada buffer executa o algoritmo em paralelo. Assim, por exemplo, o ***EE11*** *(Fig.5.9(a)) depois de transferir para o buffer* ***px11*** *um pacote de comprimento 450 bytes, a fila #1 Entrada 1 para a Saída 1, tem 50 bytes de sobra. Como o próximo pacote na fila tem 750 bytes, a Cota não pôde ser completamente usada no ciclo atual. Então, essa quantia (50) será acrescentada a Cota do próximo ciclo, quando a fila #1 poderá enviar pacotes com comprimento total igual a 50 (saldo do ciclo anterior) + 500 (Cota). O* ***EE22*** *(Fig.5.9(b)) transfere para o buffer* ***px22*** *quatro pacotes com comprimentos 200, 100, 150 e 30 bytes (total 480 bytes) e não usa completamente sua cota pois o próximo pacote na fila #2 Entrada 2 para a Saída 2 têm 100 bytes, o que ultrapassa o limite no ciclo atual. Então, os 20 bytes restantes deverão ser acrescentados na cota do próximo ciclo. O* ***EE13*** *(Fig.5.9(a)) não transfere o pacote que espera na fila #1 Entrada 1 para a Saída 3 para o buffer* ***px13****, pois seu comprimento é de 1200 bytes o que ultrapassa a cota de 500 bytes. Nesse caso, o valor do* ***Saldo*** *500 bytes será adicionada à* ***Cota*** *do ciclo seguinte e no ciclo atual, o apontador RR passa a apontar para o próximo da fila.*

Pseudo código para o Algoritmo round-robin com déficit

Na Fig. 5.10 é apresentado um pseudo-código para o esquema round-robin com déficit. Nesse algoritmo, para um enlace de entrada qualquer, uma classe de fluxo

arbitrária e saída qualquer, $Fila_i$ representa a *i-ésima* fila, a qual armazena pacotes com identificador de fluxo ***id=i***. As filas são numeradas de **0** até **(N-1)**, onde **N** é o número de enlaces de saída.

```
  Início:
      for (i=0; 0<N; i=i+1)
            Saldoi=0;

Módulo Enfileira: na chegada do pacote p
      i= ExtraiFluxo(p)
      if(ExisteNaListaAtiva(i) == FALSE) then
            InsereListaAtiva(i); (*adiciona i na lista ativa*)
            Saldoi=0;
      Enfileira(i,p); (enfileira o pacote p na fila i)

Módulo Desenfileira:
      While (TRUE) do
            if ListaAtiva não vazia then
                  Remove cabeça da ListaAtiva,
                  Saldoi = Cotai + Saldoi;
                  while((Saldoi>0) and (Filai não vazia)) do
                        ComprimentoPacote=Size(Cabeça(Filai));
                        if(ComprimentoPacote≤Saldoi) then
                              Envia(Desenfileira(Filai));
                              Saldoi=Saldoi-ComprimentoPacotei;
                        Else break; (* sai do loop while *)
                  if (Vazia(Filai)) then Saldoi=0;
                  Else InsereListaAtiva(i);
```

Figura 5-10: *Pseudo-código para o esquema Round Robim com déficit*

Enfileira(), Desenfileira() são operadores padrões de fila. Uma lista de fluxos ativos é usada, ***ListaAtiva***, com operações padrões como ***InsereListaAtiva***, que adiciona um índice de fluxo ao final da lista. $\boldsymbol{Cota_i}$ é a cota alocado à $\boldsymbol{Fila_i}$. $\boldsymbol{Saldo_i}$ contém os bytes que a $\boldsymbol{Fila_i}$ não usou no ciclo anterior.

5.3.2 Escalonadores dos pontos de Cruzamento (EX): *Algoritmo round-robin tradicional.*

Na Fig. 5–11 está ilustrada a situação dos buffers do pontos de cruzamento (**EX**) depois que os $\mathbf{EE_{jk}}$ transferiram pacotes para os buffers $\mathbf{px_{jk}}$.

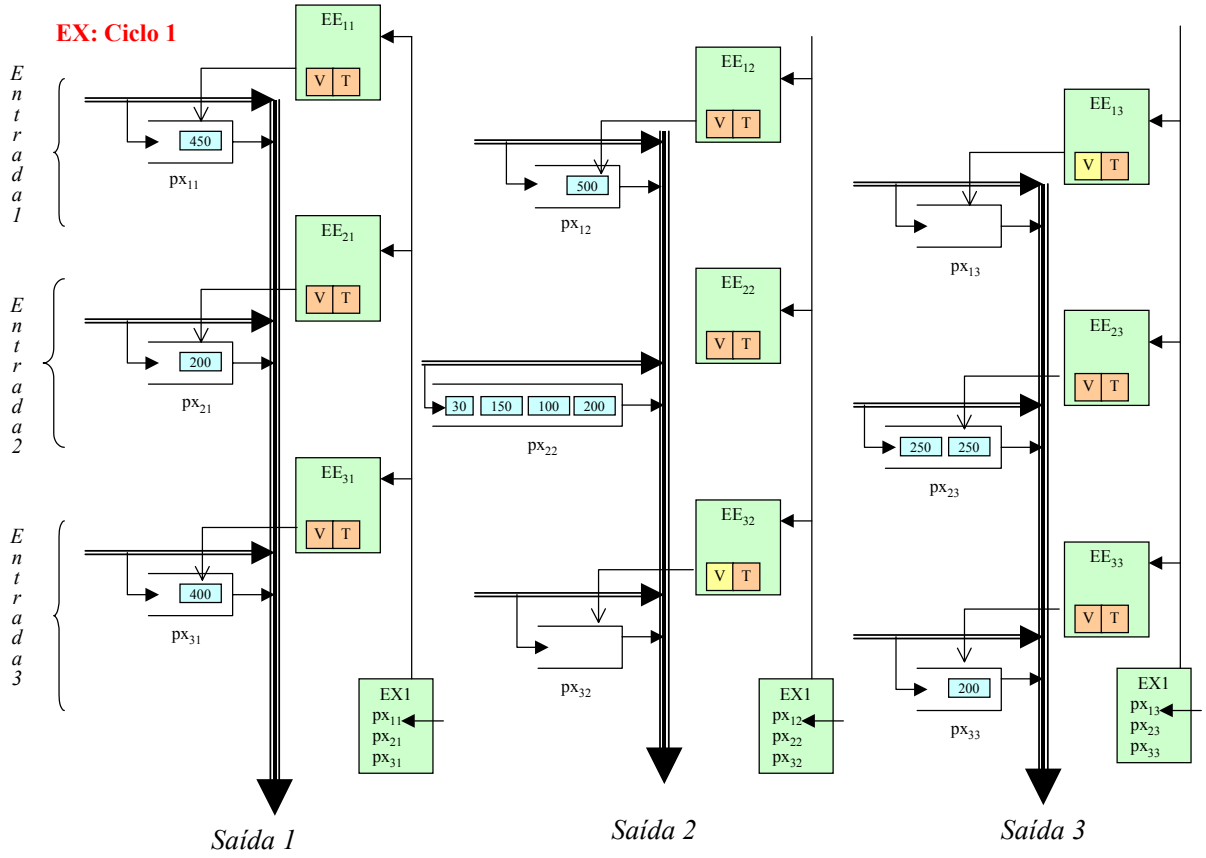


Figura 5-11: *IP Switching com escalonamento distribuído usando round-robin com déficit nos escalonadores de entrada (EE_{ij}) buffers de entrada e round-robin tradicional nos escalonadores dos pontos de cruzamento (EX_j). Ciclo 1 dos escalonadores EE_j.*

Cada escalonador ***EX*** mantém o controle do algoritmo round-robin para atendimento dos buffers dos pontos de cruzamentos. O escalonador $\boldsymbol{EX_1}$, por exemplo, examina o *flag* ***V*** de cada um dos buffers dos pontos de cruzamento ($\boldsymbol{px_{11}}$, $\boldsymbol{px_{21}}$ e $\boldsymbol{px_{31}}$) e verifica que existem pacotes a serem transmitidos. O apontador da lista round-robin de $\boldsymbol{EX_1}$ indica que o próximo buffer a ser atendido é $\boldsymbol{px_{11}}$. Paralelamente, $\boldsymbol{EX_2}$, executa algoritmo correspondente e verifica que o próximo buffer a transmitir pelo enlace de saída 2 é $\boldsymbol{px_{12}}$, e o escalonador $\boldsymbol{EX_3}$ verifica que pelo enlace de saída 3, o buffer $\boldsymbol{px_{23}}$ é o buffer autorizado a transmitir, uma vez que o buffer $\boldsymbol{px_{13}}$ apontado pelo controle *round-robin* está vazio e $\boldsymbol{px_{23}}$ é o próximo da lista. Cada buffer $\boldsymbol{px_{jk}}$ autorizado transmite seu pacote pelo enlace de saída correspondente, e o apontador da lista round-robin é atualizada em cada ***EX*** correspondente. Ao término da transmissão o *flag* **T** é alterado, sinalizando o final da transmissão e indicando ao ***EE*** correspondente que o algoritmo de transferência de pacotes dos buffers de entrada para os pontos de cruzamento deve ser executado novamente. A Fig.5.12 ilustra o estado dos buffers nos pontos de cruzamento, os *flags* **T** e os apontadores da lista de *round-robin* nos

escalonadores **EX** após o término das transmissões dos pacotes dos buffers autorizados pelos enlaces de saída correspondente.

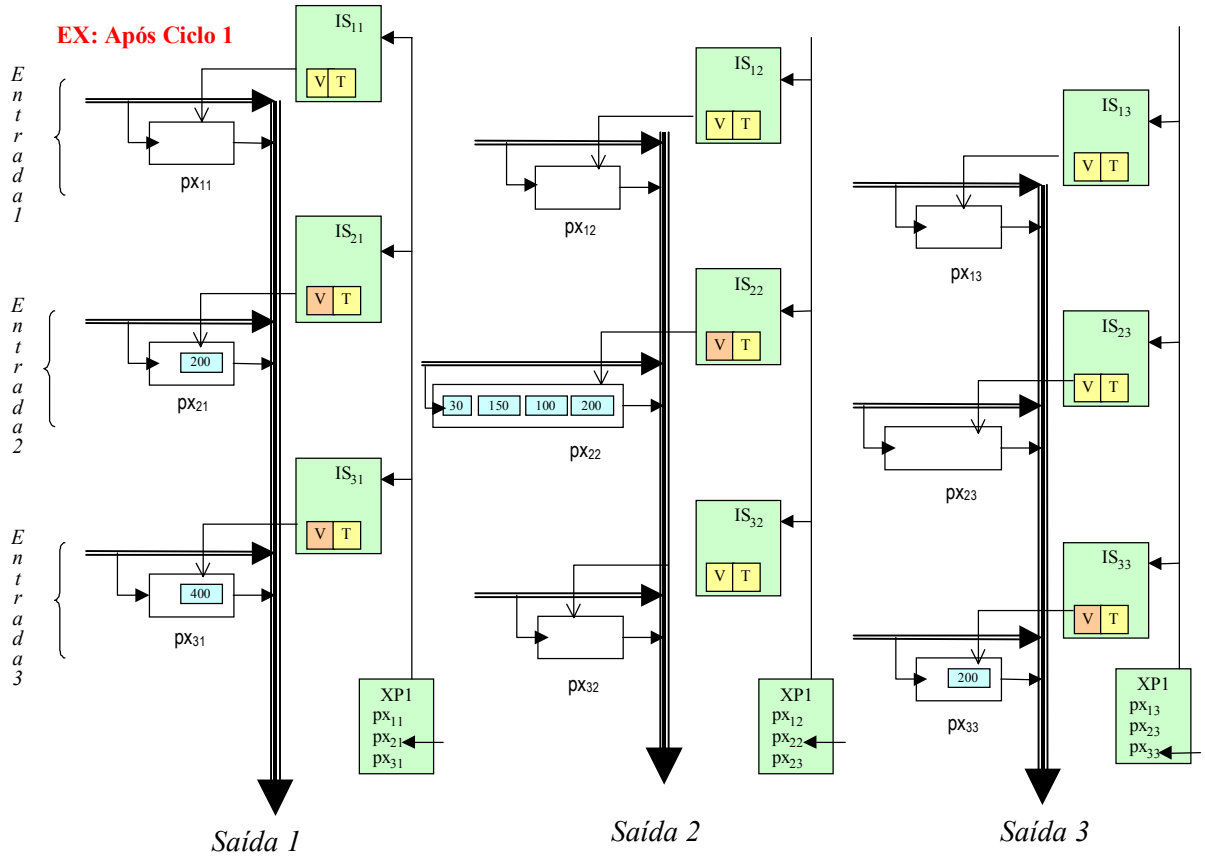


Figura 5-12: *Estrutura do IP Switching da Fig.5–11 após o primeiro ciclo dos Escalonadores dos pontos de cruzamento* $\boldsymbol{EX_{i,j}}$

Como cada escalonador de entrada e cada escalonador de saída pode rodar em paralelo, um alto processamento agregado é conseguido, o que torna a estrutura proposta capaz de atuar em alta velocidade.

Resultados teóricos do algoritmo round-robin com déficit

A proposta feita em [48] apresenta e prova algumas propriedades do algoritmo *round-robin com déficit* e as principais delas serão resumidas nessa seção, adaptadas à estrutura proposta.

Seja $fila_{ijk}$ a *fila i* da *Entrada j* para a *Saída k; Cota*$_j$ é a cota atribuída a cada fluxo da Entrada j, e $Saldo_{ijk}$ é o saldo acumulado da *Classe de Fluxo i* da *Entrada j* para a *Saída k*.

Os escalonadores dos buffers de entrada $\boldsymbol{EE_{jk}}$ atendem as filas das diferentes classes de fluxo da *Entrada j* para a *Saída k*, de acordo com o algoritmo *round-robin com déficit*. Um ciclo é uma repetição do algoritmo *round-robin* com déficit sobre as $\boldsymbol{M}$ filas, das $\boldsymbol{M}$ classes de serviço, de uma determinada entrada. Considera-se que os ciclos são numerados a partir de 1, e o início do hipotético *Ciclo 1* pode ser considerado o final de um *Ciclo 0* hipotético.

Propriedade 5.1: Para qualquer $\boldsymbol{ijk}$ e ao término de um ciclo em toda execução do algoritmo do round-robin com déficit:

$$0 <= Saldo_{ijk} < Max \qquad (5\text{–}2)$$

Prova:

Inicialmente, $Saldo_{ijk}=0 \Rightarrow Saldo_{ijk}<Cota_j$. A variável $Saldo_{ijk}$ muda de valor apenas quando a $fila_{ijk}$ é servida. Durante um ciclo, quando o serviço é completado existem duas possibilidades:

- Se um pacote permanece na $fila_{ijk}$, então ele deve ter tamanho estritamente maior que o $Saldo_{ijk}$. Por definição, o tamanho de qualquer pacote nunca é maior que *Max*, assim $Saldo_{ijk}$ é estritamente menor que *Max*. E a implementação do algoritmo garante que o $Saldo_{ijk}>=0$.
- Se nenhum pacote permanece na fila, o algoritmo atribui o valor zero a variável $Saldo_{ijk}$.

Para o caso em que apenas a $fila_{ijk}$ sempre tem saldo remanescente, ou seja, os outros fluxos podem ou não ter saldo, ou estarem ou não ativos, a diferença entre a distribuição ideal e real para a $fila_{ijk}$ sempre é limitada pelo tamanho de um pacote de tamanho máximo.

Definição 5.1: Uma classe de fluxo tem saldo em um ciclo de execução, se a $fila_{ijk}$ do *fluxo i* nunca está vazia em qualquer ponto, durante a execução.

Propriedade 5.2: Considere qualquer execução do esquema *round-robin* com déficit no qual o fluxo *i* tem saldo. Depois de quaisquer *X* ciclos, a diferença *entre* $X*Cota_j$ (ou seja, a quantidade de bytes que fluxo i poderia transferir ao buffer $\boldsymbol{px_{jk}}$) e os bytes que o *fluxo i* de fato transferiu para o buffer $\boldsymbol{px_{jk}}$ é limitado por *Max*.

Prova:.

Seja $Saldo_{ijk,x}$ o valor do saldo remanescente, para o *fluxo i* da *Entrada j* para a *Saída k*, ao término do *Ciclo x*. Seja $bytes_{ijk,x}$ os bytes enviados pelo $fluxo_{ijk}$ de um *Ciclo x*. Seja $Enviados_{ijk,x}$ os bytes enviados pelo $fluxo_{ijk}$ nos *Ciclos 1 até x*. Observe que,

$$Enviados_{ijk,X} = \sum_{x=1}^{X} bytes_{ijk,x} . \qquad (5–3)$$

Inicialmente, para qualquer ijk, $Saldo_{ijk,0}=bytes_{ijk,0}=0$. Mas, supondo que o $fluxo_{ijk}$ sempre tem saldo, $bytes_{ijk,x}+Saldo_{ijk,x}=Cota_j+Saldo_{ijk,x-1}$,. Assim, no ciclo x, a alocação total do $fluxo_{ijk}$ é $Cota_j+Saldo_{ijk,x-1}$. Assim, se o $fluxo_{ijk}$ envia $bytes_{ijk,x}$ então o valor remanescente será armazenado em $Saldo_{ijk,x}$ porque a fila nunca esvazia durante a execução. Esta equação fica reduzida à:

$$bytes_{ijk,x}=Cota_j+Saldo_{ijk,x-1}-Saldo_{ijk,x} \qquad (5–4)$$

Da Eq.5.3-2 e somando-se a Eq. 5.3-3 em X ciclos, têm-se que:

$$Enviado_{ijk,X}=X*Cota_j-Saldo_{ijk,X}. \qquad (5–5)$$

O número de bytes reservados para o $fluxo_{ijk}$ após *X* ciclos é $X*Cota_j$. Efetuando-se a subtração do Saldo, acumulado nos ciclos anteriores e usando a Eq.5.3-1 pode-se observar que essa diferença é ≤ *Max*.

A **Propriedade 5.3** a seguir, decorrente da **Propriedade 5.2** mostra que, dois fluxos que têm ***Saldos*** e ***Cotas*** iguais recebem serviços iguais.

Propriedade 5.3: Após uma quantidade qualquer de ciclos, a diferença máxima entre o total de bytes enviados por quaisquer dois fluxos com saldo, que têm **Cotas** de tamanhos iguais, é menor que *Max*.

Para as **Definição 5.2** e **Definição 5.3** seguintes, considera-se que:

- inicialmente, os pacotes são enviados aos buffers dos pontos de cruzamento no instante 0;
- o número total de bytes enviado pelo *fluxo i* , *entrada j* e *saída k*, no intervalo de tempo ***t*** é $\boldsymbol{enviado_{ijk,t}}$;
- o número total de bytes enviados por todos os ***M*** *fluxos*, da *entrada 1* para a *saída k* no intervalo de tempo ***t*** é $\boldsymbol{enviado_{jkt}}$;
- o *Quociente de Equidade* para *fluxo i*, da *entrada j* para a *saída k* é definido como a razão, de pior caso, entre os bytes enviados pelo *fluxo* ***i*** e os bytes enviados por todos os fluxos, da *entrada i* para a *saída k*.
  Isto expressa somente a "parte" do pior caso obtido pelo *fluxo* ***i***. Embora esse quociente possa ser definido para qualquer quantia de tempo ***t***, é mais natural considerar o limite quando ***t*** tende ao infinito.

Definição 5.2: $$FQ_{jk} = Max\left(\lim_{t \to \infty} \frac{envidado_{ijk,t}}{enviado_{jk,t}} \right) \qquad (5–6)$$

Na Eq. 5−6, o máximo é tomado considerando todas as possíveis distribuições de tamanhos de pacote de entrada, para todos os fluxos.

Assume-se portanto, que existe uma quantidade f_{ijk} – atribuída por um administrador – que expressa a contribuição ideal a ser obtida pelo *fluxo i, da entrada j para a saída k*. Assim o *Quociente de Equidade "Ideal"* para *fluxo i da entrada j para a saída k* é dado por $\frac{f_{ijk}}{\sum_{l=1}^{M} f_{ljk}}$.

No caso mais simples, todos os f_{ijk} são iguais e o Quociente de Equidade Ideal é $1/M$.

Definição 5.3: O *Índice de Eqüidade* para o $fluxo_{ijk}$ em uma implementação de enfileiramento imparcial é dado por:

$$Índice\ de\ Eqüidade_{ijk} = \frac{FQ_{ijk} \sum_{l=1}^{M} f_{ljk}}{f_{ijk}} \tag{5.7}$$

Para a **Propriedade 4** dada a seguir, será considerado um cenário de tráfego pesado, em que, cada entrada possui *M fluxos* continuamente ativos. Os pacotes são de tamanhos arbitrários na faixa entre *Min* e *Max*.

Propriedade 5.4: Para qualquer $fluxo_{ijk}$, o $Índice\ de\ Equidade_{ijk}$=1.

Prova:

Pela Propriedade 5.2, a diferença máxima entre a quantidade de bytes que um fluxo deveria transferir e o que foi realmente transferido é *Max*. Então, para qualquer $fluxo_{ijk}$ e qualquer *Ciclo x:*

$$X \cdot Cota_{ijk} - Max \le enviado_{ijk,\,x} \le X \cdot Cota_{ijk}$$

Agregando todos as *M* classes de fluxos da *Entrada j* para a *Saída k*, têm-se:

$$X \cdot \sum_{i=1}^{M} Cota_{ijk} - M \cdot Max \le Enviado_X \le X \cdot \sum_{i=1}^{M} Cota_{ijk}$$

Então, tomando-se a razão entre as duas expressões anteriores:

$$\frac{X \cdot Cota_{ijk} - Max}{X \cdot \sum_{i=1}^{M} Cota_{ijk}} \le \frac{Enviado_{ijk,x}}{Enviado_X} \le \frac{X \cdot Cota_{ijk}}{X \cdot \sum_{i=1}^{M} Cota_{ijk} - M \cdot Max}$$

No limite, quando *t* tende ao infinito, o número de *Ciclos é X*. Quando X tende ao infinito tem-se que:

$$\frac{Cota_{ijk}}{\sum_{i=1}^{M} Cota_{ijk}} \leq FQ_{ijk} \leq \frac{Cota_{ijk}}{\sum_{i=1}^{M} Cota_{ijk}}, \text{ o que implica em:}$$

$$FQ_{ijk} = \frac{Cota_{ijk}}{\sum_{i=1}^{M} Cota_{ijk}} \tag{5.8}$$

Uma vez $f_{ijk}=Cota_{ijk}$ têm-se que:

$$\text{Índice de Eqüidade} = {}_{ijk} = \frac{FQ_{ijk} \sum_{l=1}^{M} f_{ljk}}{f_{ijk}} = 1$$

Propriedade 5.5: *A complexidade do esquema round-robin com déficit é O(1), se para todo fluxo* ***i****, da entrada j para a saída k,* $Cota_{ijk} \geq Max$.

Prova:

Enfileirar um pacote requer, determinar a fila usada pelo fluxo e adicionar o pacote no final da fila. Se for utilizado *hashing* para determinar a fila, ignorando as colisões, a complexidade desse algoritmo é dada por O(1). Retirar um pacote da fila requer, determinar a próxima fila para o serviço examinando o cabeçalho da lista ativa (*ListaAtiva*), e então realizar um número constante de operações, por pacote enviado da fila, afim de atualizar o *Saldo* da lista ativa. Se a *Cota* ≥ *Max*, garante-se que ao menos um pacote será enviado cada vez que uma fila é visitada, e então no pior caso, a complexidade é O(1).

Demonstrou-se, na análise anterior, que se as colisões introduzidas pelo *hashing* são ignoradas, o esquema *round-robin com déficit* tem *Índice de Equidade* próximo de 1 e *Complexidade igual a O(1)*. Se o esquema *round-robin com déficit* for usado com *hash*, o *Índice de Eqüidade* deve ser ajustado pelo número médio de colisões. O número médio colisões entre um determinado fluxo com outros quaisquer é *M*/*Q*, onde *M* é o número de fluxos e *Q* é o número de filas. Se B é a largura de banda reservada a um fluxo, a largura da banda efetiva é dada por B/(1+M/Q). Assumindo uma escolha satisfatória para o tamanho da *Cota* e um número razoável de filas o esquema global alcança *Índice de Equidade* próximo de 1 e *Complexidade* igual à O(1).

Análise de desempenho da estrutura IP Switching com escalonamento distribuído usando round-robin com déficit.

Para avaliar o desempenho da estrutura proposta foi desenvolvido um modelo de simulação usando a ferramenta *AweSim*.

No modelo de simulação desenvolvido para avaliar o desempenho da estrutura de *IP Switching* com escalonamento distribuído foi considerado uma estrutura com três entradas e três saídas. Os pacotes que chegam à estrutura são distribuídos em três classes de fluxos f_1, f_2 e f_3. A classe de fluxo f_1 tem pacotes com comprimento médio igual a 3000 bits e representa 18% do total de pacotes de entrada; a classe de fluxo f_2 tem pacotes com comprimento médio igual a 2000 bits e representa 27% do total de pacotes de entrada e a classe de fluxo f_3 tem pacotes com comprimento médio igual a 1000 bits e representa 55% do total de pacotes de entrada. Considerou-se ainda que, as cotas usadas em cada entrada são iguais para as três classes de fluxo, mas são diferentes em cada entrada. Assim, os escalonadores da *Entrada 1* usam *Cota* de 3000 bits, os escalonadores da *Entrada 2* usam *Cota* de 2000 bits e os escalonadores da *Entrada 3* usam *Cota* de 1000 bits. A capacidade do enlace de saída é de 625,58 Mbps.

A curva obtida para a média do Tempo de trânsito dos pacotes no Sistema, em μs, em função da carga para uma dada saída é apresentada nas Fig. 5–13 (a) e Fig. 5–13 (b).

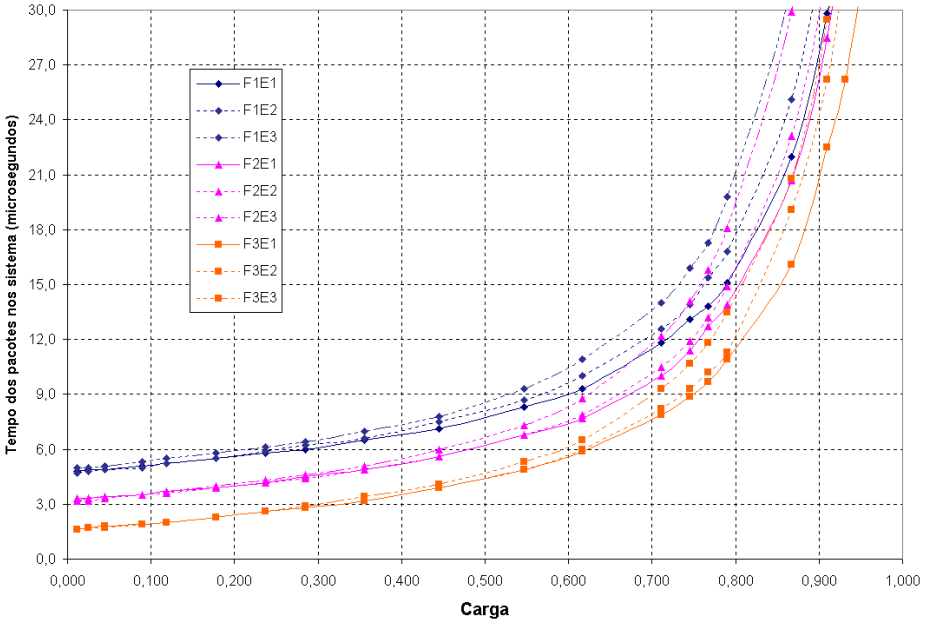


Figura 5.13(a)

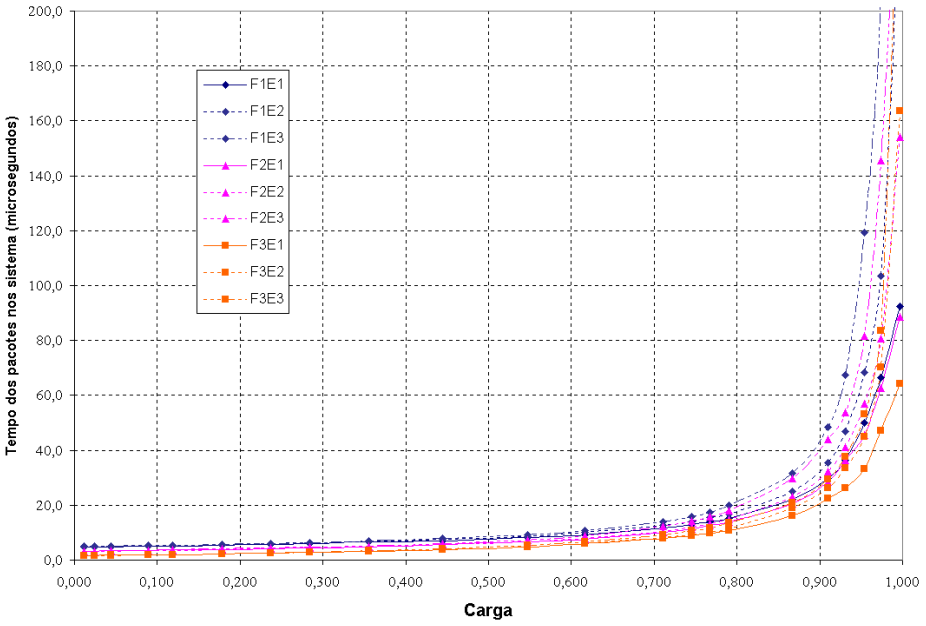


Figura 5.13 (b)

Figura 5-13: *Desempenho da Estrutura IP Switching usando escalonamento distribuído, com o algoritmo round-robin com déficit nos buffers de entrada e round-robin tradicional nos buffers dos pontos de cruzamento.*
Capacidade do enlace: 625,58 Mbps.
Fluxo FiEj: Fluxo f_i da entrada E_j.

Fluxo		*Cota (bit)*	*Comprimento do Pacote (bit)*	*Porcentagem de Fluxo*
F1E1	*Fluxo 1 Entrada 1*	*3000*	*3000*	*18%*
F1E2	*Fluxo 1 Entrada 2*	*2000*		
F1E3	*Fluxo 1 Entrada 3*	*1000*		
F2E1	*Fluxo 2 Entrada 1*	*3000*	*2000*	*27%*
F2E2	*Fluxo 2 Entrada 2*	*2000*		
F2E3	*Fluxo 2 Entrada 3*	*1000*		
F3E1	*Fluxo 3 Entrada 1*	*3000*	*1000*	*55%*
F3E2	*Fluxo 3 Entrada2*	*2000*		
F3E3	*Fluxo 3 Entrada 3*	*1000*		

Nessas figuras estão representadas as curvas para o Tempo de Serviço, de cada classe de fluxo, em cada uma das entradas. Observa-se que, quando a carga total é baixa, o tempo médio no sistema é maior para a classe de fluxos com pacotes de comprimentos médio maiores, devido ao maior tempo necessário para transmissão do pacote; entretanto, não existe diferença considerável de tempo no sistema, entre os pacotes de uma mesma classe de fluxo em cada uma das entradas, cada qual com um valor diferente de *Cota*. Isso acontece, porque em condições de baixa carga, o *Saldo* necessário para enviar pacotes maiores que a *Cota* é rapidamente acumulado, uma vez

que a taxa de chegada de pacotes nas linhas de entrada é baixa. Porém, à medida que a carga torna-se maior, Fig. 5–13 (b), as classes de fluxo com pacotes menores ou iguais às *Cotas* passam a ter menor Tempo no Sistema que aquelas com pacotes maiores que as *Cotas*. Isso indica que, a estrutura para *IP Switching* com escalonamento distribuído, usando o esquema *round-robin com déficit* nos escalonadores dos *buffers* de entrada e *round-robin tradicional* nos *buffers* dos pontos de cruzamento promove a eqüidade no uso da banda de passagem dos enlaces de saída.

A curva apresentada na Fig. 5–14 ilustra a quantidade de bits transmitidos para cada uma das classes de fluxos de cada entrada, obtida pela simulação. Para um total de 50.000 pacotes transmitidos com médias de comprimentos de pacotes e taxa de distribuição entre as classes de fluxo determinadas pelo modelo de simulação e apresentados anteriormente, a curva da Fig. 5.14 mostra o total de bits transmitidos, representados em mega bits, para cada uma das classes de fluxo f_1, f_2 e f_3.

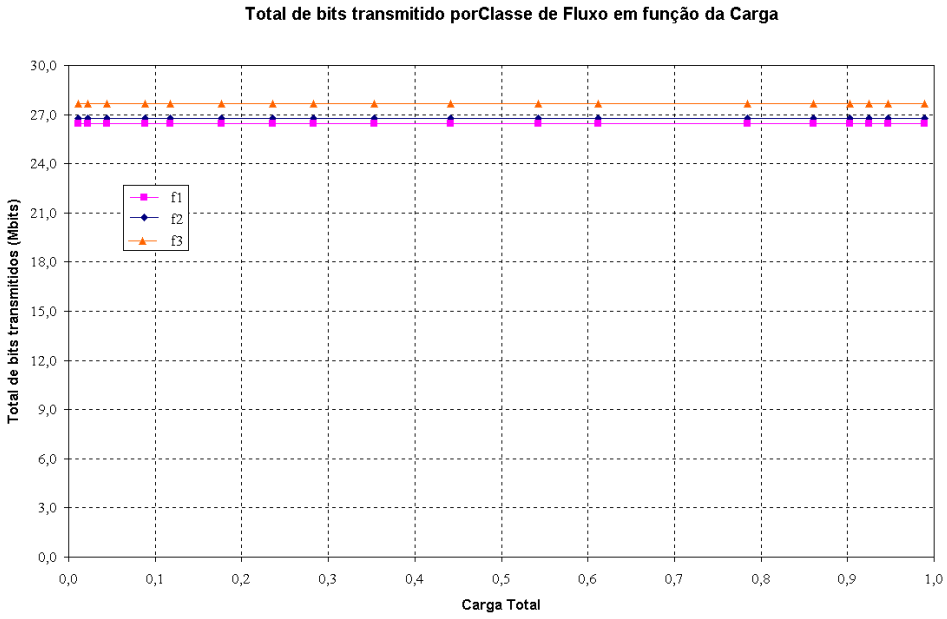


Figura 5-14: *Quantidade de bits transmitidos (em Mbits) para cada classe de fluxo na Estrutura IP Switching, algoritmo round-robin com déficit nos buffers de entrada e algoritmo round-robin tradicional nos buffers dos pontos de cruzamento em função da Carga total.*

Observa-se que, para a classe de fluxo f_3, o comprimento médio dos pacotes é menor que a *Cota* adicionada ao *Saldo* em cada ciclo do algoritmo *round-robin com déficit*, o que proporciona uso mais efetivo da largura de banda, pois, em cada ciclo em que essa classe de fluxo é selecionada, pelo menos um pacote é enviado ao buffer do ponto de cruzamento. Os pacotes da classe de fluxo f_1 têm comprimento médio maior que a *Cota*

usada pelos escalonadores dos buffers nas entradas 2 e 3, enquanto que os pacotes da classe de fluxo f_2 têm comprimento médio maior que a *Cota* usada pelos escalonadores dos buffers na entradas 3, tendo portando menor uso efetivo da banda de passagem.

Na Fig. 5–15 estão representadas as curvas da Banda de Passagem (em Mbps) utilizada para cada classe de fluxo f_1, f_2 e f_3 em função da carga total. Também nessa figura, observa-se que o fluxo f_3 usa maior largura de banda em função da adequação do comprimento médio dos pacotes com as *Cotas* adotadas para a estrutura modelada.

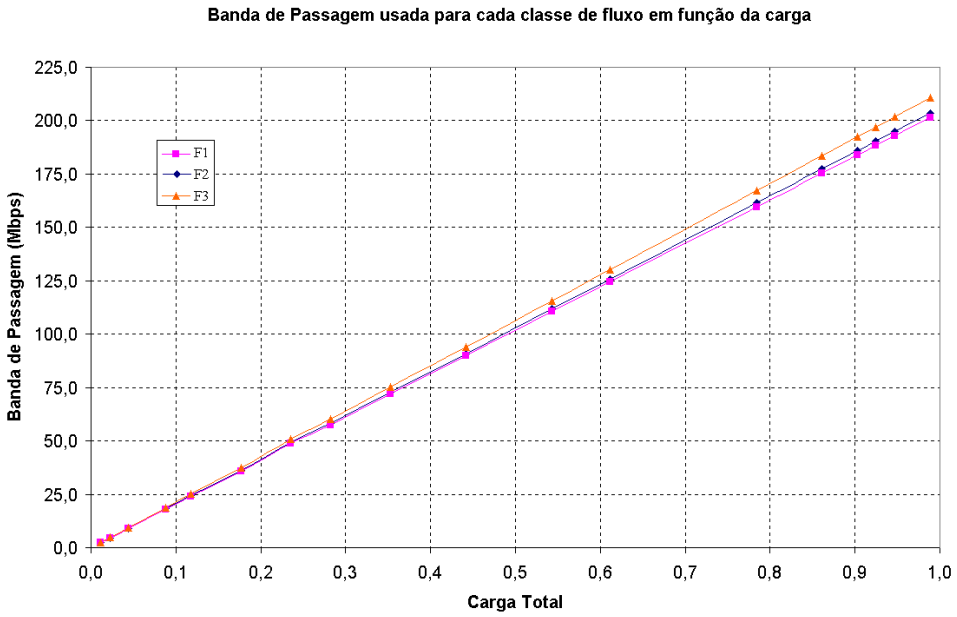


Figura 5-15: *Banda de Passagem usada para cada classe de fluxo na estrutura IP Switching com escalonamento distribuído, algoritmo round-robin com déficit nos buffers de entrada e algoritmo round-robin tradicional nos buffers dos pontos de cruzamento.*

A curva da utilização do enlace de saída em função da carga, obtida através de simulação, está apresentada na Fig. 5–16. Observa-se nessa curva, que a utilização do enlace é proporcional à carga nos enlaces de entrada, e que para condição de carga de 100%, o enlace de saída tem utilização plena.

Na Fig. 5–17 é apresentada a vazão da estrutura *IP Switching* obtida pela simulação realizada. As curvas mostram a razão dos pacotes que chegaram nos enlaces de entrada da estrutura, com destino a uma dada saída, e os pacotes que foram transmitidos durante o tempo de simulação, de acordo com a variação da carga. Pode-se observar que a estrutura mantém vazão de 100% para qualquer situação de carga, para qualquer uma das classes de fluxo.

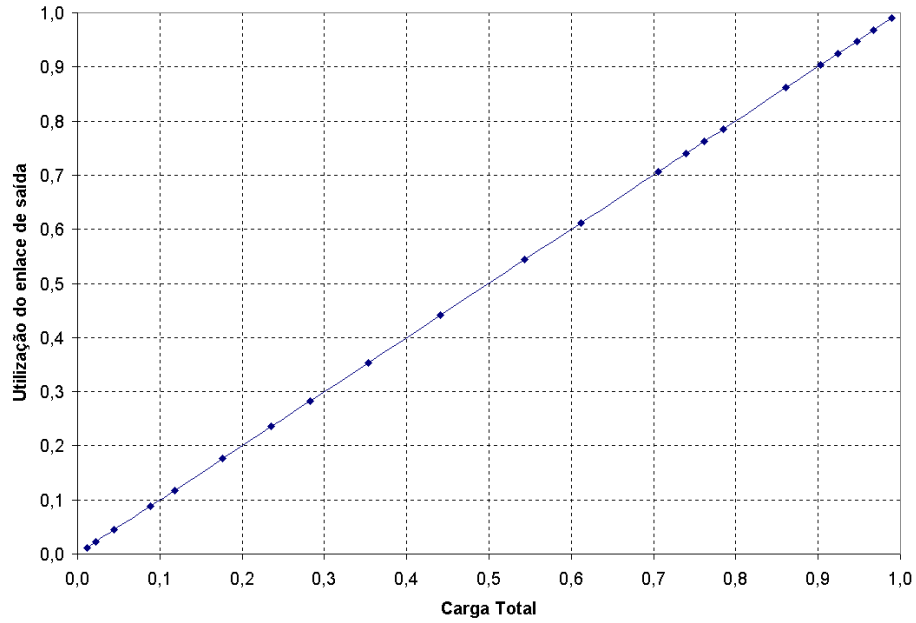


Figura 5-16: *Utilização do enlace de Saída em função da Carga Total na estrutura IP Switching com escalonamento distribuído, usando o algoritmo round-robin com déficit nos buffers de entrada e round-robin tradicional nos buffers dos pontos de cruzamento.*

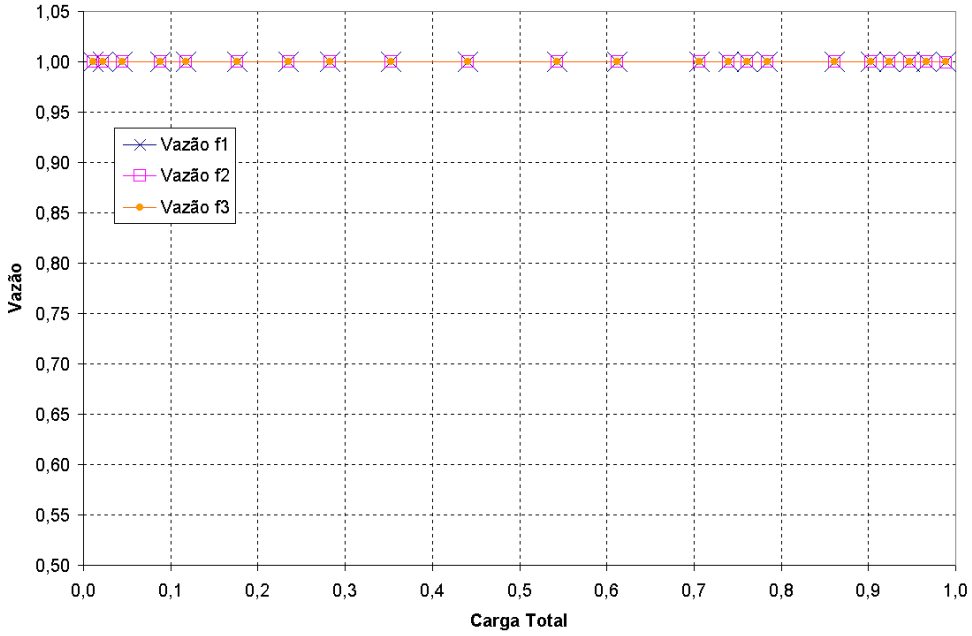


Figura 5-17: *Vazão para cada classe de fluxo, em função da carga, da estrutura IP Switching com escalonamento distribuído, usando o algoritmo round-robin com déficit nos buffers de entrada e round-robin tradicional nos buffers dos pontos de cruzamento.*

5.4 Conclusão

Neste capítulo foram apresentadas duas propostas de estruturas de IP Switching usando escalonamento distribuído.

A primeira proposta utiliza escalonadores baseados em prioridade das classes de fluxos e é capaz de garantir QoS. Porém o algoritmo não garante a equidade no uso da largura de banda, se uma das classes de fluxo usa pacotes grandes enquanto outras classes de serviço usam pacotes pequenos.

A segunda proposta utiliza escalonadores distribuídos baseados no esquema round-robin com déficit que garante equidade no uso da largura de banda do meio de transmissão. Este algoritmo, a princípio, não prevê a priorização de classes de fluxo, e portanto não implementa facilidades para garantir QoS, porém ele pode facilmente ser adaptado para atribuir maiores quantidade de largura de banda à classes de tráfego de maior prioridade, por exemplo, atribuindo valores diferenciados de Cotas para as diferentes classes de fluxo.

Capítulo 6

6 Conclusão

Este trabalho apresentou propostas de estruturas de comutação de alto desempenho e algoritmos de escalonamento de pacotes, que se mostraram flexíveis e eficientes para serem usadas em redes de transmissão de alta velocidade.

Inicialmente, foram apresentadas três estruturas de comutação ATM, que utilizam a combinação de *buffers* nas entradas com um conjunto de cinco *buffers* em cada ponto de cruzamento, um *buffer* para cada tipo de serviço. A diferença principal entre elas é a quantidade de caminhos paralelos no barramento que liga os *buffers* de entrada aos *buffers* dos pontos de cruzamentos. O esquema de encaminhamento adotado favorece as células de serviços prioritários.

O desempenho das três estruturas apresentadas no Capítulo 3 foi analisado, através de um estudo de caso. Verificou-se que, a Estrutura 3 apresentou melhor desempenho e priorizou efetivamente o tráfego com maior exigência de QoS; a Estrutura 1, de menor complexidade, apresentou o menor desempenho; e a Estrutura 2, de complexidade intermediária, apresentou desempenho compatível com a Estrutura 3, porém não priorizou efetivamente o tráfego mais exigente.

A Estrutura 3 utiliza barramentos dedicados e transmite em paralelo células de uma entrada para cada *buffer* de ponto de cruzamento de cada tipo de serviço. Na adoção dessa estrutura em uma rede de alta velocidade, ou como assunto de trabalhos futuros, a quantidade de barramentos paralelos poderia ser variada. Assim, por exemplo, os serviços menos prioritários poderiam transferir poucas células, ou mesmo uma única célula, em vez de transmitir em paralelo uma célula a cada *buffer* de ponto de cruzamento de cada tipo de serviço. A quantidade de células que seriam transferidas em paralelo aos buffers dos pontos de cruzamento poderiam ser especificadas usando modelos de simulação, assumindo-se compromisso entre desempenho e complexidade do comutador.

Para realizar a análise de desempenho da Estrutura 3, um modelo matemático foi apresentado. Os resultados da análise mostraram que essa estrutura obteve desempenho adequado para garantir a QoS exigida pelas classes de serviços previstas no ATM. Para cargas de 90% e maior concentração de tráfego nas classes de serviço de

maior prioridade (40% de CBR, 20% de rtVBR e 20% de nrtVBR), as células mais prioritárias (CBR e rtVBR) sofrem atraso máximo de três vezes o tempo necessário para transmitir uma célula ($3xT_{slots}$). Apenas as células menos prioritárias (ABR e UBR) sofrem grandes atrasos, e mesmo assim para cargas maiores 80%. Quando 60% do tráfego estiver distribuído entre as classes de serviço de menor prioridade, observou-se que os serviços CBR, VBR e ABR sempre têm atrasos pequenos (menores que $3xT_{slots}$) para qualquer situação de carga, e que o serviço UBR tem atraso considerável apenas para carga superior a 95%.

A Estrutura 3 foi generalizada em uma nova proposta, no Capítulo 4, de forma que fosse possível desvincular os algoritmos de encaminhamento de células da estrutura do comutador. Essa nova proposta, também utiliza buffers grandes nas entradas e buffers pequenos nos pontos de cruzamento, porém utiliza escalonamento de células distribuído. Pela discriminação das células de entrada de acordo com a classe de serviço e usando uma modificação do algoritmo de enfileiramento de saída virtual (VOQ *Virtual Output Queued*), o algoritmo de escalonamento de células pôde ser distribuído nos buffers de entrada e nos buffers dos pontos de cruzamento. Esta estrutura tem vazão teórica de 100%.

O comutador ATM com escalonamento de células distribuído, como foi chamada a estrutura apresentada no Capítulo 4, também foi modelado matematicamente e baseado nesse modelo e em resultados de simulação, foi feita a análise de desempenho. O modelo matemático e a simulação consideraram que as células que chegam em cada porta de entrada obedece à distribuições de Bernoulli idênticas e independentes. Os resultados obtidos com a simulação mostraram que o modelo matemático é adequado e pode ser usado para analisar a estrutura proposta. Os resultados da análise de desempenho mostraram que a estrutura é adequada para ser usada em redes de alta velocidade. Em trabalhos futuros, o comportamento do comutador com escalonamento distribuído de células poderá ser avaliado considerando outros padrões de tráfego de entrada.

Os comutadores ATM são empregados na infra-estrutura da Internet, quando é necessário obter velocidade extremamente alta, ou quando se deseja atender diferentes requisitos de QoS. Entretanto, para comutar pacotes de comprimento variável, que é o caso dos pacotes IP, é necessário processamento adicional, a fim de segmentar os pacotes na entrada do comutador e remontá-los na saída. No Capítulo 5 foram apresentadas duas propostas de comutadores de IP (*IP Switching*) que são capazes de comutar pacotes de comprimento variável, sem a necessidade de segmentação/remontagem.

A primeira proposta para *IP Switching* utiliza escalonadores baseados em prioridade das classes de fluxos e é capaz de garantir QoS. Porém o algoritmo não garante a equidade no uso da largura de banda. Se um determinado fluxo IP usa pacotes arbitrariamente grandes, enquanto outros fluxos usam pacotes pequenos, aquele que usa pacotes grandes obtém maiores fatias da largura de banda. Para o caso de fluxos de maior prioridade com pacotes longos, a possibilidade de ocorrer a inanição (*starvation*) dos fluxos de menor prioridade aumenta se eles usam pacotes menores.

A segunda proposta *IP Switching* utiliza escalonadores distribuídos baseados no esquema round-robin com déficit, que garante equidade no uso da largura de banda do meio de transmissão. Este algoritmo, em princípio, não prevê a priorização de classes de fluxo, e portanto não implementam facilidades para garantir QoS, porém ele pode ser facilmente adaptado para atribuir maiores quantidade de largura de banda à classes de tráfego de maior prioridade.

Em trabalhos futuros serão considerados variações do algoritmo round-robin com déficit, para que ele inclua facilidades de garantir QoS. Algumas das modificações que podem ser pesquisadas são: a atribuição de valores diferenciados de Cotas para os diferentes fluxos e o estabelecimento de ciclos de round-robin ponderados, em que os fluxos com maior prioridade são servidos mais vezes, durante um mesmo ciclo.

Referências Bibliográficas

[1] STALLINGS, W. "ISDN and Broadband ISDN with Frame Relay and ATM", *Prentice Hall, Inc.,* Third Edition, chaps. 15-16, pp. 408-516, 1995.

[2] TANENBAUM, A. S., "Computer Networks", *Prentice Hall, Inc.,* Third Edition, chap. 2, pp. 147-155, 1996.

[3] XIAO, X. NI, L.M. e VUPPALA, V. "An Overview of Ip Switching and Tag Switching", *Proceedings on Parallel and Distributed Systems, International Conference on*, pp. 669 -675, 10-13 Dez 1997.

[4] NEWMAN, P.; EDWARDS, W. et al.; "Ipsilon's General Switch Management Protocol Specification, Version 1.1", RFC1987, 1996

[5] NEWMAN, P.; EDWARDS, W. et al.; "Ipsilon's Flow Management Protocol Specification for IPv4, Version 1.0", RFC1953, 1996

[6] REKHTER, Y.; DAVIE, B. et al.; "Cisco Systems' Tag Switching Architecture Overview", RFC2105, 1997

[7] Faucheur, F.; "IETF Multiprotocol Label Switching (MPLS) Architecture" *Le ATM, 1998. ICATM-98., 1998 1st IEEE International Conference on*, pp. 6 -15, 22-24 Jun -1998

[8] MAMAIS, G.; MARKAKI, M.; POLITIS, G.; VENIERIS, I.S "Efficient buffer management and scheduling in a combined IntServ and DiffServ architecture: a performance study"; *ATM, 1999. ICATM-99. 1999 2nd International Conference on,* pp. 236 -242, 1999

[9] BRADEN, R.; ZHANG, L.; BERSON, S.; HERZOG, S. e JAMIN, S., "Resource ReSerVation Protocol (RSVP) - Version 1 - Functional Specification", RFC2205, Set - 1997

[10] NICHOLS, K.; BLAKE, S.; "Differentiated Services Operational Model and Definitions", *Internet Draft*, Fev. 1998.

[11] BLACK, D.; BLAKE, S. CARLSON, M. DAVIES, E.; WANG, Z. e WEISS, W.; "An Architecture for Differentiated Services", *Internet Draft*, Maio 1998.

[12] CARPENTER, B.E.; NICHOLS, K.; "Differentiated services in the Internet"; *Proceedings of the IEEE, Volume: 90 Issue: 9*, pp. 1479 -1494; Sep 2002

[13] BRADNER, S.; MANKIN, A. "The Recomendations for the IP Next Generation Protocol", RFC175, Jan 1995

[14] ARANTES, M.P.C; MOTOYAMA, S. "Propostas de Estruturas de Comutador para Redes de Comunicação de alta velocidade"; *Revista de Informática*; Vol III; N° 4; Outubro de 2000; pp 07 a 28; Faculdade de Valinhos; 2000. Artigo também publicado em *Proc. do CACIC Congresso Argentino de Ciência da Computação* realizado em outubro de 2000.

[15] ARANTES, M.P.C.; MOTOYAMA, S. "Análise de desempenho de um comutador ATM com facilidade para garantir qualidade de serviço*"; Proc. do IX Simpósio Brasileiro de Telecomunicações,* Fortaleza – CE, Setembro de 2001.

[16] ARANTES, M.P.C. ; MOTOYAMA, S. "An ATM Switch with Distributed Cell Scheduling*"; Proc. of ITS2002 - International Telecommunications Symposium*; Natal – RN, 08-12/09/2002

[17] MOTOYAMA, S.; ARANTES, M.P.C. "An IP Switching with Distributed Scheduling"; *IEE ELECTRONICS LETTERS,* Vol. 38 No. 8 pp 392-393, 11th April 2002

[18] ATM Forum/af-tm-0056.000 - ATM Forum Traffic Management Speacification Version 4.0 - ATM Forum Technical Committee - Traffic Management Working Group - April 1996

[19] VERMA, S, "ATM Switch Architetures", *A Report in The Departament of Electrical and Computer Engineering*, Concordia University, Montréal, Québec, Canada, Agosto de 1994.

[20] KAROL, M.; HLUCHYJ, M. e MORGAN, S. P. "Input versus output queuing in a space division switch", *IEEE Trans. Commun.*, vol. COM-35, pp.1347-1356, Dec. 1987.

[21] MOTOYAMA, S; PETR, D. W. e FROST, V. S. "Input-queued switch based on a scheduling algorithm", *Electron. Lett.*, vol. 31, nº 14, pp. 1127-1128, July 1995

[22] MCKEOWN, N.; VARAIYA, P. e WALRAND, J. "Scheduling cells in an input-queued switch", *Electron. Lett.*, vol. 29, nº 25, pp. 2174-2175, December 1993

[23] MCKEOWN, N.; ANANTHARAM, V. e WALRAND, J. "Achieving 100% throughput in an input-queued switch", in Proc. IEEE INFOCOM'96, San Francisco, CA, Mar. 1996

[24] S. MOTOYAMA, L. M. ONO, e M. C. MAVIGNO, "An interative Cell Scheduling Algorithm for ATM Input-Queued Switch with Service Class Priority" in *IEEE Communications Letters*, vol. 03, nº 11, pp. 323-325, November 1999.

[25] LAMAIRE, R. O.; SERPANOS, D. N. "Two-Dimensional Round-Robin Schedulers for Packet Switches with Multiple Input Queues. *IEEE/ACM Transactions on Networking*, v.2, n5, p.471-482, October 1994.

[26] THOMAS, G. "Bifurcated Queueing for Throughput Enhacement in Input-Queued Switches. *IEEE Communications Letters*, v.1, n.2, March 1997.

[27] ANDERSON, T. R.; OWICKI, S.S.; SAXE,J.B. "High Speed Switch Scheduling for Local Networks". ACM Transactions on Computer Systems, v.11, n.4, p.319-352, November 1993.

[28] ONO, L M "Estudo de Desempenho de Algoritmos de Encaminhamento de Células em Comutadores ATM com Buffer de Entrada." Campinas, SP, *Dissertação de Mestrado*, FEEC - UNICAMP - Setembro 1998.

[29] OBARA, H; YASUSHI, T; "An Efficicient Contention Resolution Algorithm for Input Queueing ATM Cross-Conect Switches" *International Jour. of Digital & Analog Cabled Systems*, Vol..2, nº4, pp261-267, Outubro a Dezembro de 1989.

[30] OBARA, H; "Optimum Architecture For Input Queuing ATM Switches", *Elect. Lett.*, pp 555-557, Março de 1991.

[31] KAROL, M.; ENG, K.; OBARA, H. "Improving the performance of input-queud ATM packet switches". *Eleventh Annual Joint Conference of the IEEE Computer and Commun. Soc. ; INFOCOM'92*; Maio 1992.

[32] DEVAULT, M.; COCHENNEC, J. Y. e SERVEL, M. "The Prelude ATD Experiment: Assessments and Future Prospects" *IEEE Journal on Selected Areas in Commun.* Vol. 6, nº 9, pp1528-1537, Dezembro 1988.

[33] KUWAHARA, H.; ENDO, N.; KOZAKI, T. "A Shared Buffer Memory Switch for an ATM Exchange", *Proc. IEEE International Conf. Commun.*, Vol 7, pp1091-1103, 1989.

[34] ENDO, N.; KOZAKI, T.; OHUCHI, T.; KUWAHARA, H.; GOHARA, S. "Shared Buffer Memory Switch for an ATM Exchange", *IEEE Trans. on Commun.* , Vol. 41, nº 1, pp. 237-245, Janeiro 1993.

[35] YEH, Y. S; HLUCHY, M. G.; ACAMPORA, A. S.; A. S. "The knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching", *IEEE Journal on Selected Areas in Commun.* , vol. 5, pp.1274-1283, Outubro de 1987.

[36] ENG, K. Y.; KAROL, M. J. e YEH, Y. S. "A Growable Packet (ATM) Switch Architeture: Design Principles and Applications", *IEEE Trans. Comm.*, Vol. 40, nº 2, pp. 423-430, February 1992

[37] MCKEOWN, N.; KESIDIS, G. "Output-Buffer ATM Packet Switching for Integrated-Services Communication Networks" *Proc. ICC 97 - IEEE International Conf. on 'Towards the knowledge Millennium'*, Volk 3, pp 1684-1688, Junho de 1997.

[38] NOJIMA, S.; TSUTSUI, E. e HASHIMOTO, M. "Integrated Devices Packet Networking Using Bus Matrix Switch" *IEEE Journal on Selected Areas in Commun.* Vol. 5, pp1284-1292, 1987

[39] SUH, J-J; JUN, C-H. "Performance analysis of the knockout switch with input buffers". *IEE Proc. Commun.*, Vol. 141, nº3, pp 183-189, Junho 1994.

[40] GENDA K., N. YAMANAKA e Y. DOI, K. ENDO, "A High-Speed ATM Switch that uses a Simple Retry algorithm and Small Input Buffers", in *IEICE Trans. Commun.*, vol. E76-B, pp.726-730, July 1993

[41] GENDA K., Y. DOI, K. ENDO, e N. YAMANAKA, "A Very -High-Speed ATM Switch Architecture Using Internal Speed-up Technique", in *NTT Review.*, vol. 9, nº 2, pp.20-27, March 1997

[42] MOTOYAMA, S. "Simple high speed ATM switch with service class priority", *Electron. Lett.*, vol. 36, nº 6, pp. 590-591, March 2000.

[43] NABESHIMA, M. "Performance Evaluation of a Combined Input- and Crosspoint-Queued Swich", *IEICE Trans. COMMUN.*, vol. E83-B, nº.3, pp737-741, March 2000.

[44] DOI, Y.; YAMANAKA, N. "A high speed ATM switch with Input Cross-Point Buffers", *IEICE Trans. COMMUN.*, vol. E76-B, nº.3, pp310-314, March 1993.

[45] PRITSKER, A. A. B; O'REILLY, J. J.; LAVAL, D. K.; "Simulation with Visual SLAM and AweSim"; John Wiley & Sons- New York; Systems Publishing Corporation – Indiana; 1996.

[46] NONG, G. HAMDI, M. e LETAIF, K. B. "Efficient scheduling of variable-length IP packets on high-speed switches"; *Proceedings of Global Telecommunications Conference - Globecom'99*, pp. 1407-1411, 1999

[47] NAKAKI, Y.; et all. "Scheduling algorithm with Priority of Active Buffer for Variable-Lengh IP Packet over Input-Buffered ATM Switch"; *Proceedings of Communications, Computer and Signal Processing, Conference on 2001. PACRIM IEEE Pacific* - Vol 2; pp. 526-529; 2001.

[48] SHREEDHAR, M. e VARGUESE, G.: "Efficient Fair Queueing using Deficit Round Robin", *Proc. SIGCOMM'95 Conf.*, ACM, pp.231-243, 1995.

[49] DEMERS, A; KESHAV, S. e SHENKER, S. "Analysis and simulation of a fair queueing algorithm." *Proceedings of ACM SIGCOM, 1989*, pp. 3-12, Setembro de 1989

[50] NAGLE, J. "On packet switches with infinite storage" *IEEE Trans. On Comm.*, COM-35(4), April 1987.

[51] MCKENNEY, P. "Stochastic fairness queueing". *Internetworking: Research and Experience. Vol 2, pp.113-131, Janeiro* 1991

[52] GOLESTANI, S. "A self clocked fair queueing scheme for broadband applications." *In Proc. IEEE Infocomm. '94*, 1994.

[53] PAREKH, A. K.; GALLAGHER, R. G.; "A generalized processor sharing approach to flow control in integrated services networks" *In Proc. IEEE Infocomm '93*, 1993.

[54] HAMMOND, J. L.; O'RELLY, P. J. P. "Performance Analysis of Local Computer Networks", *Addison-Wesley Publishing Company*, Chap. 3, pp. 98-104, 1986.

[55] ACHARYA, A.; DIGHE, R.; E ANSARI, F.; "IP Switching Over Fast ATM Cell Transport (IPSOFACTO): Switching multicast flows"; *Proceedings of Global Telecommunications Conference - Globecom'97*, pp. 1850-1854, 1997

[56] SHIMONISHI, H.; MURASI, T. e YAMADA, K.; "IP-ON-THE-FLY packet processing mechanism for an ATM/IP integrated switch", *Proceedings of Global Telecommunications Conference - Globecom'99*, pp. 626-630, 1999.

[57] NEWMAN, P.; MINSHALL, G. e LYON,T.L.; "IP Switching—ATM under IP"; *IEEE/ACM Transactions on Networking,* Vol. 6, nº. 2, pp. 117-129; Abril 1998

Apêndice A

Glossário

AAL	*ATM Adaptation Layer*
ABR	*Available Bit Rate*
ARIS	*Aggregate Route-Based IP Switching*
ATM	*Asynchronous Transfer Mode*
BGP	*Border Gateway Protocol*
CAC	*Connection Admission Control*
CBR	*Constant Bit Rate*
CDV	*Cell Delay Variation*
CE	Controlador de Entrada
CLP	*Cell Loss Priority*
CLR	*Cell Loss Ratio.*
CNET	*Centre National d'Etudes des Telecommunications*
CS	Controlador de Saída
CSR	*Cell Switching Router*
DiffServ	*Differentiated Services*
DMUX	Demultiplexador
DRR	*Deficit Round Robin*
FIFO	*First-In-First-Out*
FIRO	*First-In-Random-Out*
GFC	*Generic Flow Control*
GSMP	*General Switch Management Protocol*
HEC	*Header Error Check*
HOL	*Head Of Line*
HOLB	*Head Of Line Blocking*
HSR	*High-Speed Statistical Retry*
IETF	*Internet Engineering Task Force*
IFMP	*Ipsilon Flow Management Protocol*
IntServ	*Integrated Service.*
IP	*Internet Protocol*
IP-PIM	*IP Parallel Iterative Matching*

IP *Switching*	Comutação IP
IPv4	*Internet Protocol - version 4*
IPv6	*Internet Protocol - version 6*
IS-IS	*Intermediate System to Intermediate System Protocol*
ITU-T	*International Telecommunications Union*
LSI	*Large Scale Intergrated*
LSP	*Label Switching Path*
maxCTD	*Maximum Cell Transfer Delay*
MBS	*Maximum Burst Size*
MCR	*Minimum Cell Rate*
MPLS	*Multi Protocol Label Switch*
MUX	Multiplexador
NACK	*Not Acknowledge*
NNI	*Network Network Interface*
nrtVBR	*non-real time Variable Bit Rate*
OSPF	*Open Shortest Path First*
PCR	*Peak Cell Rate*
PHP	*Per Hop Behavior*
PIM	Parallel Iterative Matching
PM	*Physical Medium*
PRM	*Protocol Reference Model*
PT	*Payload Type*
QoS	*Quality of Service*
RAM	*Random Access Memory*
RDSI	Rede Digital de Serviços Integrados
RDSI-FL	RDSI de Faixa Larga
RSVP	*Resource Reservation Protocol*
rtVBR	*real-time Variable Bit Rate*
SCR	*Sustainable Cell Rate*
SLAM	*Simulation Language for Alternative Modeling*
SLIP-IRRM	*Iterative Round-Robin Matching with SLIP*
SPIM	*Simplified Parallel Iterative Matching*
IRRM-MC	*Iterative Round-Robin Matching with Multiple Classes*
Tag Switching	Comutação de Rótulo
TC	*Transmission Converge*
TTL	*Time to Live*

UBR	*Unspecified Bit Rate*
UCR	Unidade de Controle de Requisição
UNI	*Unit Network Interface*
VCC	*Virtual Channel Connection*
VCI	*Virtual Channel Identifier*
VCL	*Virtual Channel Link*
VPC	*Virtual Path Connection*
VPI	*Virtual Path Identifier*
VOQ	*Virtual Output Queue*
WFQ	*Weighted Fair Queuing*