Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

Esquema de Escalonamento *Cross-layer* baseado no DQCA, com reserva periódica de *time-slots* para tráfego de voz

JOSIAS TEIXEIRA GUIMARÃES

NOVEMBRO / 2011

ESQUEMA DE ESCALONAMENTO CROSS-LAYER BASEADO NO DQCA, COM RESERVA PERIÓDICA DE TIME-SLOTS PARA TRÁFEGO DE VOZ

JOSIAS TEIXEIRA GUIMARÃES

Dissertação apresentada ao Mestrado em Telecomunicações do Instituto Nacional de Telecomunicações – INATEL, como requisito parcial para obtenção do título de Mestre em Telecomunicações.

ORIENTADOR: PROF. DR. JOSÉ MARCOS CÂMARA BRITO

SANTA RITA DO SAPUCAÍ – MG

2011

Guimarães, Josias Teixeira
G429e
Esquema de Escalonamento Cross-Layer Baseado no DQCA, com Reserva Periódica de Time-Slots para Tráfego de Voz / Josias Teixeira Guimarães. – Santa Rita do Sapucaí, 2011.
118 p.

Orientador: Dr. José Marcos Câmara Brito
Dissertação de Mestrado – Engenharia de Telecomunicações – Instituto Nacional de Telecomunicações – INATEL.
Inclui bibliografia e anexo.

1. Cross-Layer 2. DQCA 3. Redes sem fio 4. Escalonamento Oportunista. 5. Tráfego Heterogênio. I. Brito, José Marcos Câmara. II. Instituto Nacional de Telecomunicações – INATEL. III. Título.
CDU 621.39

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em 04/11/2011, pela comissão julgadora:

_____

Prof. Dr. José Marcos Câmara Brito (Orientador) – Inatel

_____

Prof. Dr. Carlos Roberto dos Santos – Inatel

_____

Prof. Dr. José Ferreira de Rezende – UFRJ, RJ

_____

Prof. Dr. Luciano Leonel Mendes

Coordenador do Curso de Mestrado - INATEL

Dedico este trabalho a minha Mãe,

meus filhos, minha esposa

e meus irmãos.

AGRADECIMENTOS

A Deus

A minha esposa e filhos.

A minha mãe, Irene, e meus irmãos por me apoiarem sempre, pelas orações e por todo o incentivo que me deram.

A meu orientador, José Marcos Câmara Brito, por dividir comigo o seu vasto conhecimento, pela atenção e paciência durante o processo de orientação.

Ao meu primo Dayan, pelo incentivo, pela ajuda em todos os sentidos e pela velha e sincera amizade.

Ao INATEL, a todos os funcionários e professores e aos colegas que estiveram comigo nos estudos, nos trabalhos e no LabPG.

Ao Instituto Federal de Educação, Ciência e Tecnologia – Campus Barbacena, aos colegas de trabalho e em especial ao Prof. Marlon pelo apoio e colaboração.

A todos que oram e torcem pelas minhas realizações.

Resumo

O DQCA (*Distributed Queuing Collision Avoidance*) é um protocolo de acesso ao meio de alto desempenho e estável, que mantém uma vazão alta com qualquer carga de tráfego. Entretanto, ele pode permitir variações muito altas no atraso, principalmente quando mensagens grandes ocupam o canal por períodos muito longos e este é um comportamento indesejável para tráfegos com taxa de *bit* constante, como o de voz, que possuem requisitos rígidos de QoS (*Quality of service* – Qualidade de serviço), relacionados ao atraso e variação do atraso. Neste trabalho é proposto um esquema cross-layer baseado no protocolo DQCA, que visa principalmente reduzir o atraso médio e a variação do atraso dos pacotes de voz. Este esquema, chamado DQCA-RP (DQCA com reserva periódica), procura manter a alta vazão alcançável com o DQCA, enquanto melhora a QoS dos serviços de voz, através da reserva periódica de time-slots para os pacotes deste tipo, de acordo com a taxa de geração dos mesmos.

Palavras chave: Cross-*Layer*, DQCA, Redes sem Fio, Escalonamento Oportunista, Tráfego heterogêneo, QoS.

Abstract

The DQCA (*Distributed Queuing Collision Avoidance*) is a high-performance and stable medium access protocol, which maintains a high throughput with any traffic load. However, it can allow very high variations in delay, especially when large messages occupy the channel for long periods and this is an undesirable behavior for traffic with constant bit rate, such as voice, which have strict requirements on Quality of service (QoS) related to delay and jitter. In this work we propose a cross-layer scheme based on DQCA protocol, which mainly aims to reduce the average delay and jitter of voice packets. This scheme, called RP-DQCA (DQCA with periodic reservation), aims to maintain the high throughput achievable with the DQCA, while improving the QoS of voice services through periodic reservation of time slots for packets of this type, according to the generation rate of the same.

Keywords: Cross-Layer, DQCA, Wireless networks, Opportunistic Scheduling, Heterogeneous traffic, QoS.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

ACK – *Acknowledgement*

AP – *Access Point*

ARQ – *Automatic Repeat reQuest*

Bfm – Bit de final de mensagem

C-PRMA – *Centralized Packet Reservation Multiple Access*

Cross-layer design – Projeto de otimização em camadas cruzadas

CRQ – *Collision resolution queue*

CSMA/CA – *Carrier Sense Multiple Access/Colision Avoidance*

CTS – *Clear to Send*

DCF – *Distributed Coordination Functions*

DIFS – *Distributed interframe space*

Downlink – Transmissão de uma estação base para os terminais

Downward – Descendente

DPRMA – *Dynamic Packet Reservation Multiple Access*

DQCA – *Distributed Queuing Collision Avoidance*

DQCA-RP – DQCA com reserva periódica

DQRAP – *Distribute Queueing Random Access Protocol*

DTQ – *Data transmission queue*

ECN – *Explicit congestion notifica*tion

EOT – *End-of-talkspurt*

FBP – *Feedback packet*

FC – *Frame control*

FCS – *Frame Control Sequence*

FEC – *Forward Error Correction*

IPRMA – *Integrated Packet Reservation Multiple Access*

ISO – *International Standards Organization*

jitter – Variação do atraso

LLC – *Logical Link Control*

MAC – *Medium Access Control*

MPRMA – *Modified PRMA*

NAV – *Network Allocation Vector*

NRT – *Non-Real-Time*

OS – *Oportunistic Scheduling*

OSI – *International Standards Organization*

PCF – *Point Coordination Function*

PR – *Polling Register*

PRMA – *Packet Reservation Multiple Access*

QoS – *Quality of service*

R – *Aloha Reservation Aloha*

RT – *Real Time*

RTS – *Request to Send*

SIFS – *short inter frame space*

SNR – *Signal-to-Noise Ratio*

TDMA – *Time Division Multiple Access*

Upward – Sentido ascendente

VBR – *Variable bit rate*

VTQ – *Voice Transmission Queue*

WiMAX – *Worldwide Interoperability for Microwave Access*

WLAN - *Wireless Local Area Network*

WMAN – *Wireless Metropolitan Area Network*

WPAN – *Wireless Personal Area Network*

WWAN – *Wireless Wide Area Network*

Sumário

Capítulo 1 - Introdução

1.1. Contextualização e Motivação

Nos últimos anos, tem se observado uma grande expansão do uso das redes sem fio, devido principalmente à flexibilidade e praticidade desta tecnologia. Além disso, as taxas de transmissão dessas redes estão aumentando e os custos dos equipamentos reduzindo consideravelmente. Assim, a cada dia está surgindo uma grande variedade de dispositivos que acessam as redes sem fio e, conseqüentemente, há um aumento da diversidade dos serviços oferecidos, que compreendem além dos serviços de dados, como transferência de arquivos, http, *e-mail* e outros serviços *web*, outros de tempo-real, como voz, vídeo, jogos e etc. Diante disso, este assunto representa atualmente um dos principais tópicos de pesquisa nas áreas de computação, eletrônica e telecomunicações.

Com toda essa demanda crescente por serviços de comunicação móveis e sem fio, uma ampla variedade de aplicações requer tratamentos diferenciados para os diversos tipos de serviço disponíveis. Enquanto alguns tipos de tráfego não possuem requisitos rígidos de qualidade de serviço (*QoS – Quality of service*), ditos serviços de melhor esforço, podendo ser atendidos nos intervalos em que o sistema estiver ocioso, outros como os de voz ou vídeo sob demanda não toleram altos índices de atraso ou variações de atraso (*jitter*) muito elevadas, apesar de permitir pequenos níveis de perdas de pacotes. O limite de atraso sobre os pacotes de voz, por exemplo, é um fator muito importante para uma conversação inteligível no receptor. Valores de atraso altos e variações acentuadas do atraso sobre esses pacotes alteram o modo como a conversação ocorre, gerando degradações da qualidade de voz perceptíveis ao usuário. Assim, uma conversação

contínua e de qualidade aceitável precisa ser reconstruída no receptor a partir de pacotes de voz que experimentaram atrasos variáveis através da rede. De acordo com os autores em [1], o *jitter* no receptor é minimizado armazenando os pacotes em buffer e liberando-os em intervalos regulares, ao reproduzir o som.

O padrão para redes locais sem fio (*WLAN - Wireless Local Area Network*) mais difundido atualmente é o IEEE 802.11 [2], também conhecido como *Wi-Fi*. Desde a criação deste padrão surgiram várias versões, como a 802.11a, 802.11b e 802.11g. Entretanto, apesar disso ele ainda apresenta limitações em termos de eficiência, estabilidade e QoS. Foram propostas algumas versões do IEEE 802.11 buscando melhorias em requisitos específicos das redes sem fio, como o 802.11n para o aumento na vazão, o 802.11i que busca melhorias em segurança, crítica no 802.11, e o 802.11e, que procura resolver problemas relacionados à qualidade de serviço. Apesar dos avanços obtidos com 802.11e, ele ainda apresenta problemas relacionados ao desempenho com alguns fluxos multimídia [2]. Assim, prover QoS em WLAN ainda é um desafio, devido principalmente a fatores relacionados ao meio sem fio, como a mobilidade, altas taxas de erro de *bit* e baixa largura de banda.

O método de acesso baseado no modo Função de Coordenação Distribuída (*DCF – Distributed Coordination Functions*), que é obrigatório e o mais empregado no 802.11, suporta somente tráfego do tipo melhor esforço, ou seja, não provê nenhuma garantia de transmissão ou de QoS e também não possui mecanismos de diferenciação que possibilitem a priorização no atendimento a determinados tipos de serviço. Este comportamento contrasta com as exigências das novas aplicações multimídia que têm surgido e daí a necessidade de se desenvolver novas soluções, que viabilizem a transmissão desses tipos de tráfego.

Então, vários mecanismos estão sendo propostos para prover QoS em redes sem fio e pesquisas apontam que, devido a diversos fatores inerentes nessas redes, o projeto de protocolos em camadas [3], cujo princípio mais evidente é a independência entre camadas da pilha de protocolos, não é o mais adequado para esse meio. Várias propostas exploram as trocas de informações entre camadas da pilha de protocolos para melhorar o desempenho dos sistemas de comunicação sem fio, sendo este conceito

conhecido como projeto de otimização em camadas cruzadas (em inglês, *Cross-layer design*) [4]. Vários mecanismos de acesso ao meio que empregam conceitos *Cross-Layer* têm sido propostos na literatura e esses esquemas geralmente levam em conta, na decisão de escalonamento, as informações sobre as qualidades dos enlaces dos usuários, obtidas da camada física e/ou informações sobre os tipos de serviço, vindas das camadas superiores. Os autores em [5] e [6] propõem mecanismos para tráfego heterogêneo (voz e dados) baseados no protocolo DQCA (*Distributed Queuing Collision Avoidance*) [7]. Os esquemas propostos empregam conceito *cross-layer*, no qual informações da camada física são utilizadas na camada MAC para melhorar a vazão, enquanto informações sobre os tipos de tráfego, vindas das camadas superiores, permitem a diferenciação dos tipos de serviços, a fim de prover QoS para tráfego de voz.

1.2. Objetivo deste trabalho

O objetivo deste trabalho é propor um esquema *cross-layer* baseado no protocolo DQCA, que visa principalmente reduzir o atraso e a variação do atraso dos pacotes de voz. Este esquema, chamado DQCA-RP (DQCA com reserva periódica), procura manter a alta vazão alcançável com o protocolo DQCA, enquanto melhora a QoS dos serviços de voz, através da reserva periódica de *time-slots* para os pacotes deste tipo de acordo com a taxa de geração dos mesmos.

1.3. Estrutura da dissertação

O restante desta dissertação é organizado da seguinte forma: no Capítulo 2 são apresentados alguns fundamentos teóricos necessários para o entendimento do restante da dissertação; são abordados alguns temas de forma sucinta, objetivando a contextualização dos mesmos, incluindo uma visão geral sobre o padrão para redes sem fio 802.11, projetos *cross-layer* e os protocolos DQCA e PRMA, que serviram de base para o esquema proposto neste trabalho. O Capítulo 3 descreve o esquema proposto, DQCA-RP, sua estrutura e funcionamento. O Capítulo 4 apresenta os resultados obtidos através de simulações por computador. São descritos o ambiente e os cenários

considerados nas simulações, as métricas de desempenho analisadas e os resultados obtidos, comparados com outros protocolos estudados. Finalmente no Capítulo 5 são apresentadas as conclusões finais e sugestões para trabalhos futuros.

Capítulo 2 - Fundamentação Teórica

Neste capítulo são apresentados alguns fundamentos teóricos necessários para o entendimento dos capítulos subseqüentes da dissertação. Na Seção 2.1 apresenta-se uma visão geral do protocolo da camada MAC do padrão IEEE 802.11. Na Seção 2.2 são introduzidos conceitos relativos a soluções *cross-layer* e escalonamento oportunista e é apresentada uma proposta de classificação de projetos *cross-layer*. Na Seção 2.4 é apresentado o protocolo PRMA, do qual foram obtidos os conceitos de reserva de *time-slots* para pacotes de voz empregados no esquema proposto neste trabalho. Finalmente, na Seção 2.5 é apresentado o protocolo DQCA, no qual o esquema proposto é baseado.

2.1. As Redes sem fio e o padrão IEEE 802.11

Existem vários tipos de redes sem fio, que variam de acordo com a tecnologia e a aplicação. As principais tecnologias de redes sem fio encontradas atualmente são:

Redes pessoais sem fio (*WPAN – Wireless Personal Area Network*). Também conhecidas como *Bluetooth*, são definidas pelo padrão IEEE 802.15, cobrem pequenas distâncias e são utilizadas principalmente para conectar dispositivos fisicamente próximos aos computadores, como impressoras, celulares, câmeras digitais e outros, sem a necessidade da utilização de cabos.

Redes locais sem fio (*WLAN – Wireless Local Area Netowrk*). São padronizadas pelo IEEE 802.11, oferecem altas taxas de transmissão e abrangem uma pequena dispersão geográfica, na ordem de poucas centenas de metros.

Redes metropolitanas sem fio (*WMAN – Wireless Metropolitan Area Network*). Também chamadas de WiMAX (*Worldwide Interoperability for Microwave Access* - Interoperabilidade Mundial para Acesso de Micro-ondas), estas redes são padronizadas pelo IEEE 802.16. Abrangem uma área maior que as WLANs e oferecem altas taxas de transmissão para um grande número de usuários, tanto para uso doméstico como empresarial.

Redes sem fio de longa distância (*WWAN – Wireless Wide Area Network*). Abrange uma grande dispersão geográfica e são voltadas para aplicações móveis, como a rede celular sem fio.

2.1.1. O padrão IEEE 802.11

Antigamente não havia compatibilidade entre os equipamentos de diferentes fabricantes devido à falta de uma padronização e esse era o maior problema encontrado em redes sem fio. Então surgiu a necessidade de se criar um padrão para essas redes que garantisse a interoperabilidade entre os equipamentos. Segundo Tanenbaum [3], além de manter a compatibilidade com o Ethernet, que já dominava o mercado de redes locais, a criação do novo padrão enfrentaria alguns desafios, como descobrir uma faixa de freqüência adequada e amplamente disponível, tratar questões relacionadas à segurança e resolver alguns problemas encontrados no meio sem fio, por exemplo, a possível mobilidade dos terminais, os problemas de terminais ocultos ou expostos [3] e a impossibilidade de os terminais transmitirem e ouvirem o canal simultaneamente, que impede a utilização de técnicas de detecção de colisões. Além disso, o padrão deveria ser economicamente viável e disponibilizar uma largura de banda suficiente. Então o IEEE lançou o padrão 802.11, que define as camadas MAC e física para transmissões sem fio. Conforme comentado anteriormente, algumas variações foram apresentadas para o padrão IEEE 802.11, como o 802.11a, o 802.11b e o 802.11g, que alcançam diferentes velocidades empregando variadas faixas de freqüência e técnicas de modulação. Outros padrões foram criados para prover melhorias em pontos específicos em redes locais sem fio, como o 802.11n, que busca um aumento da vazão, o 802.11i para melhorias na segurança e o 802.11e, para prover qualidade de serviço. Estas

versões do padrão não serão abordadas neste trabalho e podem ser consultadas em [8], [9] e [10].

O padrão IEEE 802.11 atua nas camadas de enlace e física da pilha de protocolos do modelo OSI [3], como pode ser visto na Figura 1. Os detalhes relacionados à camada física não serão abordados neste trabalho e podem ser consultados em [3]. A camada de enlace de dados do IEEE 802.11 se divide em duas subcamadas: a subcamada controle de acesso ao meio, ou camada MAC (*Medium Access Control*), que determina quem tem a permissão para transmitir, e a subcamada de Controle de Enlace Lógico (*LLC – Logical Link Control*), que faz a ligação lógica entre a MAC e as camadas superiores, tornando transparente para a camada de rede as diferenças entre as diversas variações do 802.

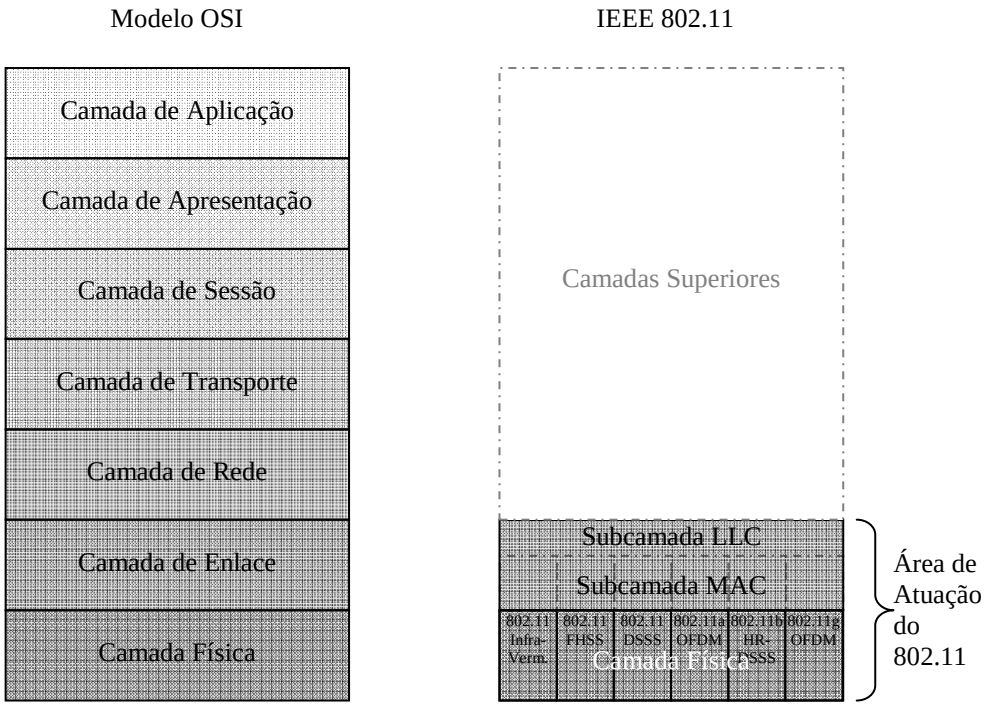

Figura 1: Parte da pilha de protocolos onde o 802.11 atua.

2.1.2. O protocolo da camada MAC do 802.11

O protocolo de acesso ao meio (MAC) do 802.11 suporta dois métodos de acesso: o PCF (*Point Coordination Function*) ou função de coordenação centralizada e o DCF (*Distributed Coordination Functions*) ou função de coordenação distribuída. O primeiro é opcional e utiliza uma estação base para controlar o acesso. O segundo, que é o mais utilizado na prática, é obrigatório em todas as implementações e não utiliza nenhuma espécie de controle centralizado; neste modo, as estações concorrem pelo acesso ao canal através do protocolo de acesso CSMA/CA (*CSMA with Collision Avoidance* – CSMA com prevenção de colisão), cujo funcionamento é descrito resumidamente a seguir. Detalhes completos sobre o funcionamento do protocolo MAC do 802.11 podem ser obtidos em [2].

O CSMA/CA, empregado no DCF, admite dois modos de operação: no primeiro modo, quando uma estação deseja transmitir um quadro, ela primeiro escuta o canal. Se o canal estiver ocioso por um período maior que um intervalo DIFS (*distributed interframe space* – tempo entre quadros distribuído) ela começa a transmitir ou se o canal estiver ocupado, adia a transmissão até que esteja ocioso por um período de tempo igual a DIFS e então, a fim de minimizar a probabilidade de colisão, a estação gera um atraso aleatório antes de transmitir, por meio de um contador de *backoff*, que é inicializado com um valor aleatório entre zero e um determinado número de janelas de tempo. Se houver colisão, as estações envolvidas aguardam um tempo aleatório, usando o algoritmo de recuo binário exponencial [3] e depois tentam novamente. O outro modo de operação do CSMA/CA emprega o esquema de detecção de canal virtual, que se baseia nos quadros RTS (*Request to Send* – Solicitação de acesso), CTS (*Clear to Send* – Livre para o envio) e *ACK* (*Acknowledgement* – Confirmação): quando uma estação deseja transmitir, se o canal estiver ocioso, ela envia um quadro RTS e aguarda o CTS do terminal de destino. Se receber o CTS, a estação envia seu quadro, inicia um contador de tempo (*timer ACK)* e aguarda o quadro de confirmação (*ACK*); se o *timer* expirar antes da chegada do *ACK*, o protocolo é reiniciado. Se o canal estiver ocupado, a estação é capaz de calcular o tempo que o canal permanecerá ocupado com base nos quadros RTS e CTS enviados pelos outros terminais e mantém uma espécie de canal

virtual para a transmissão, que a própria estação controla através de um contador de tempo NAV (*Network Allocation Vector* – Vetor de alocação de rede).

2.2. Soluções Cross-Layer

2.2.1. O modelo em camadas

O modelo de referência OSI (*Open Systems Interconnection*) proposto pela ISO (*International Standards Organization*), define um conjunto de regras para padronizar o projeto de protocolos de comunicação e facilitar a interconexão de sistemas de computadores.

Segundo Tanenbaum [3], neste modelo as redes são organizadas em pilhas de camadas ou níveis, para se reduzir a complexidade do projeto de protocolos. O objetivo de cada camada é prover determinados serviços às camadas superiores, definidos através de uma interface existente entre cada par de camadas adjacentes. Neste contexto, os protocolos são conjuntos de regras para comunicação entre uma camada de um dispositivo de origem e a camada correspondente no nó de destino. Como pode ser visto através das linhas tracejadas na Figura 2, a camada *n* de uma máquina se comunica com a camada *n* da outra máquina. Entretanto, na realidade os dados não são transferidos desta forma, mas é no meio físico que se dá a comunicação propriamente dita. Para que a informação chegue à camada correspondente da máquina de destino, o protocolo de cada camada transfere os dados e informações de controle para a camada adjacente imediatamente abaixo dela até alcançar a camada mais baixa, próxima ao meio físico, por onde a informação é transmitida. Ao atingir a máquina de destino, o fluxo segue na ordem inversa, do maio físico para as camadas superiores, até chegar à camada que iniciou a comunicação.

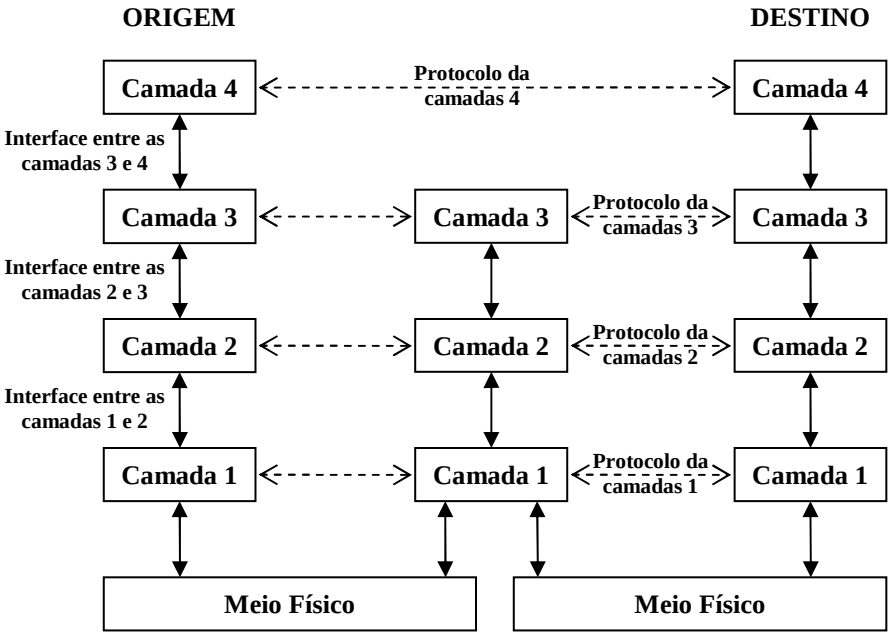

Figura 2: Modelo ilustrativo com 4 camadas

O modelo OSI é composto de sete camadas, conforme pode ser visto na Figura 1. Ele define somente o que cada camada deve fazer, sem especificar exatamente os serviços e protocolos a serem utilizados em cada uma delas. As sete camadas do modelo OSI são apresentadas a seguir, destacando-se de forma sucinta apenas as funções mais significativas de cada uma, apenas para o entendimento dos conceitos apresentados nas seções subseqüentes:

A camada física: controla basicamente a modulação e a potência de transmissão, para enviar os *bits* pelo meio físico;

A camada de enlace de dados: controla o acesso ao meio compartilhado, evita ou reduz o número de colisões e melhora a confiabilidade do enlace através da fragmentação dos dados em quadros e do uso de FEC (*Forward Error Correction* – Correção direta de erros) e/ou ARQ (*Automatic Repeat reQuest* – Requisição de repetição automática);

A camada de rede: define o melhor caminho a ser seguido pelos pacotes, da origem até o destino;

A camada de transporte: estabelece comunicações fim-a-fim através da rede;

A camada de sessão: permite o estabelecimento de sessões entre usuários de diferentes máquinas;

A camada de apresentação: realiza conversão do dado recebido pela camada de aplicação para outro formato, por exemplo, pode realizar compressão ou criptografia sobre este dado;

A camada de aplicação: contém os protocolos relativos aos serviços das aplicações dos usuários.

2.2.2. Cross-layer

Segundo Srivastava e Motani [11], o modelo em camadas divide as tarefas a serem executadas em uma rede em camadas e define a hierarquia dos serviços a serem providos através do projeto de protocolos específicos para as diferentes camadas. Esta arquitetura impõe algumas regras no projeto de protocolos; por exemplo, não permite comunicação entre camadas não adjacentes e a comunicação entre camadas adjacentes deve ser feita através de procedimentos de chamadas e respostas, de forma que protocolos de camadas mais altas fazem uso dos serviços providos pelas camadas mais baixas apenas através da interface definida pela arquitetura, sem conhecer os detalhes de implementação destes serviços. Ainda de acordo com [11], o projetista tem a opção de escolha entre seguir ou não estas regras e qualquer projeto que viola as regras impostas pela arquitetura é considerado um projeto *cross-layer*. Por exemplo, permitir a comunicação direta entre camadas não adjacentes, compartilhando variáveis globais entre diferentes camadas, ou ainda projetando um protocolo em uma camada levando-se em conta a implementação do protocolo de outra camada.

As violações do modelo se fizeram necessárias principalmente devido a problemas presentes no meio sem fio, como mobilidade dos dispositivos e interferências, que acarretam em erros na recepção. Entretanto, os meios sem fio oferecem várias

possibilidades de comunicação oportunista que não poderiam ser exploradas com projetos unicamente em camadas.

2.2.2.1. Tipos de projetos *cross-layer*

De acordo com [11], os projetos *cross-layer* podem ser classificados em quatro categorias, dependendo das formas como as regras da arquitetura em camadas são violadas:

Criação de novas interfaces;

Fusão de camadas adjacentes;

Acoplamento de projetos sem a criação de novas interfaces;

Calibração vertical através das camadas.

A) Criação de novas interfaces

Este tipo de projeto *cross-layer* é baseado na criação de novas interfaces para compartilhamento de informações entre as camadas. O que caracteriza a violação da arquitetura em camadas é a criação de uma nova interface não prevista na arquitetura.

Pode-se dividir essa classificação em três subcategorias, dependendo da direção do fluxo de informação entre as camadas: *Upward* (ascendente), com o fluxo indo de uma camada inferior para uma superior, *Downward* (descendente), de uma camada superior para uma inferior ou *Back-and-Forth* (vai-e-vem), em ambas as direções.

1) Fluxo de informação *Upward*

O protocolo de uma camada superior requer alguma informação de uma camada inferior, através da criação de uma nova interface da camada mais baixa para a mais alta, conforme mostrado na Figura 3. Como exemplos podem-se citar o esquema proposto neste trabalho, DQCA-RP, e outros esquemas de escalonamento oportunistas apresentados no decorrer desta dissertação, os quais procuram aumentar a vazão da rede dando prioridade de transmissão aos terminais com melhores condições dos enlaces. Nesses esquemas oportunistas, a informação sobre a condição do canal pode ser enviada da camada física para a de enlace, que por sua vez a utiliza na decisão de escalonamento. De acordo com [11], outros exemplos que utilizam essa interface podem ser obtidos em esquemas de modulação adaptativa ao estado do canal, e.g., no trabalho de Ji *et al*. [12], os quais procuram adaptar os parâmetros da transmissão, como modulação e codificação usadas, de acordo com a informação do canal recebida da camada física, a fim de reduzir perdas de quadros no canal.

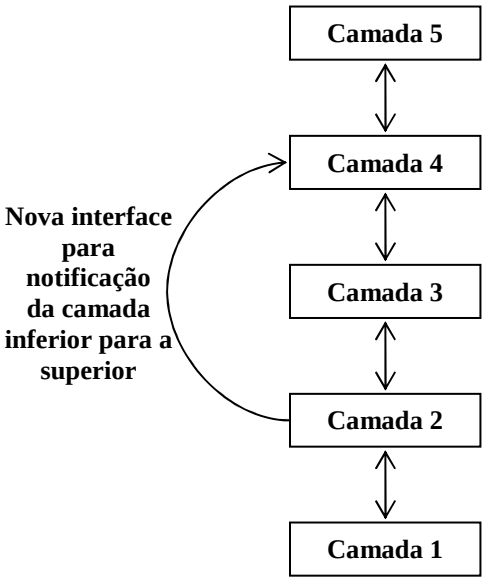

Figura 3: Projeto *cross-layer* com a criação de nova interface *Upward*

Outro exemplo de projeto *cross-layer* com interface *upward* é a notificação explícita de congestionamento (*ECN - Explicit congestion notification*) enviada de um roteador para a camada de transporte do transmissor TCP. Segundo Shakkottai *et al.* [4], erros no enlace podem ser interpretados erroneamente pelo transmissor como congestionamento, fazendo com que este diminua a janela de transmissão, reduzindo o desempenho da rede. Este problema pode ser evitado através de um projeto *cross-layer* com interface

upward, através da qual o roteador envia uma ECN para a camada de transporte, permitindo-se diferenciar entre perdas devido a erros no enlace de perdas por congestionamentos.

2) Fluxo de informação *Downward*

O protocolo de uma camada mais acima na pilha de protocolos poderia utilizar uma nova interface direta com uma camada mais baixa para ajustar dinamicamente alguns parâmetros desta camada, conforme mostrado na Figura 4. Por exemplo, a camada de enlace pode utilizar nas decisões de escalonamento parâmetros sobre os tipos de tráfego, recebidos da camada de aplicação, visando atender aos requisitos de QoS de determinados tipos de serviço. Pode-se citar como exemplo deste tipo de projeto *cross-layer* o protocolo DQCA-RP proposto neste trabalho, apresentado em detalhes no Capítulo 3.

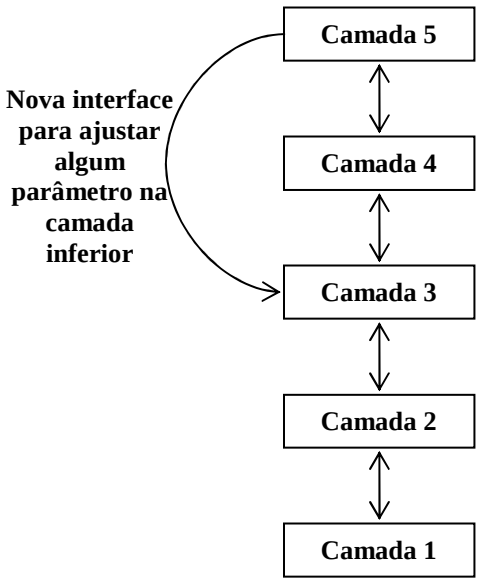

Figura 4: Projeto *cross-layer* com a criação de nova interface *Downward*

Outro exemplo desse tipo de interface pode ser visto na proposta apresentada em [13], onde a persistência do ARQ na camada de enlace pode ser alterada pela camada de aplicação. Quando for transmitido um fluxo de tempo real, é melhor perder quadros a atrasá-los, e assim a quantidade de retransmissões efetuadas pelo ARQ pode ser

reduzida. A proposta de Abd El Al *et al.* em [14] combina a solução anterior com FEC, evitando o descarte de quadros de vídeo com poucos erros.

3) Fluxo de informação *Back-and-Forth*

Este tipo de fluxo caracteriza-se por dois protocolos trabalhando em camadas diferentes de forma colaborativa. Informações são trocadas continuamente através de uma interface *back-and-forth*, com os parâmetros fluindo dinamicamente nos dois sentidos, conforme mostrado na Figura 5. Em [15] é proposto um algoritmo que realiza um controle conjunto de escalonamento e de potência em redes *ad hoc*, utilizando um projeto *cross-layer* com interface *Back-and-Forth*, entre as camadas de enlace e física. O algoritmo determina na camada de enlace o número admissível de usuários que podem tentar transmissões simultâneas em um dado *slot*. De acordo com o cenário definido na camada de enlace, na camada física é feito um controle de potência, a fim de satisfazer ao requisito de QoS, traduzido em taxa de erro de *bit*.

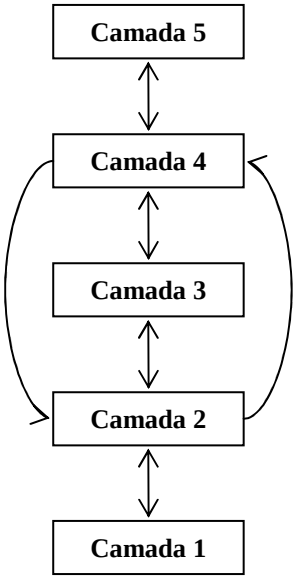

Figura 5: Projeto *cross-layer* com a criação de nova interface *Back-and-Forth*

B) Fusão de camadas adjacentes

Este tipo de projeto *cross-layer* consiste em unir os projetos de duas ou mais camadas adjacentes formando uma nova supercamada, de modo que o serviço provido por esta

supercamada seja a união dos serviços providos pelas camadas que a compõe. Esta arquitetura não exige a criação de novas interfaces, visto que a comunicação com as outras camadas da pilha de protocolos pode utilizar as interfaces existentes, como apresentado na Figura 6.

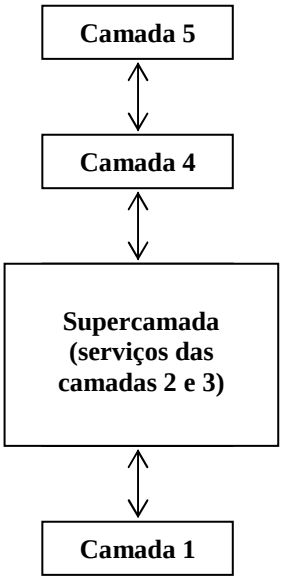

Figura 6: Projeto *cross-layer* com fusão de camadas adjacentes

De acordo com [11], apesar de não haver uma proposta *cross-layer* que cria explicitamente uma supercamada, o projeto colaborativo entre as camadas física e MAC tendem a ultrapassar a barreira existente entre elas, como visto na proposta de modulação adaptativa apresentada em [12], comentada no fluxo de informação *Upward* do tópico anterior.

C) Acoplamento de projetos

Neste tipo de projeto *cross-layer*, durante o projeto de um protocolo, alguns de seus parâmetros podem ser deixados explícitos para que possam ser acessados por um protocolo de outra camada, como apresentado na Figura 7. Assim, não há criação de novas interfaces para compartilhamento dinâmico de informações, porém esses protocolos estarão acoplados, de modo que um deles não pode ser alterado sem acarretar em modificações correspondentes no outro. Na realidade, o que caracteriza este tipo de

projeto *cross-layer* é o fato de o desenvolvimento de um protocolo em uma camada ser feito levando-se em conta o processamento em um protocolo de outra camada.

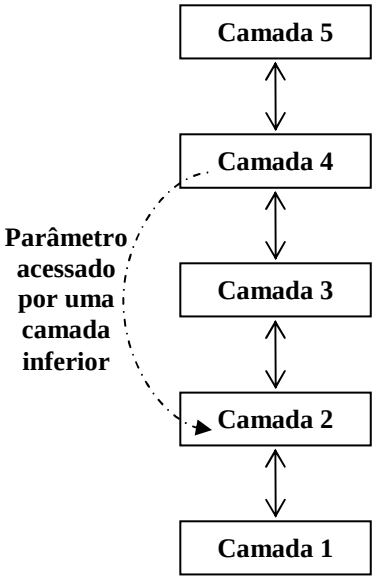

Figura 7: Projeto *cross-layer* com acoplamento de projetos

Um exemplo deste tipo de projeto *cross-layer* pode ser visto em [16], onde uma camada física seria capaz de receber múltiplos pacotes simultaneamente. Esta nova capacidade da camada física exige que a camada de enlace seja reprojetada, uma vez que ela não pode acomodar pacotes vindos de fluxos diferentes em um mesmo quadro.

D) Calibração vertical através das camadas.

Como o nome sugere, este tipo de projeto *cross-layer* caracteriza-se pelo ajuste de parâmetros que cruzam as camadas, como apresentado na Figura 8. Segundo os autores em [11], o desempenho do ponto de vista da camada de aplicação é uma função dos parâmetros das camadas inferiores. Portanto, a calibração através das camadas resultaria em maior ganho de desempenho que o ajuste dos parâmetros feito em pares de camadas.

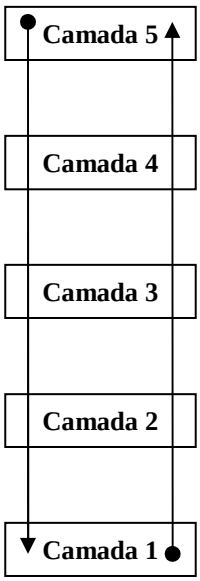

Figura 8: Projeto *cross-layer* com calibração vertical

Um exemplo de calibração vertical pode ser visto no projeto *cross-layer* em [17], no qual a persistência do ARQ na camada de enlace é controlada de acordo com os requisitos de atraso, definidos na camada de aplicação e é utilizada para ditar a seleção de taxa, através do esquema de modulação adaptativo na camada física.

2.2.2.2. Considerações sobre projeto *cross-layer*

Como visto nas seções anteriores, geralmente *cross-layer* se refere a projeto de protocolo desenvolvido burlando as regras de protocolos em camadas ou explorando a dependência entre as camadas da pilha de protocolos. Os benefícios que podem ser obtidos com o uso deste conceito são inúmeros, mas, no geral, as propostas existentes na literatura se concentram em três objetivos principais:

- resolver problemas específicos próprios do meio sem fio, que não poderiam ser solucionados satisfatoriamente com protocolos estritamente em camadas;

- a possibilidade de comunicação oportunista. De acordo com [11], a característica variável no tempo da qualidade do enlace permite a utilização oportunista do

canal; por exemplo, os parâmetros de transmissão podem ser ajustados dinamicamente de acordo com as variações da qualidade dos enlaces.

- tratar novas modalidades de comunicação oferecidas, com as quais a arquitetura em camadas não se ajusta.

Observa-se que os esforços em *Cross-Layer* têm sido feitos de forma independente, por pesquisadores de diferentes áreas e que trabalham em diferentes camadas na pilha de protocolos. Além disso, questões fundamentais como coexistência de diferentes propostas de projetos *Cross-Layer*, quando deveriam ser invocadas, quais regras seguir, etc., não estão sendo unificadas ou tratadas.

Nas seções anteriores foram apresentados os conceitos fundamentais de projetos *cross-layer*. Os principais tipos de projetos *cross-layer* foram mostrados, conforme classificação proposta em [11], e foram dados alguns exemplos de propostas relativas a cada classe identificada. Entretanto, este assunto não se esgota aqui, visto que este é um assunto recente e bastante amplo. Informações mais completas sobre propostas *cross-layer* existentes na literatura, assim como questões relativas à implantação de projetos *cross-layer* e suas interações nas diferentes camadas da pilha de protocolos, podem ser obtidas em [11], [18] e [4].

2.3. Esquemas de escalonamento oportunistas para tráfego heterogêneo

Alguns esquemas de escalonamento, ditos oportunistas (*OS - Oportunistic Scheduling*), procuram melhorar o desempenho das redes, explorando as variações das condições do canal, inerentes às redes sem fio. Os autores em [19] apresentam uma visão geral de esquemas de escalonamento oportunista, os quais utilizam algoritmos que decidem qual usuário escalonar em cada *time slot*, levando-se em conta a qualidade atual do enlace de cada usuário. Ainda apresentam esquemas de OS multi-serviço, que procuram dar garantias de QoS para diferentes tipos de fluxos de tráfego.

Apesar de esquemas de OS trabalharem na subcamada de controle de acesso ao meio (*MAC - Medium Access Control*), eles são inerentemente *Cross-Layer*. Como pode ser visto na Figura 9, OS multi-serviço é um protocolo *Cross-Layer* no qual o escalonador, que opera na subcamada MAC, recebe informações dos indicadores da qualidade do canal da camada física, traduzidos em taxas de transmissão suportável, e outras relacionadas ao tipo de serviço das camadas superiores, a fim de utilizá-las nas decisões de escalonamento.

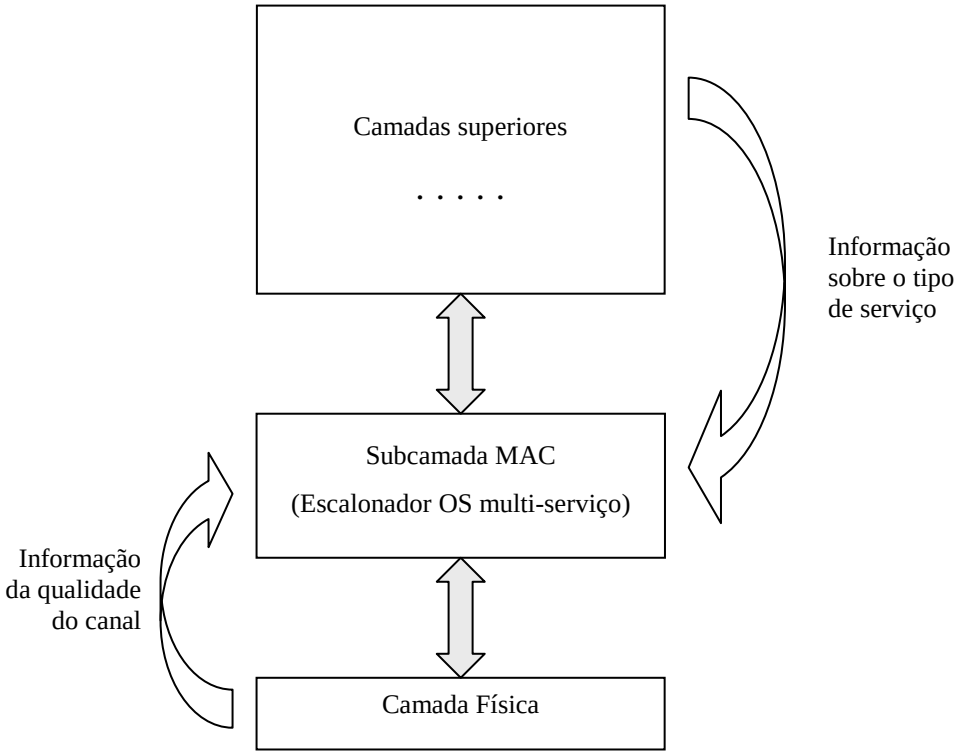

Figura 9: Interfaces *cross-layer* em Esquemas de OS multi-serviço

Com o objetivo de melhorar a vazão de redes sem fio, algoritmos de OS exploram as variações nos estados dos canais. Tais escalonadores procuram escolher dentre os usuários, aquele que experimenta as condições do canal relativamente melhores e suspender a transmissão de dados dos outros usuários, até que as condições de seus enlaces se aproximem das melhores condições de operação possíveis. Assim, o tempo gasto para transmitir a mesma quantidade de dados pode ser reduzido e, conseqüentemente, o número de usuários compartilhando o meio pode ser ampliado.

O funcionamento de um esquema de OS requer o conhecimento das qualidades do enlace de cada usuário ativo que compartilha um controlador, antes de cada decisão de escalonamento. Normalmente, cada usuário mede a relação sinal-ruído (*SNR - Signal-to-Noise Ratio*) do sinal recebido no sentido *downlink* (transmissão de uma estação base para os terminais) e esta informação retorna ao controlador central no *uplink* (transmissão de um terminal para uma estação base).

Para alguns esquemas de OS, ditos gananciosos, o aumento da vazão é alcançado com prejuízo de justiça e de qualidade de serviço. Em redes multi-serviços emergentes, nas quais os usuários fazem uso ao mesmo tempo de vários tipos de aplicações diferentes, tanto de tempo real (*RT – Real Time*), como voz sobre WLAN, vídeo e jogos, quanto não de tempo real (*NRT – Non-Real-Time*), como email e transferência de arquivos, somente um esquema de OS multi-serviços pode garantir um nível de serviço mínimo para cada um dos fluxos de tráfego ativos, buscando uma solução de compromisso ótima entre os objetivos de vazão, justiça e garantia mínima de serviço para múltiplos fluxos concorrentes de tráfego de dados.

2.4. PRMA

O protocolo de acesso ao meio PRMA (*Packet Reservation Multiple Access* – Acesso múltiplo com reserva de pacote), proposto por GOODMAN *et al.* [20], é uma adaptação do protocolo R-Aloha (*Reservation Aloha*) para o ambiente celular, que reserva *time-slots* periodicamente e dinamicamente para terminais de voz ativos transmitirem seus pacotes enquanto durar a comunicação. Ele permite que os terminais de voz ativos compartilhem o canal de maneira semelhante ao esquema TDMA (Acesso Múltiplo por Divisão de Tempo – *Time Division Multiple Access*) e que os *time-slots* não reservados para usuários de voz possam ser utilizados pelos usuários de dados, para transmitirem seus pacotes.

2.4.1. Estrutura e funcionamento

No PRMA, os terminais compartilham um único canal para transmissão dos pacotes no sentido *uplink*, dos terminais para um Ponto de Acesso (*AP – Access Point*). As fontes de informação podem ser classificadas em periódicas, como os pacotes de voz e outros sensíveis ao atraso, ou aleatórias, como pacotes de dados, mensagens de sinalização e algumas informações de controle [21].

O PRMA é baseado na divisão do canal em quadros, cuja duração é o período entre chegadas de pacotes de voz em um terminal. Como mostrado na Figura 10, o quadro é dividido em *time-slots*, sendo cada *time-slot* igual ao tempo de transmissão de um pacote de voz. Dessa forma, no PRMA, os tamanhos dos *time-slots* e dos quadros são projetados especificamente para acomodar eficientemente usuários de voz, permitindo que, se um terminal transmitiu o primeiro pacote em um determinado *time-slot*, ele poderá transmitir os pacotes seguintes utilizando o mesmo *time-slot* nos quadros subseqüentes. Segundo HANZO *et al.* [21], a duração do quadro PRMA, assim como o número de *time-slots* em um quadro, *n*, está relacionado a variáveis como taxa do canal, taxa do codificador de fonte de voz e número de *bits* no cabeçalho de cada pacote.

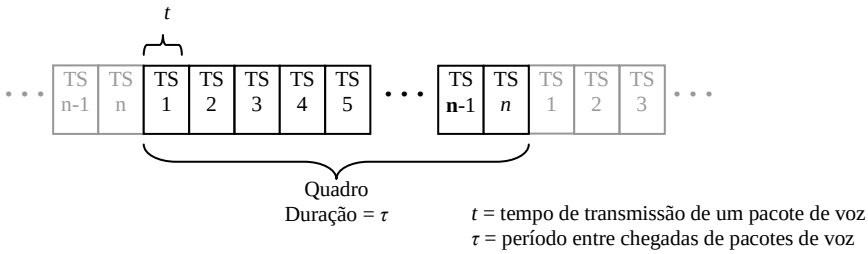

Figura 10: Estrutura do quadro PRMA

Cada *time-slot* em um quadro PRMA pode estar no estado disponível ou reservado, sendo que cada *time-slot* reservado é utilizado pelo terminal de voz (periódico) que o reservou e os *time-slots* disponíveis são disputados pelos outros terminais ativos, tanto aleatórios quanto os periódicos que ainda não têm *time-slots* reservados, quando eles têm pacotes para transmitir. O número de *time-slots* do quadro é do conhecimento de

todos os terminais, entretanto, eles não necessitam saber qual é o primeiro ou último *time-slot* do quadro. Para manter um controle sobre as situações dos *time-slots* de cada quadro, disponíveis e reservados, cada terminal mantém um registrador de reserva de quadro, com um *bit* para cada *time-slot*, sendo 0 (zero) para disponível ou 1 (um) para reservado.

O tipo de informação, periódico ou aleatório, é informado pelos terminais ao AP através de um *bit* no cabeçalho de cada pacote. O AP envia pacotes de realimentação no sentido *downlink* para todos os terminais, contendo informações relativas ao último pacote recebido e ao estado do *time-slot*, se está reservado ou disponível.

Quando um terminal periódico (voz) deseja iniciar uma comunicação ou um terminal aleatório tem pacote pronto para transmitir, eles competem pelo acesso ao canal utilizando os *time-slots* não reservados, seguindo regras semelhantes às do protocolo *Slotted-Aloha*. Na contenção, cada terminal transmite seu pacote no próximo *time-slot* disponível e aguarda o pacote de realimentação do AP, que o informa se o pacote foi recebido com sucesso ou se ocorreu uma colisão (transmissão de dois ou mais terminais no mesmo *time-slot*). No caso de colisão, os terminais periódicos tentam retransmitir os pacotes em *time-slots* disponíveis subseqüentes com probabilidade *q*, enquanto os terminais aleatórios tentam com probabilidade *r*. Os parâmetros *q* e *r* representam as probabilidades de permissão dos terminais periódicos e aleatórios respectivamente. Assim, definindo $q > r$, o sistema daria certa prioridade para as informações periódicas sobre as aleatórias, que podem tolerar atrasos muito maiores. Os terminais de voz descartam os pacotes que excedem o limite de atraso sem serem transmitidos com sucesso.

Os terminais aleatórios têm que competir pelo acesso ao canal para a transmissão de cada pacote presente no seu *buffer*. Já um terminal periódico, quando consegue transmitir com sucesso o primeiro pacote de uma conversa em um determinado *time-slot*, tal *time-slot* é marcado como reservado e o terminal poderá utilizá-lo nos quadros subseqüentes para continuar a comunicação. Isso garante que os pacotes restantes da comunicação sejam transmitidos sem disputa e sem colisão, pois um *time-slot* reservado pode ser usado nos quadros seguintes somente pelo usuário que o reservou, até que o

time-slot seja liberado no final da conversa. O terminal periódico sinaliza o final da conversa deixando o *time-slot* que foi previamente reservado para ele vazio, fazendo com que este volte a ser marcado como disponível.

Na Figura 11, apresenta-se uma seqüência de três quadros com *n time-slots* cada, nos quais o terminal 1 (T1) mantém uma reserva do terceiro *time-slot* e o terminal 8 (T8) mantém uma reserva do quinto *time-slot*. Ao terminar a conversa de um desses dois terminais, ele deixará o *time-slot* que está reservado para ele vazio (sem transmitir), fazendo com que no quadro seguinte tal *time-slot* seja considerado disponível.

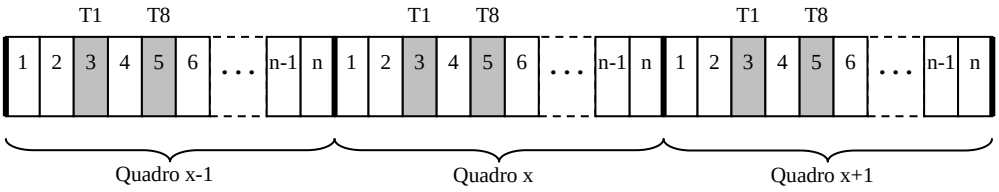

Figura 11: Seqüência de quadros PRMA com n slots

2.4.2. Considerações sobre o desempenho

BIANCHI et all. em [22], discutem algumas limitações do PRMA, como a ineficiência, devido ao recurso gasto com a reserva de acesso e, principalmente, a incapacidade de recuperar pacotes corrompidos. A retransmissão de um pacote corrompido é permitida somente um quadro depois, mesmo com *time-slots* disponíveis dentro desse intervalo, e esse atraso extra se acumula *slot* a *slot.* Além disso, a perda de um pacote, corrompido ou não recebido, pode causar a liberação do *time-slot* que estava reservado, podendo ocasionar colisões e fazer com que o terminal tenha que disputar novamente o canal para continuar transmitindo pacotes da mesma conversa.

JANI [23] faz um levantamento dos testes de desempenho já realizados e apresenta dois protocolos baseados no PRMA, os quais visam melhorar a vazão e solucionar problemas relacionados ao canal com desvanecimento, tornando o sistema mais robusto.

Mecanismos, como a proteção do pacote ou do seu cabeçalho através de códigos de correção de erros, propostos por HANZO *et al.* [21], poderiam amenizar alguns dos problemas citados. Entretanto, devido às características do meio sem fio, normalmente não há garantias de que o pacote seja recebido pelo AP, tornando essa solução inviável, se aplicada isoladamente.

2.4.3. Algumas variantes do PRMA

Foram propostas várias adaptações ao protocolo PRMA. Todos esses protocolos baseados no PRMA, como em [22]-[30], mantêm sua estrutura básica e o mecanismo de reserva para pacotes de voz, mas incluem melhorias e adaptações que visam uma série de benefícios, como a aplicação em outros ambientes, melhora no desempenho, acomodar melhor tráfegos com taxas de *bits* variáveis e tráfego multimídia ou resolver algumas limitações.

2.4.3.1. C-PRMA (*Centralized Packet Reservation Multiple Access*)

Em [22] é proposta uma versão melhorada do PRMA, chamada C-PRMA (*Centralized Packet Reservation Multiple Access*), baseada em *minislots* para contenção e num maior controle do acesso ao canal pelo AP. Nesse protocolo, o AP atua efetivamente como um controlador centralizado da largura de banda e da política de acesso. É ele quem decide, de acordo com regras específicas, sobre o estado de cada *time-slot* (disponível ou reservado) e a distribuição das reservas entre os terminais ativos.

No C-PRMA, os *time-slots* disponíveis são dedicados exclusivamente a requisições de reservas de acesso ao canal, não transportando informações com carga útil (voz ou dados). Como as requisições de reserva contêm uma quantidade limitada de informações, elas podem ser enviadas dentro de mini-pacotes. Então, os *time-slots*

disponíveis são divididos em *m mini-slots*, de tamanho igual ao tempo de transmissão do mini-pacote de requisição.

O protocolo C-PRMA procura aumentar a vazão, otimizando a utilização dos *time-slots*, através do emprego de *minislots* na contenção e da implementação de um algoritmo de escalonamento no AP, que gerencia eficientemente os *time-slots* de reserva e de transmissão. O algoritmo de escalonamento permite que os *time-slots* não reservados sejam utilizados, além das requisições de reserva, para a retransmissão imediata de pacotes perdidos, ou para a antecipação da transmissão de pacotes que aguardam nos *buffers* dos terminais. As decisões de escalonamento são tomadas levando-se em conta os requisitos de atraso de cada pacote a ser transmitido pelos terminais.

A cada *time-slot*, o AP decide e informa, através do pacote de realimentação, qual terminal terá a permissão para transmitir o seu pacote no próximo *time-slot* ou se tal *time-slot* estará disponível para contenção. Quando o AP informa aos terminais que um *time-slot* está livre para contenção, cada terminal que deseja fazer uma reserva escolhe um *mini-slot*, envia um pacote de requisição e aguarda o pacote de realimentação do AP. O pacote de realimentação informa se a reserva foi bem sucedida ou não. No caso de colisão, cada terminal envolvido escolhe um dos próximos *time-slot* livres e tenta novamente; se uma reserva foi bem sucedida, o terminal é dito “reservado” e fica aguardando a permissão do AP para transmitir cada pacote.

Para que o AP possa considerar os requisitos de atraso nas decisões de escalonamento, cada terminal informa o atraso que o próximo pacote a ser transmitido (pacote na saída do *buffer* do terminal) já sofreu, aguardando no seu *buffer*. Então, cada pacote transmitido pelos terminais, seja de requisição de reserva ou de carga útil, transporta o valor do atraso ω_i, em *time-slots*, experimentado pelo próximo pacote a ser transmitido, desde o instante em que foi gerado no terminal.

O algoritmo de escalonamento

O algoritmo de escalonamento implementado no C-PRMA considera somente fontes de voz, embora possa ser adaptado para suportar outros tipos de tráfego. Nessa seção,

apresenta-se uma descrição resumida do algoritmo, suficiente para o entendimento do protocolo e das regras de escalonamento. Informações mais detalhadas podem ser encontras em [22].

As fontes de tráfego, quando no estado ativo, geram pacotes a uma taxa constante de $1/T$ pacotes por segundo. O limite de atraso para pacotes de voz é de τ *time-slots* e pacotes que não forem transmitidos dentro desse limite são descartados. Como visto anteriormente, junto com cada pacote transmitido por um terminal i, é enviado o valor do atraso, ω_i, do próximo pacote presente no *buffer*. O valor de ω_i é considerado na decisão de escalonamento, mas também é utilizado pelo algoritmo como *flag*, para liberar a reserva quando o terminal não tiver mais pacotes para transmitir (fazendo $\omega_i = \tau$), ou para indicar que o pacote seguinte ainda não está no *buffer* do terminal, mas que a reserva deve ser mantida (fazendo ω_i negativo).

O algoritmo mantém um registrador de deslocamento (*PR – Polling Register*), contendo τ posições e cada terminal reservado tem o seu identificador ocupando uma posição nesse registrador. A posição que um terminal ocupa no PR, representa o tempo máximo (em *time-slots*) que o seu pacote pode esperar para ser transmitido, antes de ser descartado. Por exemplo, considerando a ocupação do PR apresentada na Figura 12, o terminal T5 deve ser atendido em no máximo 5 *time-slots*.

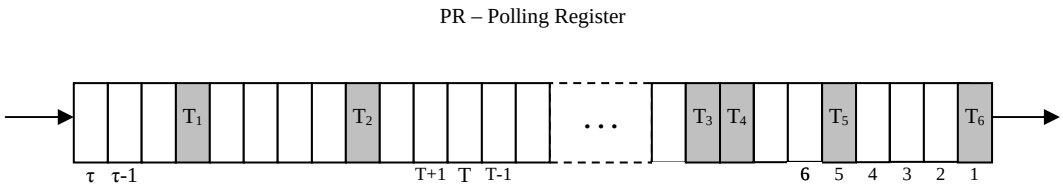

Figura 12: Exemplo de padrão de ocupação do PR

O terminal entra na PR quando ele obtém o acesso e a posição na qual ele é inserido depende do limite de atraso do seu pacote. Quando o AP recebe uma requisição com sucesso, ele calcula o valor de C_i, através da expressão $C_i = \tau - \omega_i$. Observa-se que, C_i representa o número máximo de *time-slots* que o pacote ainda pode esperar para ser transmitido (o limite de atraso menos o atraso já sofrido) sem ultrapassar o limite τ.

Então, o terminal entra no registrador na posição vazia de maior índice não maior do que C_i. Por exemplo, considerando a Figura 13, um novo terminal com valor calculado de C_i igual a 4, entraria na posição 3, já que a posição 4 está ocupada pelo terminal T5. Se não tiver posição vazia no PR entre 1 e C_i, significa que o requisito de atraso desse pacote não poderá ser atendido. Nesse caso o pacote é descartado, mas o algoritmo mantém a reserva para o terminal inserindo-o na posição $T + C_i$ do PR.

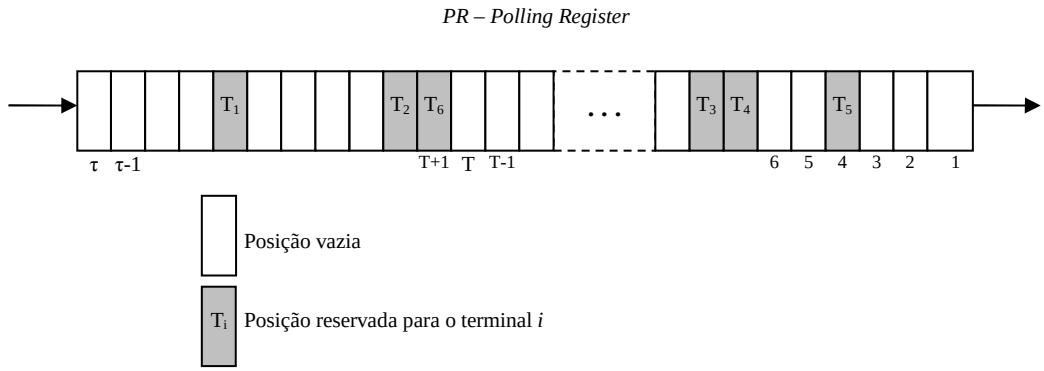

Figura 13: Ocupação do PR com a posição 1 livre

A cada *time-slot*, o PR é deslocado e o terminal que nele ocupar a posição 1 receberá a autorização para transmitir um pacote no próximo *time-slot*, como seria o caso do terminal T6 na Figura 12. A não ser que o terminal libere a reserva, ele entra novamente na PR, em uma posição de acordo com o valor de ω_i ou na posição $T + 1$, conforme a transmissão do pacote tenha ocorrido com sucesso ou não respectivamente. Vale notar que, em caso de erro na transmissão o pacote não poderia ser retransmitido, pois o seu limite de atraso já teria excedido.

Caso nenhum terminal ocupe a posição 1 do registrador, o *time-slot* seguinte pode ser utilizado para requisições de reserva, para retransmissão de um pacote recebido com erro, ou para antecipar a transmissão do próximo terminal na PR. O *time-slot* será utilizado para contenção se uma das seguintes condições for verdadeira: se ocorreu colisão no último *time-slot* disponível, se o PR estiver vazio ou se não houve contenção em *T time-slots*; se nenhuma das condições for satisfeita, será dada oportunidade de transmissão ao terminal que ocupar a posição não vazia mais baixa no PR (seria o terminal T5 na Figura 13). Nesse último caso, a transmissão é antecipada; isso é

interessante, pois, se ocorrer algum erro na transmissão o pacote pode ser retransmitido antes de estourar o seu limite de atraso, desde que tenha *time-slot* disponível.

2.4.3.2. IPRMA (Integrated Packet Reservation Multiple Access)

O autor em [23] apresenta um protocolo baseado no PRMA, o IPRMA (*Integrated Packet Reservation Multiple Access*), o qual visa melhorar a vazão, permitindo a reserva de *time-slots* tanto para pacotes de voz (periódicos) como para pacotes de dados (aleatórios).

No IPRMA, um mecanismo de priorização garante facilidade de acesso aos *time-slots* vazios para terminais de voz, que têm requisitos de atraso mais rigorosos, mas, enquanto os terminais de voz podem reservar *time-slots* individuais quadro a quadro, os terminais de dados podem reservar múltiplos *time-slots* dentro de um mesmo quadro. O principal objetivo do IPRMA é aumentar a vazão de dados e melhorar o desempenho geral do sistema, com o uso mais eficiente dos *time-slots* vazios.

Para pacotes de fontes periódicas, o processo de contenção e reserva de *time-slots* no IPRMA é idêntico ao usado no PRMA. Para pacotes aleatórios, o processo de contenção é semelhante, mas, diferente do PRMA, no qual um terminal deve entrar no processo de contenção para enviar cada pacote, no IPRMA, se um terminal tem vários pacotes prontos para transmissão, ele pode informar no pacote usado na contenção o número de *time-slots* que ele deseja reservar. Esse valor não pode exceder a quantidade de pacotes no *buffer* do terminal e, para garantir justiça e prioridade para terminais de voz, esse número deve ser limitado. Cada terminal sabe o número de *time-slots* vazios e as localizações deles no quadro.

Conforme comentado em [23], considerando um quadro com N *time-slots*, sendo K ($K \leq N$) *time-slots* vazios, para uma prioridade de voz de M ($M \leq N$) *time-slots*, um terminal de dados não pode reservar mais do que (K-M-1) *time-slots*, além do utilizado na

contenção. Se não houver colisão durante a transmissão do pacote de contenção, serão reservados até (*K*-*M*-1) *time-slots* disponíveis ou menos para os pacotes restantes. Quando terminarem os *time-slots* reservados para um terminal, se ele ainda tiver pacotes no *buffer*, o processo é reiniciado. Esse mecanismo garante acesso para os terminais de voz e proporciona maior *vazão* para terminais de dados.

2.4.3.3. MPRMA (Modified Packet Reservation Multiple Access)

Erros de transmissão de pacotes no sentido *uplink* podem levar ao cancelamento prematuro de reservas de *time-slots*. Devido à impossibilidade do AP decodificar corretamente os cabeçalhos, ele pode informar equivocadamente o *time-slot* como disponível. Os autores em [24] propõem a redução na taxa de erros nos cabeçalhos dos pacotes através de códigos de correção de erros relativamente simples, dispensando modificações no PRMA. Além disso, erros no sentido *downlink* podem fazer com que os terminais não reconheçam corretamente os estados dos *time-slots*, disponíveis ou reservados, ocasionando colisões ou *time-slots* não utilizados. Os autores em [25] propõem um protocolo PRMA modificado (*MPRMA – Modified PRMA*) para solução desses problemas e comprovam através de análise e simulação que esse protocolo é mais robusto que o PRMA padrão.

O funcionamento do MPRMA é basicamente igual ao do PRMA, com a implementação das seguintes modificações:

Um *bit* de final de conversa (*EOT – End-of-talkspurt*) é inserido no cabeçalho de cada pacote enviado pelos terminais, para indicar o final da comunicação. O valor 1 (um) para o *bit* EOT instrui o AP a liberar o *time-slot*. Este mecanismo permite a liberação do *time-slot* um quadro antes do PRMA básico, evitando a ocupação de um *time-slot* somente para essa operação. Para proteção de erros no *uplink*, se o AP falhar na decodificação desse *bit* por dois quadros seguidos ele libera a reserva do *time-slot*.

Após o recebimento de cada pacote, o AP envia um pacote de realimentação no sentido *downlink* (ACK), o qual informa aos terminais, através do cabeçalho, se o pacote foi recebido com sucesso e também o estado do *time-slot* para o quadro seguinte, se está reservado ou disponível. Para prevenir erros no pacote de realimentação, caso os terminais não consigam detectar o cabeçalho do pacote, todos, exceto o que transmitiu o último pacote, deixarão de transmitir nesse *time-slot* no próximo quadro.

2.4.3.4. DPRMA (Dynamic Packet Reservation Multiple Access)

O DPRMA (*Dynamic Packet Reservation Multiple Access*) proposto em [26] é um protocolo baseado no PRMA e, assim como ele, também é baseado em quadros, cuja duração é o tempo entre chegadas de pacotes de voz. O quadro é dividido em *time-slots*, cada um do tamanho de um pacote. O processo de contenção também é semelhante ao do PRMA básico, no qual os usuários utilizam *time-slots* disponíveis para disputar o acesso ao canal através do protocolo *Slotted Aloha*. Entretanto, a principal diferença entre os dois protocolos está na forma como os *time-slots* são reservados para os terminais. No DPRMA, o AP atua como um controlador centralizado, responsável por alocar recursos do canal entre os terminais.

O DPRMA permite a distribuição priorizada dos *time-slots* conforme o tipo de tráfego. Na medida em que dá prioridade aos usuários de tráfego de tempo real (*RT – Real Time*), mantém esforços para acomodar tráfego não de tempo real (*NRT – Non Real Time*) quando possível, sendo assim mais adequado para suportar tráfego multimídia, como voz, vídeo conferência e tráfego de dados. Ele é mais flexível na atribuição de largura de banda aos usuários, permitindo a reserva de um ou mais *time-slots* em um quadro ou de *time-slots* em quadros alternados para um usuário, de acordo com o seu requisito de taxa de transmissão e a capacidade disponível do canal. Além disso, as alocações de *time-slots* aos usuários podem ser modificadas dinamicamente, para permitir a alocação de *time-slots* para usuários mais prioritários e para suportar as necessidades de mudança de usuários com taxa de *bits* variável (*VBR – Variable bit*

rate). No DPRMA, usuários de dados (aleatórios) também podem fazer reservas, entretanto, a maneira com que o AP tenta acomodá-los é diferente de usuários RT.

Funcionamento do DPRMA

De acordo com o tráfego a ser transmitido e a taxa de geração dos pacotes, cada terminal determina a taxa de transmissão necessária para garantir a transmissão temporizada de seus pacotes e envia esse valor de requisição de taxa para o AP, através de *bits* de requisição de reserva (RR) que são incluídos no cabeçalho de cada pacote enviado. Para usuário com taxa de *bit* constante (*CBR – Constant bit rate*), o valor da taxa requerida deve ser o mais próximo possível da taxa de geração de pacotes. Para reduzir o *overhead*, o número de taxas de transmissão diferentes que um usuário pode requerer é limitado.

Todo terminal ativo deve monitorar o canal de *downlink* a cada *time-slot*, para saber quando ele poderá transmitir. Os novos pacotes gerados em cada terminal são armazenados em uma fila enquanto aguardam a transmissão e o usuário pode aumentar ou diminuir a requisição de taxa, conforme o tamanho da fila cresce ou diminui respectivamente. Assim, o valor da requisição de taxa aumenta ou diminui quando a ocupação da fila atinge certos níveis (limites) e esses níveis devem ser escolhidos cuidadosamente para cada tipo de tráfego. Esse processo permite evitar o estouro da capacidade do *buffer*, atrasos na transmissão e que, quando um *time-slot* for atribuído a um usuário ele não tenha pacote para transmitir (esvaziamento do *buffer*).

O AP distribui os *time-slots* dos quadros DPRMA entre os terminais ativos de acordo com os tipos de tráfego e as taxas requeridas pelos usuários. Quando um usuário se torna ativo, ou seja, transmitiu o primeiro pacote com sucesso, o AP tenta imediatamente atribuir dentre os *time-slots* disponíveis, tantos quantos forem necessários para atender à taxa requerida. Se os *time-slots* disponíveis não forem suficientes, é dado tratamento diferenciado para os diferentes tipos de tráfego: no caso de usuário de dados, o AP faz uma alocação parcial, atribuindo todos os *time-slots* disponíveis ao usuário e mantém um controle para permitir a alocação completa quando mais *time-slots* forem liberados; para usuários com requisitos rígidos de tempo, *time-*

slots que antes estavam reservados aos usuários de dados são utilizados para completar a requisição de taxa. Estes usuários de dados que tiveram seus *time-slots* "confiscados" aguardam em uma fila para terem novas reservas quando *time-slots* se tornarem disponíveis.

Quando um usuário se torna ativo, ou solicita um aumento da taxa requerida, mais *time-slots* devem ser atribuídos a ele. Ao atribuir *time-slots* a um usuário, o AP determina quais *time-slots* serão utilizados, procurando distribuir as reservas aleatoriamente entre os *time-slots* disponíveis, de forma que as atribuições de *time-slots* a um usuário fiquem espalhadas aleatoriamente pelo quadro.

Para fazer esse "espalhamento", primeiro o AP determina quantos e quais *time-slots* estão livres. Depois, examina seqüencialmente (um a um) cada *time-slot* disponível e determina se ele será alocado para o usuário ou não. Durante esse processo de decisão se aloca ou não cada *time-slot*, o AP utiliza duas variáveis: S_n, que representa o número de *time-slots* que falta para serem alocados para o terminal, e S_c, que representa o número de *time-slots* disponíveis que ainda faltam para serem examinados. Cada *time-slot* disponível é examinado e é atribuído ao usuário com probabilidade $P_a = S_n/S_c$; depois, o valor de S_c é decrementado e, caso o *time-slot* tenha sido atribuído, o valor de S_n também é decrementado; o processo é repetido, *slot* a *slot*, até o último disponível. Na Figura 14 representa-se a alocação de 4 *time-slots* para um usuário, em um quadro de 15 *time-slots*, sendo inicialmente (no quadro N-1) 10 *time-slots* disponíveis e 5 reservados. Na seqüência ilustrada na figura, o *time-slot* é atribuído sempre que a relação S_n/S_c atinge o valor 1/2 (50%). Observa-se que no quadro N, estarão reservados para o usuário os *slots* 4, 7, 11 e 14.

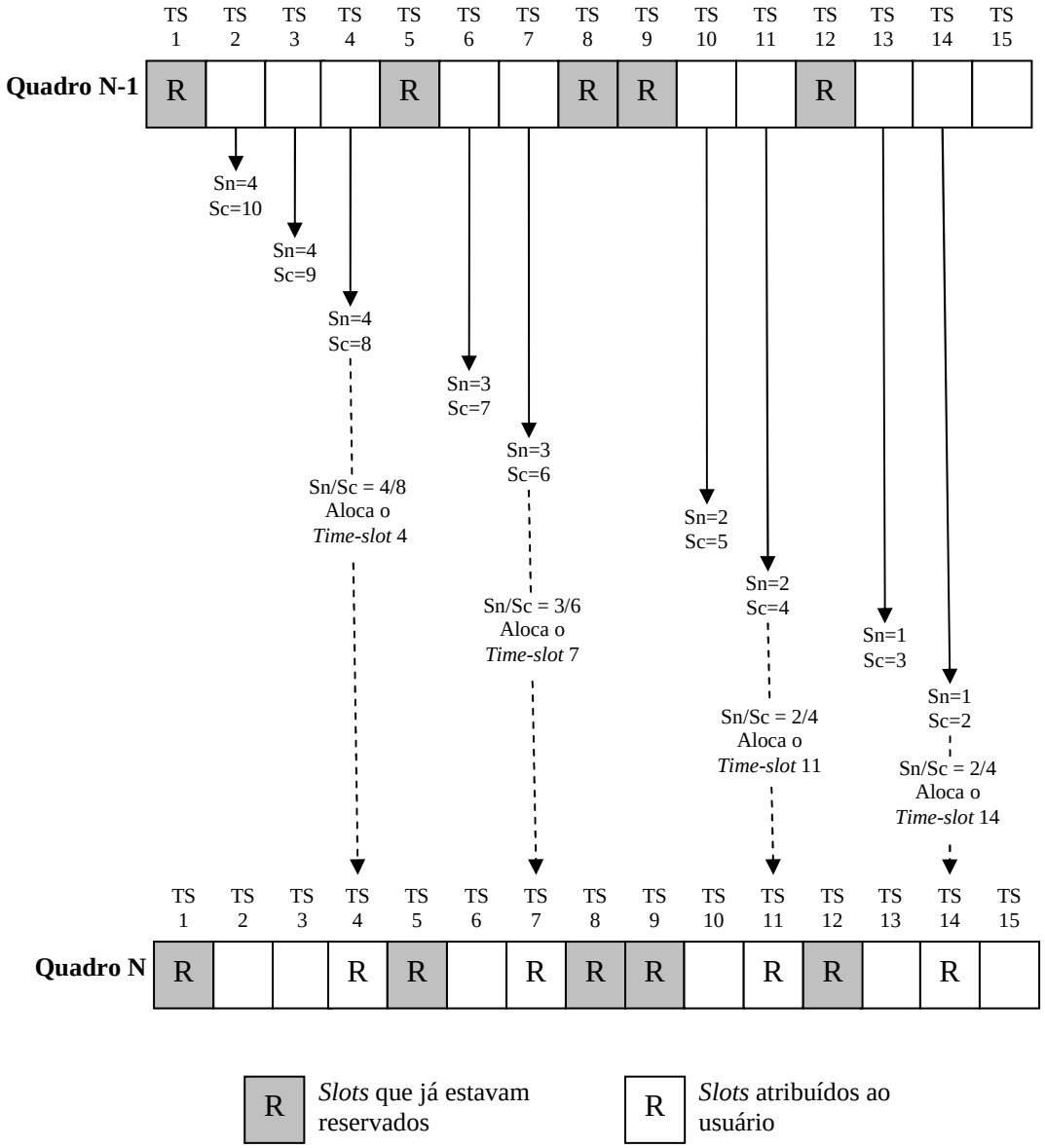

Figura 14: Atribuição de quatro time-slots para um usuário

O número de *time-slots* reservados para um usuário pode ser diminuído, quando ele reduz a taxa requerida ou quando é necessário atribuir *time-slots* a um usuário mais prioritário. Em ambos os casos, o AP escolhe os *time-slots* a serem liberados de forma que as alocações do usuário se mantenham espalhadas aleatoriamente pelo quadro.

Nesse processo de decisão sobre quais *time-slots* devem ser liberados, o AP examina seqüencialmente (um a um) cada *time-slot* reservado para o usuário e o libera com

probabilidade $P_d = S_d / S_r$, onde S_d é o número de *time-slots* que faltam para serem liberados e S_r é o número de *time-slots* que faltam para serem examinados; depois de examinar cada *time-slot*, liberando-o ou não, o algoritmo atualiza as variáveis S_d e S_r e passa para o *time-slot* seguinte. Na Figura 15 é representada uma redução dos *time-slots* reservados para um usuário, de 8 *time-slots* para 4. Na seqüência ilustrada na figura, o *time-slot* é liberado sempre que a relação S_d/S_r atinge o valor 1/2 (50%). Observa-se que no quadro N, os *slots* 2, 6, 11 e 13 estarão disponíveis.

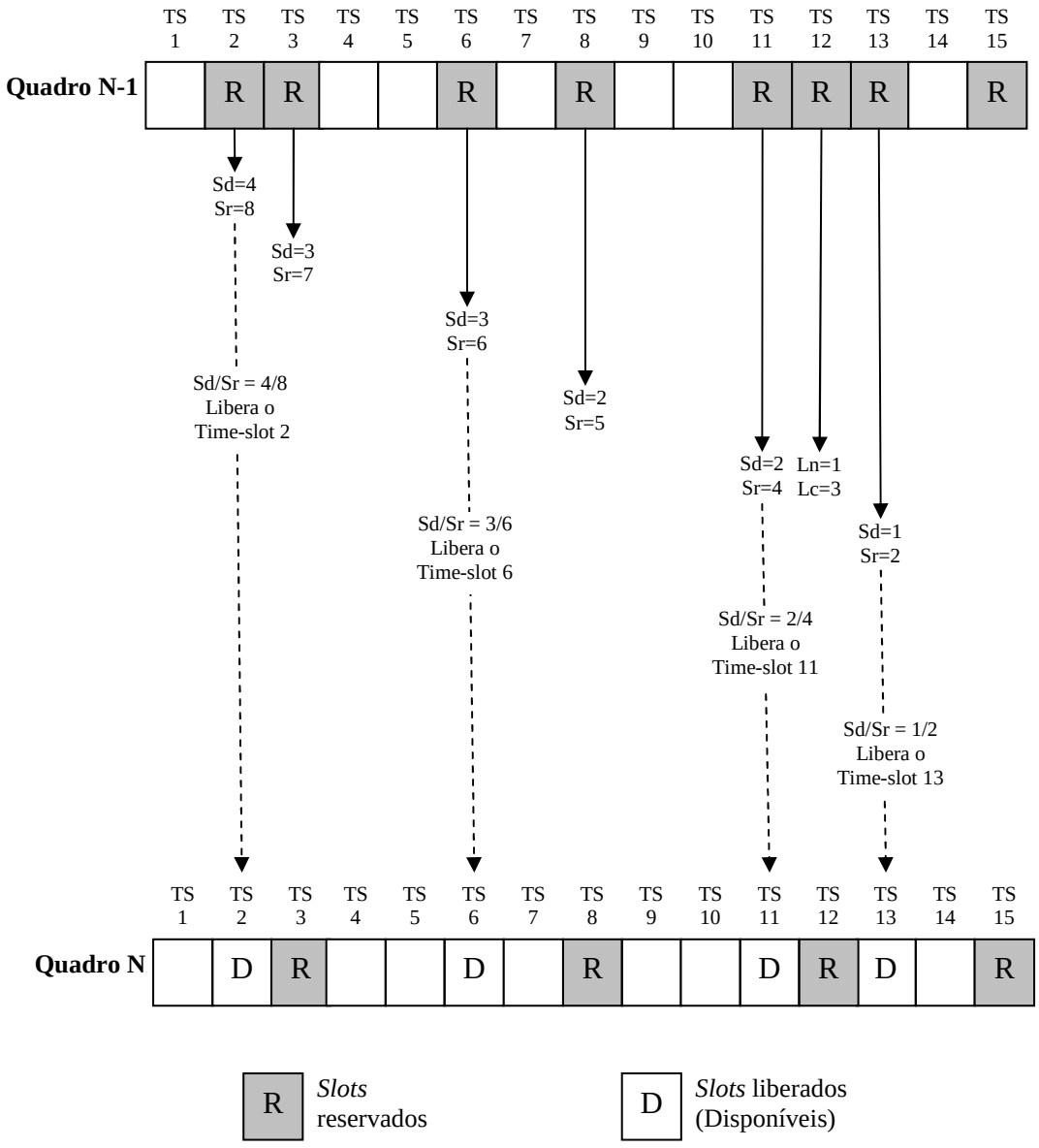

Figura 15: Liberação de 4 time-slots de um usuário que tinha 8 time-slots reservados

Depois de desalocar *time-slots* de um usuário que terminou a transmissão ou reduziu a sua requisição de taxa, o AP verifica se existem usuários aguardando por *time-slots* adicionais. Nesse caso, são considerados primeiro os usuários mais prioritários e depois, se sobrarem *time-slots*, os menos prioritários.

Desempenho do DPRMA

Os autores, em [27] e [28], investigaram o desempenho do DPRMA considerando tráfego de voz, vídeo conferência e dados. Foi utilizado como parâmetro de comparação um protocolo PRMA modificado para suportar tráfego multimídia, denominado PRMA*. No PRMA*, cada usuário pode reservar múltiplos *time-slots* para atender a sua necessidade de requisição de taxa. Entretanto, cada reserva ou liberação é feita individualmente para cada *time-slot*. Quando um usuário deseja reservar mais de um *time-slot*, a reserva é feita do mesmo modo que no PRMA e as reservas que o usuário já possuía são mantidas; para liberar reservas de um ou mais *time-slots*, o usuário os deixa vazios, como no PRMA. O usuário precisa monitorar o número de *time-slots* que ele precisa e o número de *time-slots* já alocados para ele.

Resultados de simulações mostram desempenhos semelhantes dos dois protocolos, quando é considerado somente tráfego de voz, com um usuário requerendo 1 *time-slot* por quadro. Entretanto, com outros tipos de usuários o desempenho do DPRMA é consideravelmente superior.

Testes realizados em [27], através de simulação, mostram que: 1) em um sistema de voz e vídeo conferência, aumentando o número de usuários de vídeo conferência, a superioridade da eficiência do DPRMA, medida em número de usuários de voz suportável, cresce, permitindo admitir 70% mais usuários de voz que o PRMA*; 2) com voz, vídeo e dados, com o DPRMA podem ser admitidos 200% mais usuários de dados que com o PRMA*.

Os resultados em [28] mostram que o DPRMA: 1) apresenta menos perdas de *time-slots* devido a colisões e a perdas de reservas; 2) para diferentes quantidades de usuários de voz e vídeo, ele permite acomodar muito mais usuários de dados, mantendo os

requisitos de qualidade de serviço (*QoS – Quality of service*) dos outros tráfegos; 3) apresenta menores atrasos nas filas de usuários de voz e vídeo, mas a um custo de maior atraso de pacotes de dados; 4) alcança uma vazão maior, de 0,66 a 0,97 contra 0,33 a 0,59 do PRMA*, com o sistema operando na capacidade máxima.

Em um ambiente com desvanecimento, o estado do canal alterna entre “bom” ou “ruim”, com ocorrências freqüentes de perdas de pacotes, quando no estado ruim. Essas perdas de pacotes podem levar a perdas de reservas de *time-slots*, além de comprometer a QoS dos usuários de dados, que aceitam atrasos maiores mas não toleram perdas. Então, em [29] foram propostas duas modificações no DPRMA, para permitir a sua utilização nesse ambiente, considerando usuários de voz, vídeo e dados: na primeira modificação é implementado um esquema de requisição de repetição automática (*ARQ - Automatic Repeat reQuest*) no DPRMA, visando solucionar o problema de perdas de pacotes. A segunda modificação foi proposta para restabelecer as reservas, perdidas durante o canal ruim. Quando um usuário passa por um estado do canal ruim, ele não consegue identificar as informações relativas às reservas dos *time-slots*, então ele adia a transmissão até que o canal melhore. Nesse caso, depois o usuário não sabe se poderá continuar a sua transmissão ou se ele perdeu a reserva de *time-slots*. Então, usuários de dados e de voz utilizam um temporizador (timer) que é inicializado quando o estado do canal se torna bom; se um usuário detectar, antes que o *timer* expire, um *time-slot* reservado com o seu identificador, significa que a reserva continua e ele poderá continuar a sua transmissão, caso contrário, significa que a reserva foi perdida e ele deverá enviar nova requisição. Já o usuário de voz conhece exatamente o *time-slot* que é reservado para ele quadro-a-quadro. Assim, se em tal *time-slot* o AP não informar o identificador do terminal, ele sabe que perdeu a reserva e terá que participar novamente da contenção.

2.4.3.5. PRMA com reserva fixa para Vídeo

Em [30] é proposto um protocolo baseado no PRMA para suportar diferentes tipos de tráfego, como voz, dados e vídeo em tempo real com taxa de *bit* variável (*rt-VBR*), atendendo os requisitos de QoS de cada um. O protocolo proposto reserva no início de

cada quadro um número fixo de *M time-slots* para transmitir alguns dos pacotes de vídeo. Os pacotes de vídeo restantes podem ser transmitidos nos *time-slots* que sobrarem, competindo com pacotes de voz e de dados, seguindo as regras do PRMA básico.

O número de *time-slots* reservados para transmissão de pacotes de vídeo, *M*, deve ser minimizado, para permitir a utilização máxima da largura de banda e deve ser grande o suficiente para manter a probabilidade de descarte de pacotes de vídeo em um nível aceitável. Se o valor de *M* for muito grande, pode ocorrer desperdício dos *time-slots* não utilizados, diminuindo a eficiência do sistema. Por outro lado, valor de *M* muito pequeno também pode levar a menor utilização da largura de banda, devido ao maior número de colisões, com mais pacotes de vídeo competindo nos *time-slots* disponíveis.

Encontrar um valor ótimo de M não é tarefa fácil, pois devem ser consideradas muitas variáveis, como as probabilidades de permissão dos diferentes tipos de tráfego, número de usuários de cada tipo de tráfego, os requisitos de QoS e a capacidade do sistema. Assim, deve-se encontrar uma combinação ideal desses valores que faça o sistema trabalhar na sua capacidade máxima.

Em [30] foram realizados testes para encontrar a melhor combinação do valor de *M* e as probabilidades de permissão de terminais de voz, dados e vídeo. Foi investigado o efeito do valor de *M* sobre o desempenho do sistema, determinado pelo número máximo de usuários que o sistema pode suportar. O estudo levou em consideração as probabilidades de permissão dos tipos de tráfego e os requisitos de probabilidade de descarte de pacotes de voz e vídeo e atraso médio de pacotes de dados.

2.5. DQCA

Em [31], [5], [32] e [6] são apresentados mecanismos de escalonamento *Cross-Layer* para prover tráfego heterogêneo (voz e dados) sobre WLAN, utilizando o protocolo de acesso ao meio DQCA (*Distributed Queuing Collision Avoidance*).

2.5.1. Visão geral do protocolo DQCA

Como pode ser visto em [2], o protocolo de controle de acesso ao meio no padrão IEEE 802.11 é baseado no CSMA/CA (*Carrier Sense Multiple Access/Colision Avoidance* – Acesso múltiplo com detecção de portadora e prevenção de colisão), que usa o algoritmo de *Backoff* exponencial binário para reduzir colisões. Como comentado em [33], sabe-se que esse protocolo é ineficiente, por causa dos períodos ociosos devido ao *backoff* e da alta presença de colisões, na medida em que a carga de tráfego cresce. Isso se torna crítico com aplicações altamente exigentes, como as que envolvem voz, vídeo sob demanda e tráfego multimídia em geral. Portanto, de acordo com os autores em [34], existe uma necessidade de desenvolver novos protocolos MAC que suportem requisitos de alto desempenho em termos de vazão, atraso, variação de atraso (*jitter*), justiça e consumo de energia e também que sejam auto adaptáveis às mudanças de condições do sistema, como a carga de tráfego ou o número de nós.

O DQCA, proposto em [7], é um protocolo de acesso ao meio distribuído, de alto desempenho, projetado para ambientes WLAN, que oferece melhor desempenho que o protocolo DCF implementado no padrão 802.11. Sob condições de tráfego baixo, ele funciona como um mecanismo de acesso aleatório e, na medida em que a carga de tráfego aumenta, passa suavemente e automaticamente para um esquema de reserva.

Nesta seção apresenta-se uma descrição resumida das características do DQCA e seu funcionamento, suficiente para entendimento do mecanismo de escalonamento proposto. Informações detalhadas sobre o DQCA podem ser encontradas em [7], [35] e [6].

2.5.1.1. Estrutura do quadro DQCA

O quadro DQCA consiste de três campos, como pode ser visto na Figura 16. O primeiro é o campo de acesso, ou janela de contenção, que é dividido em *m mini-slots* de controle, durante os quais os nós podem requerer acesso ao canal enviando requisições

de acesso (*RTS – Request to send*); o segundo campo é o campo de dados (*Data Slot*), para transmissão de dados com carga útil, e o terceiro o pacote de realimentação (*FBP – Feedback packet*), que é transmitido pelo AP a todos os nós e contém o *feedback* necessário para que cada nó fique sempre ciente dos estados das filas e suas posições nas mesmas. O campo FBP contém as CTS (*clear to send* – livre para envio), contendo informações sobre o estado de cada *minislot* de controle, indicando sucesso, colisão ou vazio, a confirmação (*ACK – acknowledgment*) para o pacote transmitido no campo de dados e informações adicionais que podem ser incluídas, como as taxas de transmissão disponíveis dos usuários ativos. São inseridos pequenos intervalos SIFS (*short inter frame space*) a cada alternância entre transmissão no sentido *uplink* e *downlink*, ou seja, após o campo de Dados e após o pacote de realimentação.

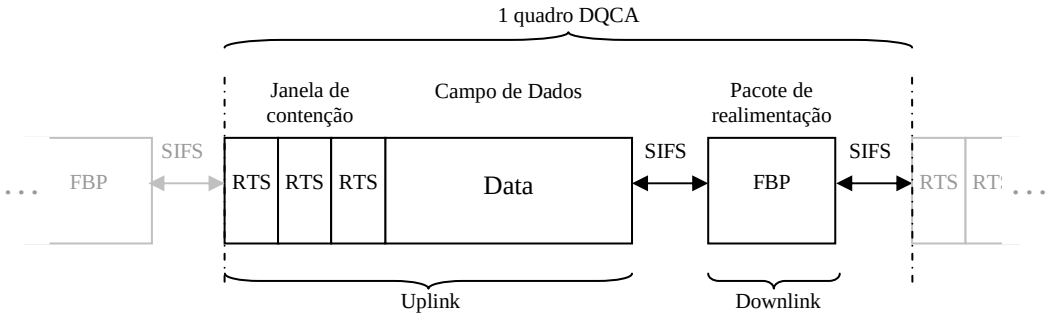

Figura 16: Estrutura do quadro DQCA

O protocolo DQCA oferece acesso aos usuários em um intervalo de tempo reservado pelas requisições de acesso (RTS), confinando as colisões somente nessa parte do quadro [6], e o campo de dados é destinado à transmissão de dados livre de colisões. As RTSs têm um formato específico, suficiente para que seja detectada pelo AP e possa ser usada para a detecção do acesso e a estimação da condição do enlace. Em [36] e [37], é comprovado que 3 *minislots* de controle ($m = 3$) são suficientes para garantir o desempenho do sistema próximo de ótimo com qualquer carga de tráfego, pois, como as RTS são transmitidas em um intervalo de tempo muito pequeno, o processo de resolução de colisões, com $m \geq 3$, trabalha mais rápido do que a transmissão de dados. Valores maiores de m podem resultar em atrasos ligeiramente mais baixos nas contenções, a um custo de *overhead* maior.

2.5.1.2. Funcionamento do DQCA

Segundo os autores em [7], apesar de o DQCA usar um nó coordenador, ele trabalha de forma distribuída, pois o AP atua simplesmente como detector das RTSs e ACKs e um repetidor, a fim de difundir essas informações para o resto dos nós. Por isso, qualquer nó na rede poderia executar essa tarefa de coordenação, dispensando um AP ou qualquer estrutura centralizada. As regras do protocolo são executadas em cada terminal de forma independente, assíncrona e completamente distribuída.

Conforme ilustrado na Figura 17, o DQCA mantém duas filas distribuídas: a fila de transmissão de dados (*DTQ – Data transmission queue*) que é empregada no escalonamento da transmissão de pacotes, e a fila de resolução de colisões (*CRQ – Collision resolution queue*). A cada quadro, o nó na cabeça da DTQ está apto a transmitir seu pacote, enquanto os nós que estiverem na saída da CRQ tentarão resolver a colisão. Dessa forma, os processos de transmissão de dados e de resolução de colisões trabalham em paralelo.

Quando um nó tem uma nova mensagem para transmitir, ele observa os estados das filas para verificar se está apto a enviar uma requisição. Para enviar uma requisição ele escolhe aleatoriamente um dos *m minislots* de controle e envia uma RTS. Em cada *minislot* de contenção, um entre três eventos podem ocorrer: nenhuma tentativa de requisição, sucesso na requisição por um único nó, ou requisições simultâneas por mais de um nó, gerando colisão. No caso de sucesso, o nó que enviou a requisição entra na DTQ e aguarda sua vez de transmitir a mensagem; se houver colisão, os nós envolvidos entram na CRQ e no próximo quadro é aplicado o algoritmo de resolução de colisão. Quando a DTQ e a CRQ estão vazias, um controle de acesso semelhante ao *Aloha* toma lugar, com o objetivo de reduzir o atraso de acesso. Nesse esquema, o nó envia a RTS normalmente e transmite imediatamente o pacote no mesmo quadro, sem a requisição prévia do canal. Isto pode permitir colisões em transmissões de dados, mas em contrapartida diminui o atraso quando o sistema estiver com carga de tráfego baixa. Na Figura 17 apresenta-se o processo de resolução de colisões do DQCA.

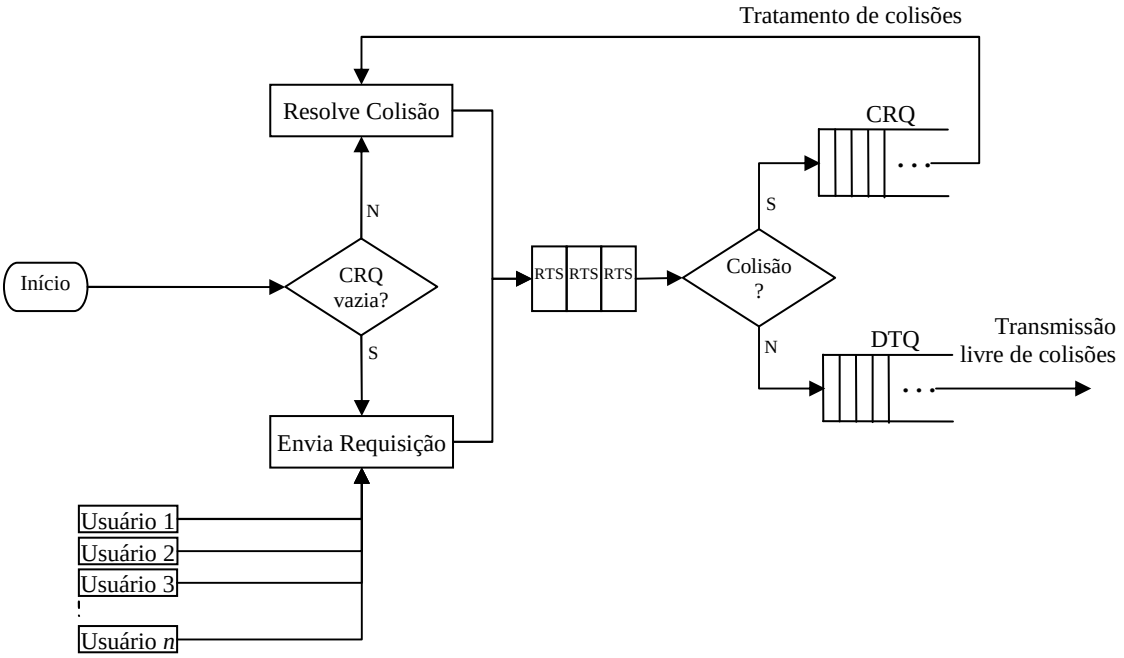

Figura 17: Fluxograma da resolução de colisões em um quadro DQCA

Conforme comentado anteriormente, nós que enviam requisições de acesso em um mesmo *minislot*, gerando colisão, entram na mesma posição da CRQ. No algoritmo de resolução de colisões, os nós que estiverem na cabeça da CRQ selecionam aleatoriamente um *minislot* de controle no próximo quadro e reenviam uma RTS; os nós que enviarem as RTS's com sucesso entram na DTQ e no caso de nova colisão os nós envolvidos entram novamente no final da CRQ e o processo se repete até que a CRQ fique vazia. Para evitar instabilidade, os usuários com mensagens novas não podem enviar requisições enquanto a CRQ não estiver vazia.

As informações relativas às filas não são transmitidas diretamente pelo AP, mas processadas em cada nó ao final de cada quadro, com base em parâmetros alimentados pelo AP através no pacote FBP. As filas são representadas nos nós por quatro valores inteiros, *TQ, RQ, pTQ* e *pRQ*. *TQ* representa o número de nós na DTQ aguardando para transmissão, enquanto *RQ* é o número de colisões na CRQ aguardando a resolução. Os parâmetros *pTQ* e *pRQ* representam a posição do nó na DTQ e na CRQ, respectivamente. Vale notar que os parâmetros *TQ* e *RQ* têm o mesmo valor para todos os nós, enquanto *pTQ* tem valores distintos para cada nó e *pRQ* tem o mesmo valor para nós que colidiram no mesmo *mini-slot*.

As decisões de escalonamento não são tomadas pelo AP, mas cada usuário sabe quando transmitir, calculando e mantendo as variáveis citadas e seguindo algumas regras predefinidas, com base em informações contidas no campo FBP.

Conforme apresentado em [6], quando o AP informa através do campo FBP que uma solicitação (RTS) foi bem sucedida, cada nó incrementa sua variável *TQ* (tamanho da fila de transmissão) em uma unidade, enquanto o nó que enviou a RTS define o valor da variável *pTQ* (posição na fila de transmissão) com o valor de *TQ*, já incrementado, ocupando assim o último lugar na DTQ. No caso do AP informar sobre colisão em uma RTS, todos os nós incrementam suas variáveis *RQ* (tamanho da fila de resolução de colisões) e os nós envolvidos na colisão passam a ocupar a última posição na CRQ definindo o valor da variável *pRQ* com o valor de *RQ*. Além disso, quando o AP envia uma mensagem de confirmação da transmissão de um pacote com o *bit* de final de mensagem marcado com 1 (*true*), todos os nós decrementam em uma unidade o valor da variável *TQ* e todos os terminais com *pTQ* maior que zero decrementam o seu valor, avançando uma posição na fila de transmissão. Assim, a variável *pTQ* indica de certa forma o tempo de permanência do nó na fila de transmissão, visto que é inicializada com o tamanho da fila e decrementada ao finalizar a transmissão de cada mensagem. A forma como as variáveis citadas são mantidas e interpretadas pode ser vista com detalhes em [6].

A transmissão de dados é executada de acordo com a disciplina da fila de transmissão de dados, cujo padrão no DQCA básico é FIFO (*First-In-First-Out* – Primeiro a entrar, primeiro a sair). De acordo com os autores em [6], [5] e [35], mensagens longas são fragmentadas em pacotes menores, que são armazenados nos *buffers* dos terminais e transmitidos um a um em quadros consecutivos, utilizando uma única requisição. Cada pacote possui um *bit* de final de mensagem, que indica se ele é o último pacote de uma mensagem (1 = *true*) ou não (0 = *false*). Isso possibilita a transmissão de todos os pacotes de uma mesma mensagem com uma única requisição. No entanto, se o atraso de propagação no sistema impede que o terminal receba a informação sobre o final da mensagem antes do início do próximo *slot*, ocorre a perda de um *slot* vazio no final de cada mensagem. Nesse caso, se é sabido que as mensagens geralmente são curtas,

segundo os autores em [35], deveria ser possível e conveniente desligar esse mecanismo e considerar todas as mensagens formadas por um único pacote.

Na Figura 18 apresenta-se esquematicamente uma seqüência de quadros DQCA, ilustrando quadro a quadro os estados das filas de transmissão e de resolução de colisões, DTQ e CRQ, as chegadas de novos pacotes nos terminais, assim como as transmissões das requisições de acesso e de pacotes de dados no sentido *uplink* e dos pacotes de realimentação, FBP, no sentido *downlink*. Observa-se que o pacote de realimentação informa para cada *minislot* um valor ‘s’, ‘i’ ou ‘c’, indicando sucesso no *minislot*, *minislot* vazio (*idle*) ou colisão no *minislot* respectivamente; além disso, um campo *Ack* indica que o pacote foi transmitido com sucesso e um valor 0 ou 1 para o *bit* de final de mensagem replica para todos os terminais o valor recebido no campo Data. Inicialmente, a fila de resolução de colisões encontra-se vazia e a fila de transmissão contém os terminais T1, T2 e T3, sendo que T1 ocupa a cabeça da fila. Além disso, os terminais T4, T5, T6 e T7 possuem novas mensagens para serem transmitidas devendo assim participar da contenção. A seqüência dos quadros ilustrados na Figura 18 é descrita a seguir:

Quadro n:

- **Requisições**: no primeiro quadro (n), os terminais que possuem novas mensagens enviam requisições. O terminal T4 utiliza o primeiro *minislot* e os terminais T5, T6 e T7 o terceiro *minislot*. Com isso, para o próximo quadro o terminal T4, que teve sucesso na requisição, entrará no final da DTQ enquanto os terminais T5, T6 e T7 que escolheram o mesmo *minislot* e portanto se envolveram em uma colisão, entrarão na cabeça da CRQ para a tratarem.

- **Transmissão do pacote (Campo Data):** O terminal que ocupa a saída da DTQ, T1, Transmite seu pacote e sinaliza o *bit* de final de mensagem com 0 (zero), indicando que não é o último pacote. Assim ele permanece na cabeça da DTQ.

Quadro n + 1:

- **Requisições**: os terminais T8 e T9, que possuem novas mensagens não podem enviar requisições, visto que a CRQ não está vazia. Dos terminais que ocupam a saída da CRQ, T5 escolhe o primeiro *minislot* entrando assim na DTQ, enquanto os terminais T6 e T7 que escolheram o mesmo *minislot* entram novamente na CRQ para tratarem a colisão.

- **Transmissão do pacote (Campo Data):** O terminal que ocupa a saída da DTQ, T1, Transmite seu pacote e sinaliza o *bit* de final de mensagem com 1 (um), indicando que é o último pacote da sua mensagem. Assim, no quadro seguinte ele não aparece mais na DTQ.

Quadro n + 2:

- **Requisições**: os terminais T8 e T9 ainda não podem enviar requisições, visto que a CRQ não está vazia. Dos terminais que ocupam a saída da CRQ, T7 escolhe o segundo *minislot* entrando assim na DTQ e T6 escolhe o terceiro *minislot,* entrando na próxima posição da fila.

- **Transmissão do pacote (Campo Data):** O terminal T2 que passou a ocupar a saída da DTQ transmite seu pacote e sinaliza o *bit* de final de mensagem com 0 (zero), permanecendo assim na saída da fila.

Quadro n + 3:

- **Requisições**: como a CRQ agora está vazia, os terminais T8 e T9 enviam requisições no primeiro e terceiro *minislots* respectivamente, entrando assim nas próximas posições da DTQ para o próximo quadro (n + 4), visto que hão houve colisão.

- **Transmissão do pacote (Campo Data):** O terminal T2 que ocupa a saída da DTQ transmite seu pacote e sinaliza o *bit* de final de mensagem com 0 (zero), permanecendo assim na saída da DTQ.

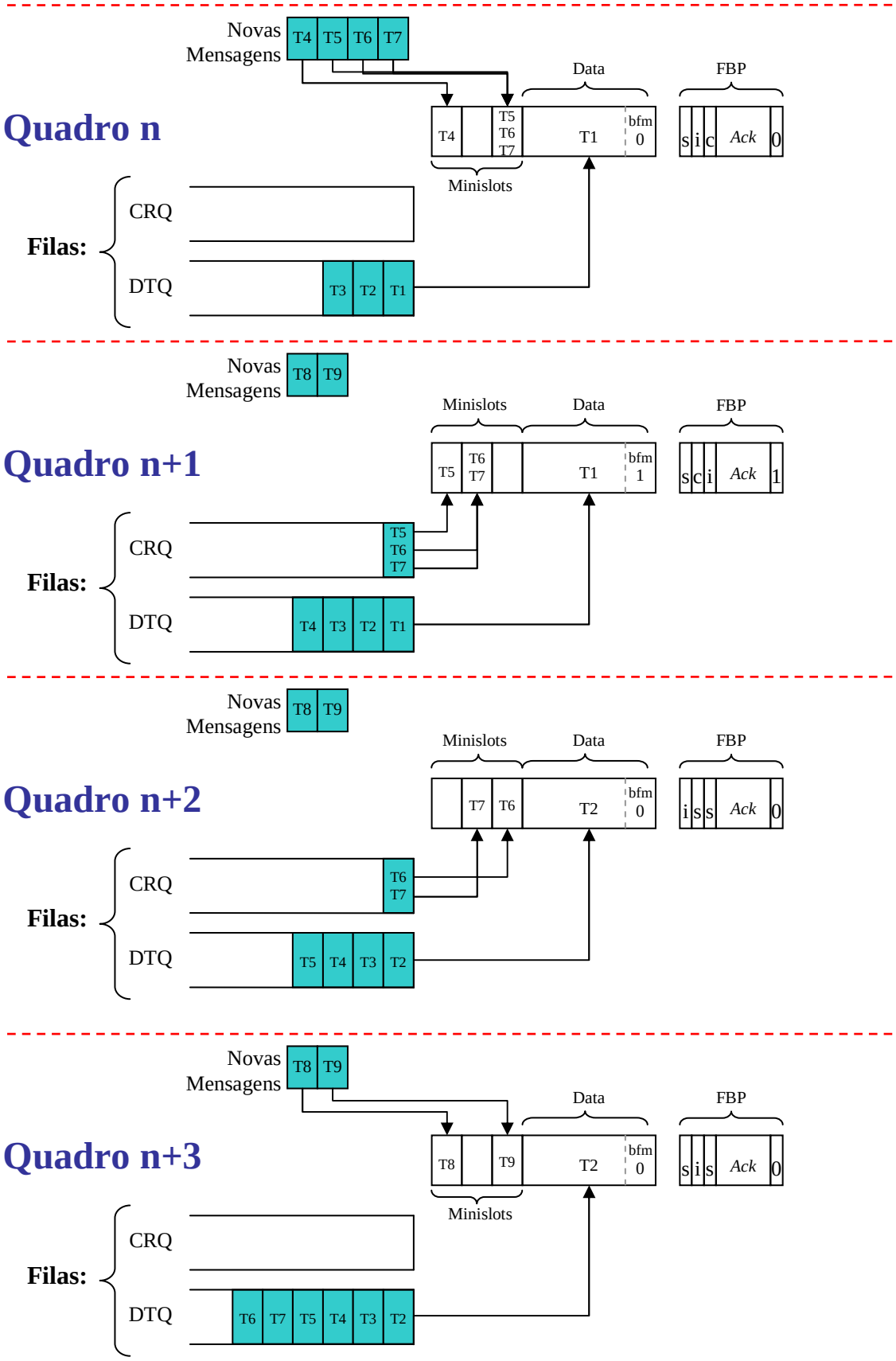

Figura 18: Seqüência de quadros DQCA

2.5.2. Algumas variantes do DQCA propostas

Na proposta original do DQCA, considerou-se a disciplina de serviço FIFO, na qual o usuário na primeira posição da DTQ tem sempre permissão para transmitir. Entretanto, outros parâmetros podem ser considerados, como o tipo de serviço ou a qualidade do sinal, para definir a ordem de transmissão dos usuários. Em sistemas com tráfego heterogêneo, os requisitos de QoS de cada tipo de tráfego devem ser satisfeitos e para isso, a informação sobre o tipo de tráfego deve ser considerada na decisão de escalonamento. Por exemplo, no caso de tráfego de voz e dados, o serviço de voz deve ser mais prioritário, pois é mais sensível ao atraso.

Assumindo que a camada física suporta transmissão multi-taxa, o melhor estado do canal corresponde à taxa de transmissão mais alta. Assim, o conhecimento sobre os estados dos enlaces entre cada terminal e o AP pode ser utilizado para garantir transmissão a taxas mais altas e, conseqüentemente, uma maior utilização da largura de banda, ao priorizar a transmissão de nós com melhores condições do canal. Enquanto os pacotes com carga útil são transmitidos a taxas mais altas, todos os pacotes de sinalização e controle (RTS, CTS, ACKs, etc), são transmitidos a uma taxa mínima, para garantir transmissão confiável.

Os autores em [31], [5], [32] e [6] apresentam técnicas *Cross Layer* para WLAN que utilizam o protocolo DQCA. Todos essas técnicas mantêm o esquema de reserva de acesso e o algoritmo de colisão, entretanto, para incorporar tais mecanismos, algumas modificações foram feitas na operação do DQCA: para permitir que alguns esquemas façam a distinção de tipos de serviço, as RTS passam a transportar diferentes tipos de sinais, através de diferentes tipos de rajadas predefinidas, relacionadas a cada tipo de serviço, e essa informação é incluída no FBP, para que cada nó tenha conhecimento. Para que seja possível considerar o estado do canal nas decisões de escalonamento, o AP estima a taxa de transmissão disponível (ótima) de cada nó, calculando a relação sinal/ruído (*SNR – Signal-to-Noise Ratio*) das RTS´s recebidas. Então, incorpora-se no pacote FBP um vetor com os valores das taxas de transmissão disponíveis, R_b, de todos os nós presentes na DTQ, sendo esse vetor ordenado pela mesma ordem que os nós se encontram na fila. Isso permite que cada nó conheça a sua própria taxa e as taxas dos

outros nós na DTQ e saibam em quais posições da DTQ estão os nós com as melhores taxas. Segundo proposta em [38], considerando quatro valores de taxas disponíveis possíveis de 1, 2, 5.5 e 11 *Mbps*, como na especificação do padrão IEEE 802.11b, R_bs podem ser representadas por 2 *bits*, gerando assim um *overhead* adicional de 2 x *TQ bits*, arredondado para o próximo número inteiro de *bytes*. Para que o AP possa estimar a taxa de transmissão ótima de cada nó, é utilizado um diálogo *Cross-Layer* entre as camadas física e MAC. Quando o AP recebe uma RTS de algum usuário, ele mede a potência do sinal recebido e calcula a SNR do enlace, que define a taxa máxima disponível, R_b.

2.5.2.1. Alguns Esquemas Oportunistas sem considerar o tipo de serviço

Em [31] são propostos dois mecanismos *Cross Layer* (CL1 e CL2) que se diferenciam pelo modo como fazem uso das informações do vetor de taxas disponíveis.

A primeira técnica (CL1) explora o fato de que o tempo de acesso ao canal é inversamente proporcional à taxa, para prover uma utilização mais eficiente da largura de banda. Nesse mecanismo, o nó com maior taxa disponível, R_b, terá a oportunidade de transmissão, e, caso dois ou mais nós possuam o mesmo valor de R_b, serão consideradas as idades dos nós na fila, definidas por suas variáveis *pTQ*. Como o vetor de taxas disponíveis dos nós, que é incorporado ao pacote FBP, apresenta os nós na mesma ordem em que se encontram na DTQ, cada nó sabe exatamente quando ele é o terminal com maior valor de R_b, e, conseqüentemente poderá iniciar a sua transmissão no quadro seguinte.

Na segunda técnica (CL2) a ordem de prioridade de transmissão é determinada por uma função de prioridade virtual (*VP – Virtual Priority*), sendo que em cada quadro o nó com maior valor de *VP* irá transmitir. No final do quadro, cada nó na DTQ calcula o seu valor de *VP* e os valores de *VP* dos outros nós da fila, seguindo uma fórmula definida cuja expressão pode ser selecionada com diferentes critérios, de acordo com os

objetivos fixados. Em geral, a seleção dessa função pode ser feita de muitas formas, dependendo da disponibilidade de parâmetros da camada física e da métrica de desempenho desejada, como por exemplo, vazão, atraso ou justiça.

Na implementação em [31], a função *VP* considera dois parâmetros, a taxa de *bits* disponível R_b e a idade do pacote, indicada pela sua posição na fila *pTQ*:

$$VP = \frac{Rb_i}{pTQ_i} \tag{1}$$

A idéia é permitir a transmissão de usuários com taxa de *bits* disponível mais alta, sem ignorar seu tempo de espera na fila. Para controlar o grau de contribuição dos parâmetros Rb_i e pTQ_i na decisão de escalonamento, pode ser utilizado um fator de ponderação apropriado.

Resumindo, enquanto o primeiro esquema proposto (CL1) provê melhora na vazão da rede e uma redução significativa no atraso médio de pacotes, o segundo (CL2) ainda oferece algum tipo de justiça, embora nenhum dos dois esquemas leve em conta o tipo de serviço.

Em [32] foi demonstrado, através de simulação, que com a utilização do protocolo DQCA aplicando conceitos *Cross-Layer*, dando prioridade de transmissão à nós com melhores condições do canal, é alcançada uma considerável redução no atraso médio de pacotes e um aumento na vazão. Nos testes foram considerados diferentes cargas oferecidas (*Mbps*) e tamanhos de pacotes de 100, 570 e 2312 *bytes*. Para pacotes de 2312 *bytes* pode ser obtida uma melhora na vazão acima de 75% sobre a vazão máxima obtida com DQCA básico, que não considera o estado do canal.

2.5.2.2. Alguns Esquemas que consideram o tipo do serviço

De acordo com [5] e [6], o serviço de voz pode tolerar um atraso máximo de 300 *ms* por pacote e pacotes que não forem transmitidos dentro desse limite, desde a sua geração, devem ser descartados; entretanto, o número de pacotes descartados dessa maneira deve ser mantido abaixo de 1%, para proporcionar um nível de QoS aceitável. Segundo os autores em [39], os pacotes de dados não aceitam perdas, mas podem tolerar atrasos consideravelmente grandes; esses pacotes são descartados somente no caso de estouro do buffer do terminal.

Para permitir que os escalonadores considerem os tipos de serviço na decisão de escalonamento, as alterações feitas no DQCA para os esquemas anteriores são mantidas e, adicionalmente, o parâmetro *ServiceType* (Tipo do serviço) de todos os nós da fila são inseridos no pacote FBP. Para dois tipos de serviço, por exemplo, é utilizado um *bit* extra por nó.

Os autores em [5] propõem dois esquemas *Cross Layer* (CL1 e CL2), que levam em consideração os tipos de serviço. São implementados algoritmos que priorizam o tráfego de voz e encorajam a transmissão de nós com taxas mais altas, otimizando a utilização da largura de banda, proporcionando o aumento no número de nós de voz simultâneos e melhorando a vazão do sistema.

Na Proposta CL1, em vez de uma fila para gerenciar as transmissões, são utilizadas *p* filas, sendo uma fila para cada tipo de serviço, com diferentes níveis de prioridade de acordo com as restrições de atraso. Em cada quadro, é dada garantia de transmissão ao nó na cabeça da fila não vazia de maior prioridade, assim, nós de uma fila menos prioritária só são atendidos depois que as filas mais prioritárias estiverem vazias. No caso de serviços de voz e dados são usadas duas filas, com prioridade para a fila de voz.

Para promover transmissões a taxas mais altas, os nós em cada fila são ordenados pelas respectivas R_b`s, com os nós com taxas maiores ocupando as cabeças de suas filas. No

caso de mesma taxa, os tempos de espera dos nós, representados pelos valores de *pTQ*, são considerados.

A segunda técnica, CL2, usa somente uma fila de transmissão e, assim como no segundo esquema proposto em [31], utiliza uma função VP para determinar a ordem de transmissão dos usuários. Em cada quadro o nó com *VP* maior terá prioridade, e no caso de mais de um nó com a mesma *VP*, a prioridade é dada ao nó com tempo de espera na fila maior.

Em uma abordagem para prover justiça e melhorar o desempenho do sistema, por exemplo, a função *VP* foi implementada em [5] como na Eq.(2):

$$VP = \alpha(1 + ServiceType_i) + (1 - \alpha)\left(\frac{Rb_i}{pTQ_i}\right), \text{ onde...} \qquad (2)$$

$ServiceType_i$: 0 (zero) para Dados e 1 (um) para Voz.

Rb_i: Taxa de *bits* disponível na qual o nó pode transmitir. Como está no numerador na expressão, a Taxa disponível mais alta obviamente leva à maior valor de *VP*.

pTQ_i: É a posição do nó *i* na DTQ. Representa o tempo de espera na fila, lembrando que valores menores de *pTQ* indicam maior tempo de espera. Influencia na justiça, pois aumenta as chances de transmissão de usuários que esperam por mais tempo na fila, independentemente de suas taxas disponíveis ou tipos de serviço.

α: Fator de ponderação. Valor entre 0 e 1, utilizado para definir o nível de influência que os parâmetros exercem no cálculo de VP e, conseqüentemente, na decisão de escalonamento. Valor mais alto aumenta a influência do tipo de serviço no escalonamento, enquanto valor baixo faz a taxa disponível e a idade do pacote na fila os principais critérios.

Além do tipo do serviço, os outros parâmetros necessários para o cálculo da *VP*, Rb_i e pTQ_i, podem ser obtidos pelos terminais no pacote FBP através do vetor de valores R_b e da ordem dos nós no vetor. Como comentado anteriormente, a posição do terminal no vetor de valores de R_b corresponde à posição do terminal na DTQ.

As duas propostas (CL1 e CL2) permitem a priorização pelo tipo de serviço, pela taxa e o tempo de espera, mas enquanto na proposta CL1 a priorização é nessa ordem exata, no esquema CL2 a influência de cada critério é determinada pelo fator de ponderação configurável, permitindo assim combinar desempenho com justiça. Testes realizados em [5] para diferentes valores de α concluem que: valores altos de α melhoram o serviço de voz, com prejuízo dos serviços de dados, enquanto valores baixos dão um desempenho mais balanceado.

Em [5], foram realizados vários testes de desempenho das duas técnicas, CL1 e CL2, utilizando como parâmetros de comparação, o DQCA básico e o protocolo de acesso DCF do padrão IEEE 802.11b. Os testes realizados através de simulação concluem que: com a técnica CL1, considerando diferentes números de nós de voz, a vazão total (voz e dados) foi sempre consideravelmente superior à CL2, ao DQCA básico e ao 802.11b, que apresentou uma vazão muito inferior aos outros mecanismos. Além disso, comparando com o DQCA, com um número fixo de usuários de voz, a carga de dados que pode ser suportada pelo sistema é aumentada em 109,4% com a técnica CL1, contra um aumento de 72% com a CL2. Mantendo a garantia de um nível de *vazão* total, as duas propostas permitem acomodar um número bem maior de usuários de voz, sendo esse número ligeiramente superior com a técnica CL1.

A técnica CL1 permite um ganho considerável em *vazão* e número de usuários de voz, entretanto, apresenta problemas de justiça entre os serviços e entre nós do mesmo serviço. Ela reduz consideravelmente o atraso médio de voz, com um aumento no atraso de dados que pode ser muito alto, com o aumento da carga oferecida. Assim, segundo o autor, a proposta CL2 pode dar uma solução mais balanceada dos atrasos de voz e dados, permitindo combinar desempenho com justiça, considerando os limites de atraso de pacotes de voz e dados.

Em [6] foi apresentado um mecanismo semelhante ao CL1, com a diferença que na fila de voz o serviço é de acordo com a ordem de chegada, não considerando as taxas disponíveis para a decisão de escalonamento. O desempenho desse esquema foi comparado com o do DQCA básico e os resultados obtidos por simulação concluem que esse mecanismo permite alcançar um aumento considerável na vazão de dados sobre o DQCA básico, reduzindo perdas de pacotes de voz devido a atrasos excessivos e que, com a estratégia de priorizar usuários de dados com melhores condições do canal, o aumento da vazão de dados sobre o DQCA básico chegou a 133% e houve um aumento no número total de usuários de voz que o sistema pode acomodar, com a mesma carga de tráfego de dados suportável pelo sistema.

2.5.2.3. Outras implementações, problemas levantados e propostas de soluções

Nas técnicas apresentadas, o AP determina as taxas disponíveis dos terminais, estimando os estados dos canais dos usuários através do cálculo das SNR's das RTS's recebidas. No instante em que um terminal envia a RTS, esse valor é real, entretanto, devido à variabilidade do meio sem Fio [4], o estado do enlace pode se alterar enquanto o terminal aguarda nas filas ou enquanto envia seus pacotes. Assim, o valor da taxa disponível pode passar a não refletir o estado atual do enlace e isso pode levar a degradação do desempenho do sistema. O usuário pode tentar transmitir a taxas mais altas do que a ideal, ocasionando problemas como erros na transmissão e retransmissões, ou então o usuário pode transmitir a taxas mais baixas, reduzindo a utilização da largura de banda.

Em [40] e [38] apresentam-se um mecanismo que provê atualizações freqüentes nas informações sobre taxas de transmissão disponíveis dos usuários, através de transmissões periódicas de quadros especiais dedicados à medição do estado do canal. Em [34] propõe-se notificações periódicas dos valores de *TQ* e *RQ* através do pacote FBP, para permitir que nós entrem e saiam do sistema e para evitar erros na manutenção das filas.

Capítulo 3 - Uma nova proposta de protocolo – DQCA-RP

3.1. Introdução e motivação

De acordo com [41], alguns estudos mostram que, durante uma conversa uma pessoa fala somente cerca de 40% do tempo. Então, os períodos de silêncio entre as locuções sugerem que poderia ser alcançada uma utilização maior da largura de banda disponível, transmitindo somente quando um pacote de voz estiver disponível. Como visto anteriormente na Seção 2.4, o protocolo de acesso ao meio PRMA proposto em [20] reserva *time-slots* periodicamente para terminais de voz, permitindo que eles compartilhem o canal de forma semelhante ao TDMA, enquanto os *time-slots* não reservados podem ser utilizados pelos terminais de dados, para transmissão dos seus pacotes. Entretanto, como comentado em [22], o PRMA apresenta algumas limitações, como a ineficiência devido à reserva de acesso, que é feita através do protocolo *Slotted-Aloha*, e a instabilidade quando a carga do tráfego é alta. Mesmo assim, o autor em [42] mostra que o PRMA pode suportar até 1.6 conversações por sub-canal, o que representa uma melhora considerável sobre o TDMA, com 1 conversação suportada.

O protocolo DQCA, apresentado na Seção 2.5, é um protocolo baseado no DQRAP (*Distribute Queueing Random Access Protocol*) apresentado em [37], [43] e [44], de alto desempenho e estável, que mantém uma vazão alta com qualquer carga de tráfego. Entretanto, ele pode permitir variações muito altas no atraso, que é um comportamento indesejável para tráfego com CBR, como o de voz, principalmente quando mensagens muito grandes ocupam o canal por longos períodos.

Em [41] foram feitos testes através de simulações por computador, comparando o desempenho do PRMA com o do DQRAP para tráfego de voz. Os resultados das simulações mostram que, com o DQRAP são suportadas 2,0 conversações por sub-canal, contra as 1,6 do PRMA; além disso, enquanto no PRMA pacotes podem ser descartados sob qualquer carga de tráfego, com o DQRAP pacotes não são descartados sob carga de tráfego abaixo de 1,5 conversações aproximadamente. Outro resultado importante é que o DQRAP alcança o topo de 2,0 conversações por sub-canal com um limite de atraso de 10*ms*, enquanto o PRMA suporta 1,2 conversações com esse limite de atraso.

Nesse trabalho é proposto um protocolo baseado no DQCA, denominado DQCA-RP (DQCA com reserva periódica), que emprega no DQCA um conceito de reserva de *time-slots* semelhante ao protocolo PRMA, com a proposta de manter-se a alta vazão alcançável com o DQCA e, com a reserva de *time-slots* para terminais de voz, reduzir o atraso médio e a variação do atraso desses pacotes a valores consideravelmente baixos independente da carga de dados submetida, além de reduzir os descartes destes pacotes devido a violação dos requisitos de QoS.

3.2. Descrição do protocolo

No DQCA básico, o nó que estiver na saída da fila de transmissão, DTQ, sempre tem a permissão para transmitir. Então ele divide a sua mensagem em pacotes menores e transmite todos em *time-slots* sucessivos. Cada pacote possui um *bit* de final de mensagem (BFM), ao qual o terminal atribui o valor 0 (zero) em todos os pacotes, exceto no último, quando atribui 1 (um) para indicar o final da mensagem, liberando assim o canal. Terminais de dados com mensagens muito grandes podem ocupar o canal por um período muito longo, comprometendo a QoS dos outros terminais. No caso dos terminais de voz, eles também transmitem todos os pacotes presentes no seu *buffer* e depois liberam o canal; quando chegarem novos pacotes de uma mesma comunicação (Conversa), o terminal de voz tem que participar novamente do processo de contenção e aguardar a sua vez de transmitir em um instante futuro; entretanto, esse tempo é incerto,

podendo exceder o limite de atraso do pacote ou gerar uma variação de atraso muito alta.

O protocolo DQCA-RP implementa algumas modificações no DQCA, visando atender aos requisitos de QoS dos tráfegos periódicos e resolver os problemas de *jitter* dessas fontes. Na implementação apresentada neste trabalho, é considerado somente tráfego de voz (periódico) e de dados (aleatório), embora o protocolo possa ser adaptado para atender outros tipos de tráfego. O mecanismo se baseia na reserva periódica de *time-slots* para os usuários de voz ativos, de acordo com a taxa de geração dos seus pacotes, enquanto durar a comunicação e com uma única requisição (RTS). Assim, quando um terminal se torna ativo conseguindo o acesso ao canal, passa a ter *time-slots* reservados para ele com a freqüência semelhante à em que os pacotes são gerados; os *time-slots* que não estiverem reservados podem ser utilizados pelos outros terminais da mesma forma que no DQCA básico.

3.3. Estrutura e funcionamento

O protocolo proposto, DQCA-RP, mantém a estrutura do quadro do DQCA, o algoritmo de resolução de colisões e o esquema de acesso semelhante ao *Aloha* com carga de tráfego baixa. Como no esquema proposto em [5], para permitir a distinção do tipo do serviço, as RTS´s passam a transportar diferentes sinais, através de diferentes tipos de rajadas predefinidas, que permitem ao AP identificar o tipo de serviço do terminal. Assim, o AP pode distinguir entre usuários de voz e de dados sem a necessidade de *bits* de *Overhead* extras. A informação sobre o tipo de serviço do terminal que conseguiu o acesso é incluída no pacote de realimentação (FBP), para que todos os nós tenham conhecimento.

Da mesma forma que em outras variantes do DQCA para tráfego heterogêneo, como as apresentadas em [6] e [5], o esquema de reserva de acesso é modificado para permitir o tratamento diferenciado entre os pacotes de voz e de dados. Além das duas filas utilizadas no DQCA padrão, a DTQ para transmissão de dados e a CRQ para tratamento de colisões, no DQCA-RP é utilizada uma terceira fila distribuída, a VTQ (*Voice*

Transmission Queue) para controlar as reservas e transmissões dos terminais de voz ativos. Do mesmo modo que as outras duas filas do DQCA básico, a VTQ também é mantida nos terminais por dois valores inteiros, VQ que representa o número de terminais de voz ativos (tamanho da fila) e pVQ, que representa a posição do terminal na fila, sendo que VQ tem o mesmo valor em todos os terminais e pVQ valores independentes para cada terminal. O processo de manutenção desses inteiros, que representam os estados das filas, é realizado nos terminais com base nos parâmetros recebidos através do pacote de realimentação, FBP, conforme ilustrado no fluxograma da Figura 19. Assim como no DQCA e nas outras variantes deste protocolo, no DQCA-RP as regras do protocolo são executadas em cada terminal de forma independente, assíncrona e completamente distribuída.

No DQCA-RP um *time-slot* corresponde à transmissão de um quadro DQCA. A idéia básica é dedicar aos terminais de voz ativos, *time-slots* reservados para eles de acordo com a taxa de geração dos seus pacotes. A transmissão dos pacotes de dados é parecida com a do DQCA, podendo ser empregada uma disciplina de fila qualquer. A diferença básica é que no DQCA-RP o terminal de dados que ocupar a saída da DTQ adia a transmissão de um pacote sempre que o *time-slot* estiver reservado para um terminal de voz.

No PRMA, como a duração dos *time-slots* e do quadro são definidos de acordo com a taxa de geração dos pacotes de voz, *time-slots* pré-definidos específicos podem ser reservados para esses terminais. Entretanto, no DQCA a duração dos *time-slots* é variável, definida por fatores variáveis como os tamanhos dos pacotes e as taxas de transmissão dos terminais, e isso faz com que a reserva de *time-slots* pré-definidos (enumerados) para terminais de voz seja inviável. Sendo assim a reserva para pacotes de voz no DQCA-RP não é feita para um *time-slot* específico e sim de acordo com o instante previsto para a chegada do próximo pacote.

Como o tráfego de voz é uma fonte com taxa de *bit* constante, com pacotes sendo gerados em intervalos pré-definidos de τ *ms*, a partir do instante em que um terminal ganha o acesso para transmitir o seu primeiro pacote, todos os outros terminais podem determinar, com um atraso mínimo, os instantes, em intervalos de τ *ms*, que os pacotes

seguintes estarão prontos (disponíveis) para a transmissão. É baseado nesse princípio que o esquema de reserva do DQCA-RP é implementado.

Para cada terminal de voz ativo, que estiver aguardando na VTQ, é associado um temporizador (*timer*) que indica o instante esperado para a chegada do seu próximo pacote; a VTQ é ordenada por este *timer* e, no caso de dois terminais com o mesmo *timer*, a prioridade é dada ao terminal que estiver a mais tempo na fila, indicado pelo seu valor de pVQ. Os valores dos *timer's* dos terminais de voz ativos são mantidos em todos os terminais através de um vetor de comprimento igual ao tamanho da VTQ. Então, cada terminal mantém um vetor que contém os *timer's* de todos os terminais de voz ativos e as posições dos terminais neste vetor são as mesmas que eles ocupam na VTQ.

Sempre que um terminal de voz consegue o acesso, o seu *timer* é inicializado com o instante do início do quadro no qual ele transmitiu a RTS. Após a transmissão de cada pacote, o *timer* do terminal é incrementado pelo valor do intervalo entre chegadas de pacotes de voz, τ, ou recebe zero se for o último pacote da conversa. Os processos de manutenção das filas, assim como dos valores dos *timer's* dos terminais são mostrados no fluxograma da Figura 19, que ilustra os procedimentos executados por cada terminal, após o recebimento do pacote de realimentação, FBP, para atualizar as variáveis que mantém as filas nos terminais e os *timer's* dos terminais.

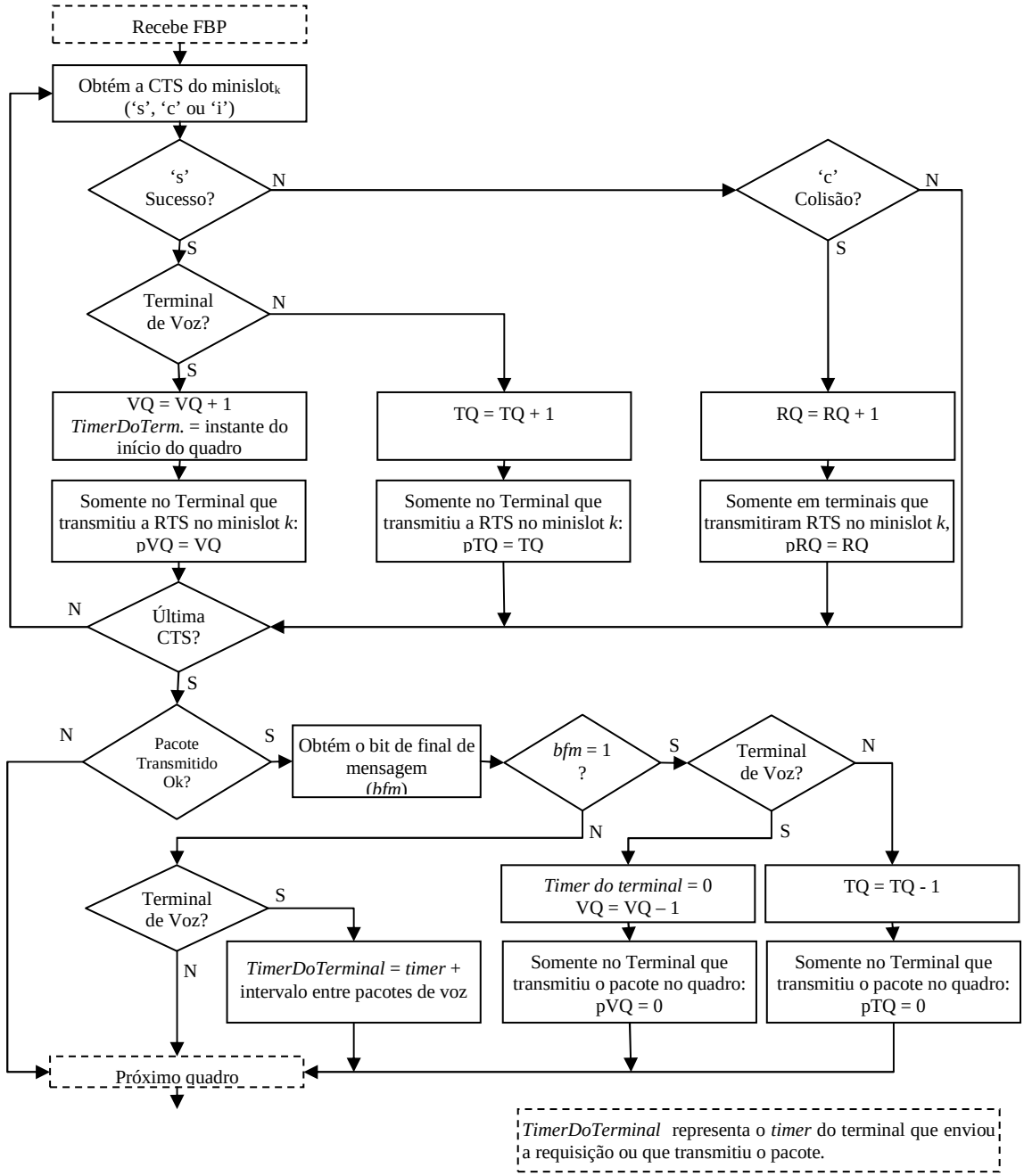

Figura 19: Fluxograma do processo de manutenção das filas distribuídas e dos *timer's* de terminais de voz

Os terminais atualizam as filas e os valores dos *timer's* de acordo com os parâmetros recebidos através do pacote de realimentação, referentes aos estados dos *minislots* de contenção e do pacote transmitido no quadro. Ao receber o pacote de realimentação, cada terminal executa o fluxograma da Figura 19, que pode ser dividido em duas partes: na primeira parte, as atualizações das filas são realizadas com base nos estados dos *minislots*, recebidos através das CTS's. Estas atualizações visam principalmente incluir

um terminal na fila de transmissão após uma requisição bem sucedida ou na fila de resolução de colisões, quando houver mais de uma requisição em um *minislot*. Na segunda parte do fluxograma é observado o estado do pacote transmitido no quadro, caso tenha sido transmitido algum, através do sub-campo *ACK* e do *bit* de final de mensagem, a fim de manter as filas de transmissão, os *timer's* e as reservas de *time-slots*.

A primeira parte do fluxograma é executada para cada CTS, referente a um respectivo *minislots* de contenção, que pode conter o valor 's', indicando sucesso no *minislot* (uma única requisição), 'c' indicando colisão no *minislot* ou 'i' para vazio, ou seja, nenhuma requisição. Ao concluir a última CTS passa para a segunda parte do fluxograma. A seguir descrevemos as ações para cada um dos possíveis status associados a cada minislot (s, c ou i).

's' – no caso de somente um terminal ter enviado requisição no *minislot* (sucesso), é verificado o tipo do terminal que a enviou. Se for um terminal de voz, a variável VQ, que indica o tamanho da fila de transmissão de voz, é incrementada, o *timer* do terminal que enviou a requisição recebe o instante do início do quadro e, no terminal que enviou a requisição, a variável pVQ recebe o valor de VQ, fazendo com que ele assuma a última posição na fila. Se for um terminal de dados, a variável TQ é incrementada e no terminal que enviou a requisição, a variável pTQ recebe o valor de TQ, assumindo assim a última posição na fila de transmissão de dados.

'c' – no caso de mais de um terminal ter enviado requisição no *minislot* (colisão), o valor de RQ, que indica o tamanho da fila de resolução de colisões, é incrementado e em cada terminal que transmitiu a requisição o valor de pRQ é incrementado, de forma que eles passam a ocupar a última posição na fila de resolução de colisões.

'i' – se não houve nenhuma requisição no *minislot*, então verifica a próxima CTS ou passa para a segunda parte do fluxograma, se for a última CTS.

Na segunda parte do fluxograma da Figura 19, se não houve transmissão de pacote com sucesso no quadro, o algoritmo é encerrado, caso contrário, verifica se é o último pacote

do terminal, indicado pelo *bit* de final de mensagem (bfm), a fim de liberar ou não a reserva. No caso de bfm=0, se o pacote transmitido foi de um terminal de dados o algoritmo é encerrado, senão o *timer* do terminal de voz que transmitiu o pacote é incrementado com o valor referente ao intervalo entre chegadas de pacotes de voz. Para bfm=1, se foi transmitido um pacote de voz, o *timer* do terminal que transmitiu recebe o valor zero, a variável VQ, que representa o tamanho da fila de transmissão de voz, é decrementada e no terminal que transmitiu o pacote, o valor de pVQ recebe zero; se o pacote transmitido foi de dados, o valor de TQ, que representa o tamanho da fila de transmissão de dados, é decrementado e no terminal que transmitiu o pacote, o valor de pTQ recebe zero. Desta forma, um terminal de voz libera a reserva sempre que passa para o estado OFF, comentado na seção 4.2.2.2.

Na Figura 20 apresenta-se esquematicamente uma seqüência de transmissões, no sentido *uplink*, de 6 quadros DQCA-RP, ilustrando quadro a quadro os estados das filas de transmissão, DTQ e VTQ, assim como as variações dos *timer's* dos terminais de voz ativos. Para facilitar o entendimento e o acompanhamento da seqüência, o primeiro quadro ilustrado inicia-se no instante 100 *ms* e são considerados os tempos de transmissão de quadros com pacotes de voz e dados de 3 *ms* e 9 *ms*, respectivamente, e o intervalo entre chegadas de pacotes de voz, τ, de 60 *ms*.

Como pode ser visto nos quadros enumerados como 1, 3, 4 e 6 na Figura 20, o processo de transmissão de dados segue como no DQCA padrão, com o terminal que ocupa a cabeça da DTQ transmitindo seus pacotes quadro a quadro de acordo com uma determinada disciplina de fila. A principal mudança implementada no DQCA-RP é que o terminal de dados que estiver na cabeça da DTQ adia a transmissão do seu próximo pacote sempre que, em um determinado *time-slot*, o terminal que ocupar a saída da VTQ tiver o valor do seu *timer* menor ou igual ao instante atual, como ocorre nos quadros 2 e 5 da figura com os terminais de voz Tv1 e Tv3, respectivamente. Isto indica que o *timer* do terminal expirou. Neste caso, tal *time-slot* é dito "reservado" para o terminal de voz, que transmitirá o seu pacote. Se for o último pacote do terminal de voz, ele indicará no *bit* de final de mensagem do quadro e sairá da VTQ, caso contrário a reserva é renovada para o terminal, que passa a ocupar o final da VTQ. A renovação da reserva é feita incrementando-se o *timer* do terminal em τ *ms*, como nos finais dos quadros 2 e 5 da

Figura 20, onde pode-se observar que os terminais de voz que transmitiram, Tv1 e Tv3, tiveram os seus *timer's* incrementados em 60 *ms*.

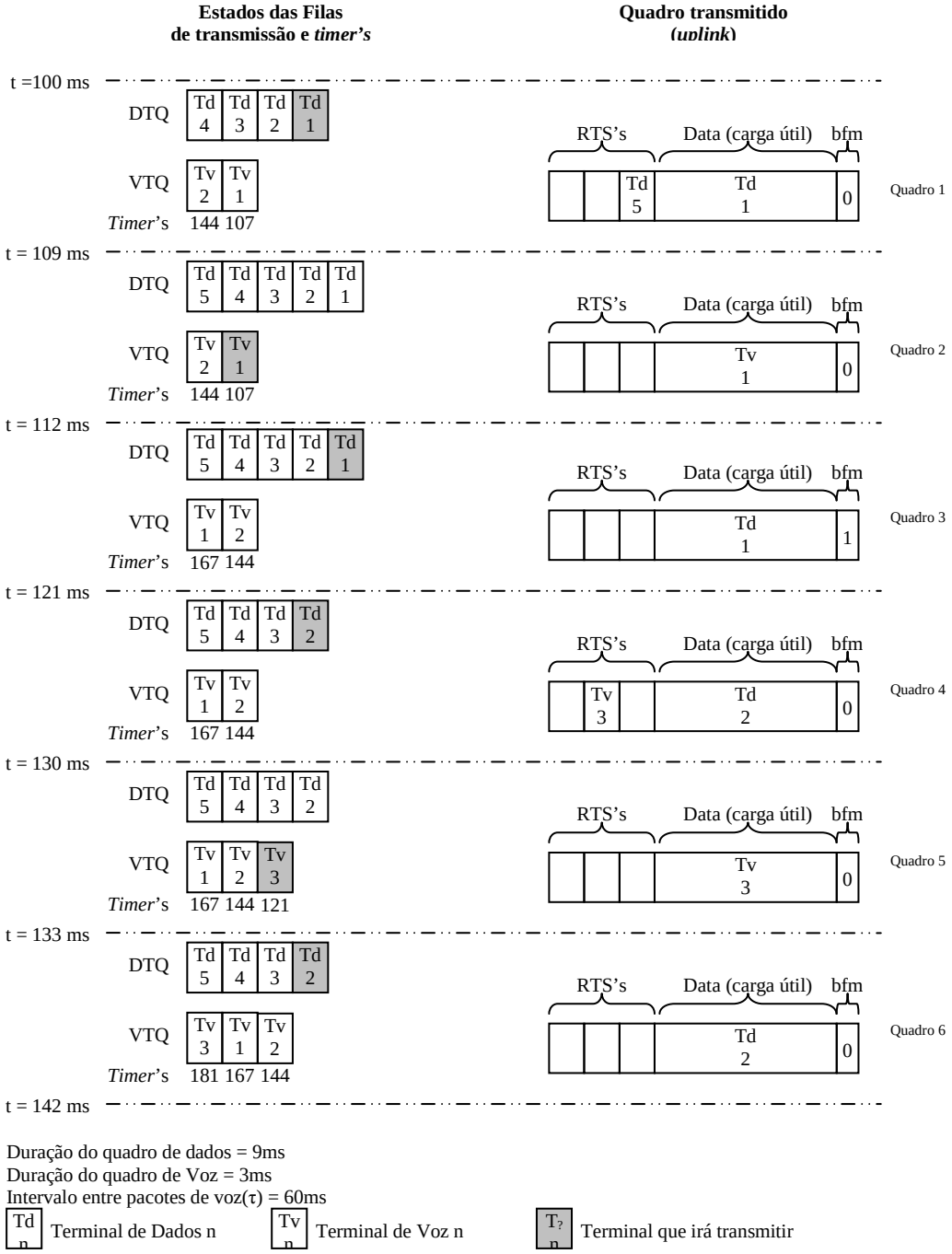

Figura 20: Seqüência de 6 quadros DQCA-RP

No início da seqüência de quadros ilustrada na Figura 20 (no instante 100ms), as filas de transmissão de dados (DTQ) e de voz (VTQ) possuem 4 e 2 terminais

respectivamente. A seguir apresenta-se uma descrição detalhada quadro a quadro da seqüência ilustrada.

(100ms) – Como o *timer* do terminal de voz que ocupa a cabeça da VTQ não expirou (107 > 100), o terminal de dados que ocupa a cabeça da DTQ, Td1, transmite o seu pacote, definindo o *bit* de final de mensagem com o valor zero. Além disso, o terminal de dados Td5 com uma nova mensagem envia uma requisição no terceiro *minislot* de contenção. Como não houve colisão na contenção, Td5 passa a ocupar a última posição da DTQ no próximo quadro.

(109ms) – Agora o *timer* do terminal de voz Tv1 é menor que o instante atual (107 < 109), indicando que expirou; então o terminal de dados Td1 adia a transmissão do seu pacote para que Tv1 possa transmitir o seu e como não é o último pacote (bfm=0), tem o seu *timer* incrementado em 60ms, que é o tempo entre chegadas de pacotes de voz considerado neste exemplo.

(112ms) – Como o *timer* do terminal de voz, Tv2, que ocupa a cabeça da VTQ não expirou (144 > 112), Td1 transmite mais um pacote e indica que é o último pacote da mensagem, definindo o *bit* de final de mensagem com o valor um. Desta forma ele não aparecerá mais na DTQ no início do quadro seguinte.

(121ms) – Como o *timer* do terminal de voz, Tv2, que ocupa a cabeça da VTQ ainda não expirou (144 > 121), Td2 que agora ocupa a cabeça da DTQ transmite um pacote. Além disso, o terminal de voz Tv3 que se tornou ativo envia uma requisição de acesso no segundo *minislot*. Como não houve colisão na contenção, Tv3 entra na VTQ e o seu *timer* é inicializado com 121, que é o instante do início do quadro atual.

(130ms) – O terminal de voz Tv3 que acabou de entrar na fila tem o seu *timer* estourado (121 < 130); então o Td2 que ocupa a cabeça da DTQ adia a sua transmissão para que Tv3 transmita o seu pacote. Após a transmissão o *timer* de Tv3 é incrementado em 60ms (121+60 = 181).

(133ms) – Como não tem terminal de voz com *timer* estourado, Td2 transmite seu pacote.

No quadro 4 da Figura 20, observa-se que o *timer* do terminal que conseguiu o acesso, Tv3, foi inicializado com o valor 121, igual ao instante do início do quadro no qual foi enviada a requisição de acesso. Isso pode fazer com que o terminal que acaba de obter o acesso tenha permissão para transmitir antes de outro que já espera a mais tempo na fila, mas em contra partida garante que o *timer* do terminal tenha uma defasagem mínima, uma vez que um atraso no primeiro pacote acarretaria em atrasos nos pacotes subseqüentes.

No DQCA-RP a fila de transmissão de dados, DTQ, pode empregar qualquer disciplina de fila, entretanto nos testes realizados nesta dissertação os terminais de dados serão ordenados pela *Rb*, como nos esquemas CL1 em [5] e DQCA+CL em [6].

Capítulo 4 - Simulações e resultados obtidos

Neste capítulo são apresentados os resultados obtidos através de simulações por computador, realizadas com o objetivo de analisar os ganhos alcançáveis com o DQCA-RP, em relação ao DQCA e outras variantes desse protocolo, descritos na Seção 4.1. Para conforto do leitor, a Seção 4.1 apresenta um breve resumo das características dos protocolos analisados. Na Seção 4.2 são definidos o ambiente considerado nas simulações, os modelos empregados na geração dos tráfegos de dados e voz e os cenários considerados. Finalmente, na Seção 4.3 os protocolos são comparados, com relação às algumas métricas de desempenho consideradas.

4.1. Os Protocolos analisados

Nesta seção são apresentados os esquemas baseados no DQCA que foram utilizados como parâmetros de comparação na avaliação do desempenho do protocolo proposto neste trabalho, DQCA-RP. Uma descrição detalhada da estrutura do simulador utilizado para avaliar o desempenho dos protocolos é apresentada no Anexo A desta dissertação. A Tabela 1 apresenta um resumo dos esquemas analisados e nela observa-se que os esquemas DQCA, DQCA-CL1 e DQCA-CL2 não fazem distinção dos tipos de serviço dos terminais e também que os esquemas DQCA e DQCA-DIF1 são os únicos que não consideram as taxas dos terminais nas decisões de escalonamento.

Tabela 1: Resumo dos esquemas DQCA estudados

Esquema DQCA	Descrição	Diferencia Voz e Dados?	Considera Taxa Disponível?
DQCA	DQCA padrão, sem priorização.	N	N
DQCA-CL1	Prioriza terminais com melhor canal. Emprega uma fila ordenada pelas taxas disponíveis dos terminais. Melhora a vazão aumentando a utilização do canal, mas apresenta problemas relacionados a justiça.	N	S
DQCA-CL2	Uma fila ordenada pela razão da taxa pela posição na fila (*Rb*/*pTQ*). Mais justiça que o esquema DQCA-CL1.	N	S
DQCA-DIF1	Duas filas FIFO para dados e voz. Terminais de dados só transmitem se a fila de voz, mais prioritária, estiver vazia. Melhora a QoS dos terminais de voz.	S	N
DQCA-DIF2	Duas filas com prioridade para Voz. A fila de dados ordenada pela taxa e a de voz com disciplina FIFO. Aumenta a vazão, com melhora da QoS de voz.	S	S
DQCA-RP	Terminais de Voz usam *time-slots* reservados. Os outros *time-slots* são usados pelos terminais de Dados, seguindo regra do DQCA-CL1. Mantém alta vazão, reduzindo atraso e *jitter* de voz.	S	S

4.1.1. DQCA

Conforme comentado no Capítulo 3, o DQCA é um protocolo de alto desempenho e estável, que mantém uma alta vazão sob qualquer carga de tráfego. Ele se comporta como um protocolo de acesso aleatório quando a carga de tráfego é baixa, passando automaticamente para um esquema de reserva, à medida que o tráfego aumenta. O DQCA básico utiliza uma fila de transmissão que emprega a disciplina FIFO, não fazendo distinção das classes de serviço dos terminais. A estrutura e funcionamento do DQCA já foram apresentados em detalhes no Capítulo 2 da dissertação.

4.1.2. DQCA-CL1

No esquema DQCA-CL1, apresentado como CL1 na Seção 2.5.2.1 e em [31], os terminais que ocupam a fila de transmissão são ordenados pela suas taxas de transmissão, de forma que os terminais com melhores condições do enlace têm prioridade na decisão de escalonamento. No caso de terminais com mesma taxa, a prioridade é dada ao terminal que estiver a mais tempo na fila. Este esquema visa principalmente o aumento na vazão e redução no atraso médio de pacotes, apesar de apresentar problemas de justiça entre os terminais.

4.1.3. DQCA-CL2

Neste esquema, apresentado na Seção 2.5.2.1 e em [31] como CL2, a fila de transmissão é ordenada por uma função de prioridade virtual, definida pela razão da taxa de transmissão do terminal pela sua posição na fila. A idéia é melhorar a vazão enquanto provê certo nível de justiça. Para isso, é incentivada a transmissão de usuários com taxa de *bits* disponível mais alta, sem desconsiderar os seus tempos de espera na fila.

4.1.4. DQCA-DIF1

A proposta do esquema DQCA-DIF1, ou DQCA+ em [6], é reduzir o atraso dos pacotes de voz dando prioridade a terminais deste tipo na decisão de escalonamento. Para isso são utilizadas duas filas FIFO, uma para terminais de dados e outra para terminais de voz, sendo esta mais prioritária. Neste esquema, terminais de dados podem obter o acesso somente quando a fila de transmissão de voz estiver vazia. Desta forma, busca-se satisfazer os requisitos de QoS dos terminais de voz.

4.1.5. DQCA-DIF2

O esquema DQCA-DIF2, proposto em [6] como DQCA+CL, também utiliza duas filas para o escalonamento das transmissões dos terminais de voz e de dados, com prioridade para os de voz. Entretanto, a diferença deste esquema com relação ao DQCA-DIF1 é que a fila de terminais de dados é ordenada pelas taxas dos terminais. A utilização desta técnica *cross-layer* permite um aumento da vazão e a redução do atraso médio dos pacotes. Um esquema parecido proposto em [5] e apresentado na Seção 2.5.2.2 como CL1, ordena as duas filas pelas taxas dos terminais.

4.1.6. DQCA-RP

O DQCA-RP, apresentado no Capítulo 3, procura reduzir o atraso, a variação do atraso e conseqüentemente as perdas de pacotes de voz, através da reserva de *time-slots* para os terminais de voz ativos, na mesma freqüência em que seus pacotes são gerados. Neste esquema, à medida que os pacotes de voz são gerados nos terminais eles encontram *time-slots* reservados para transmissão com um tempo de espera (atraso) mínimo. Os *time-slots* não reservados são utilizados pelos terminais de dados da mesma forma que no DQCA padrão, utilizando uma disciplina de fila qualquer. Na implementação empregada nesta dissertação, os terminais de dados são ordenados pelas suas taxas de transmissão disponíveis, relacionadas às suas qualidades dos enlaces.

4.2. O ambiente de simulação

Nesta seção são apresentados os modelos para o canal sem fio e para a geração de tráfego empregados na ferramenta de simulação; é feita uma contextualização geral do ambiente e dos parâmetros de simulação, permitindo um melhor entendimento dos resultados apresentados em seguida.

4.2.1. O Modelo do canal

Para simular as variações nos estados dos enlaces dos terminais foi considerado um modelo de canal como o apresentado em [34], que propõem um canal de Markov com quatro estados, cada um representando uma das quatro taxas de transmissão disponíveis, de 1 *Mbps*, 2 *Mbps*, 5,5 *Mbps* e 11 *Mbps*. De acordo com este modelo, as condições atuais dos canais móveis apresentam certa correlação com as condições que apresentavam em um instante imediatamente anterior. Isto significa que, por exemplo, um terminal que está transmitindo a uma taxa de 1 *Mbps* provavelmente não será capaz de transmitir a 11 *Mbps* e vice-versa, depois de um tempo menor ou igual ao tempo de coerência. O tempo de coerência é uma medida estatística do intervalo de tempo durante o qual a taxa de transmissão disponível de um terminal não muda significativamente, podendo ser considerada como invariante. Dessa forma o tempo de coerência determina a freqüência com que as taxas de transmissão disponíveis dos terminais precisam ser atualizadas.

A cadeia de Markov é ilustrada na Figura 21 e é representada pela matriz de probabilidades de transição, *P*, mostrada na Figura 22, cujos valores determinam as probabilidades de transição entre as taxas de transmissão disponíveis. Os valores das probabilidades foram definidos de acordo com [34] e [5] e segundo os autores estes valores representam um cenário onde as taxas de transmissão mais prováveis são de 2 e 5.5 *Mbps*.

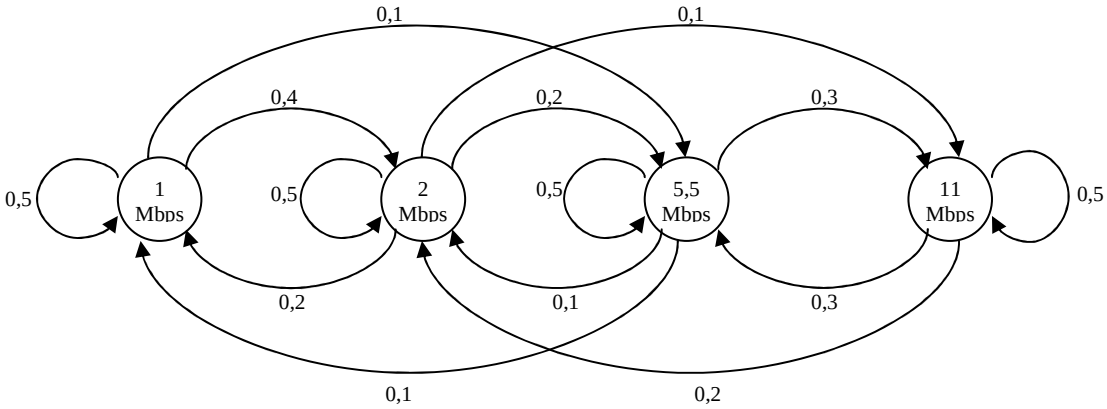

Figura 21: Modelo Markoviano para os estados do canal

Durante a simulação, após cada intervalo de tempo igual ao tempo de coerência, as taxas de transmissão dos terminais são redefinidas de acordo com a matriz de transição, *P*, apresentada na Figura 22, onde as linhas na matriz representam o estado atual e as colunas os estados futuros.

$$P = \overbrace{\begin{matrix} 1 & 2 & 5.5 & 11 \end{matrix}}^{\text{Estado futuro}} \begin{matrix} \\ \left[\begin{matrix} 0.5 & 0.4 & 0.1 & 0 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.5 & 0.3 \\ 0 & 0.2 & 0.3 & 0.5 \end{matrix}\right] \end{matrix} \begin{matrix} 1 \\ 2 \\ 5.5 \\ 11 \end{matrix} \} \text{Estado atual}$$

Figura 22: Matriz de probabilidades de transição da cadeia de Markov de estados do canal

Os valores da matriz determinam as probabilidades de se passar de um estado para outro. Entretanto, para determinar as taxas de transmissão iniciais dos terminais, foram calculadas as probabilidades em regime permanente da matriz de transição. Considerando a matriz *P* e a expressão a seguir,

$$\pi = \pi P \Rightarrow \begin{bmatrix} \pi_0 & \pi_1 & \pi_2 & \pi_3 \end{bmatrix} = \begin{bmatrix} \pi_0 & \pi_1 & \pi_2 & \pi_3 \end{bmatrix} \times \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix}, \quad (3)$$

onde π é um vetor com as probabilidades em regime permanente, foram obtidas as equações:

$$\begin{aligned} \pi_0 &= 0{,}5\pi_0 + 0{,}2\pi_1 + 0{,}1\pi_2 \\ \pi_1 &= 0{,}4\pi_0 + 0{,}5\pi_1 + 0{,}1\pi_2 + 0{,}2\pi_3 \\ \pi_2 &= 0{,}1\pi_0 + 0{,}2\pi_1 + 0{,}5\pi_2 + 0{,}3\pi_3 \\ \pi_3 &= 0{,}1\pi_1 + 0{,}3\pi_2 + 0{,}5\pi_3 \end{aligned} \quad (4)$$

e aplicando em $\sum_i \pi_i = 1$, obtém-se a matriz π, que é a FMP (Função massa de probabilidade) estacionária dos estados da cadeia de Markov, ou simplesmente FMP em regime permanente.

$$\pi = [\pi_0 \quad \pi_1 \quad \pi_2 \quad \pi_3] = [0{,}1765 \quad 0{,}2941 \quad 0{,}2941 \quad 0{,}2353] \tag{5}$$

Além da matriz de transição e das probabilidades em regime permanente, os outros parâmetros para configuração do canal são apresentados na Tabela 2 e foram fixados com os mesmos valores utilizados na referência [32].

Tabela 2: Parâmetros para configuração do canal sem fio

Tp – Atraso de propagação	1µs
Tempo de coerência	30*ms*
Taxas disponíveis	1, 2, 5.5, 11 *Mbps*

4.2.2. Modelos para geração de tráfego

Os terminais no sistema são modelados como dispositivos que geram as mensagens de informação de acordo com um determinado modelo de geração de tráfego. Nesta seção são apresentados os modelos empregados na geração de tráfego dos terminais de dados e de voz.

4.2.2.1. Geração de tráfego dos terminais de dados

O modelo de geração de tráfego de dados segue uma distribuição de Poisson e cada mensagem gerada nos terminais tem o tamanho definido por uma variável aleatória exponencial com média de 10 x *Ld*, onde *Ld* é o tamanho de cada pacote de dados. Os parâmetros relacionados ao tráfego de dados foram fixados com os mesmos valores das referências [5] e [6] e são apresentados na Tabela 3.

Tabela 3: Parâmetros para geração de tráfego de dados

Modelo de geração de tráfego	Poisson
Ld – tamanho de um pacote de dados	1000 *bytes*
L_m – Tamanho médio das mensagens	10 x Ld *bytes*
Distribuição dos tamanhos das mensagens	Exponencial

A geração das mensagens dos terminais de dados foi implementada de modo que o valor da carga de dados submetida total alcance a carga desejada em cada cenário de simulação. Para garantir a carga de dados desejada, o intervalo médio entre as chegadas de mensagens de cada terminal de dados, D_m, é calculado pela Eq.(6), levando-se em conta o tamanho médio das mensagens, L_m, em *bits*, o número de terminais de dados, *n*, e a carga de dados desejada, O_d, em *bits* por segundo.

$$D_m = (L_m / O_d) * n \quad (6)$$

Conforme comentado, o tamanho de cada mensagem gerada nos terminais é definido por uma variável aleatória exponencial com média L_m. A cada mensagem gerada em um terminal, é agendado um novo evento para a chegada da mensagem seguinte e o instante deste evento é definido por uma variável aleatória exponencial com média D_m, mais o instante atual.

4.2.2.2. Geração de tráfego dos terminais de voz

O modelo utilizado para a geração de tráfego dos terminais de voz é o modelo *ON-OFF*, representado através de uma cadeia de Markov de dois estados, *ON* e *OFF*, ilustrado no diagrama de estados da Figura 23. De acordo com este modelo, apresentado em [5], o terminal de voz pode estar no estado *OFF* (silêncio) ou no estado *ON* (ativo). Os terminais partem do estado *OFF*, no qual não geram pacotes, e a cada intervalo de tempo têm uma probabilidade α de passar para o estado *ON*, no qual são gerados os pacotes de informação ou rajadas vocais (*talkspurts*); uma vez no estado *ON* têm a probabilidade β de voltar ao estado *OFF* e assim sucessivamente. O tempo médio que um terminal permanece em cada estado, *ON* ou *OFF*, é uma variável aleatória com distribuição exponencial de médias $1/\alpha$ e $1/\beta$ respectivamente [45].

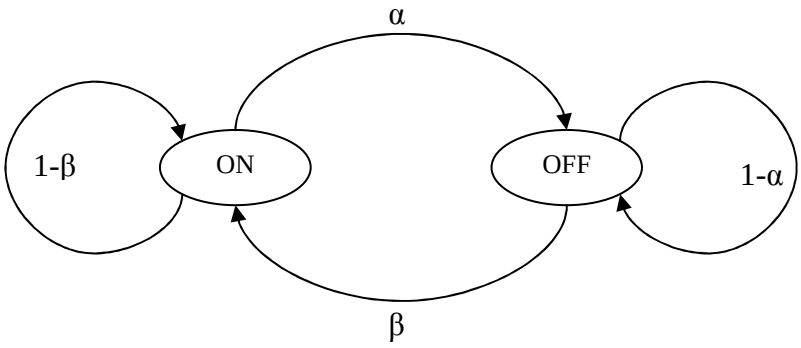

Figura 23: Modelo de tráfego de voz ON-OFF

Na Tabela 4 são apresentados os parâmetros para geração de tráfego de voz. Os valores das durações dos períodos ON e OFF foram definidos para este trabalho de acordo com [46] e os outros parâmetros de acordo com [5] e [6].

Tabela 4: Parâmetros para geração de tráfego de voz

Modelo de geração de tráfego	ON-OFF
Duração média dos períodos *ON*	1,004 seg.
Duração média dos períodos *OFF*	1,587 seg.
Taxa de geração, quando no estado *ON*	13 kbps
Lv – Tamanho de um pacote de voz	100 *bytes*
Atraso máximo permitido	300 *ms*
Porcentagem máxima de perdas permitida	1%

No protocolo DQCA-RP, quando um terminal de voz se torna OFF ele transmite o seu último pacote atribuindo o valor 1 (um) ao *bit* de final de mensagem liberando a reserva. Diferentemente, no DQCA e as outras variantes o canal é liberado sempre que não tiver mais pacotes para serem transmitidos.

4.2.3. Configuração do quadro

Como comentado anteriormente no Capítulo 2, o quadro DQCA é formado pelo campo de contenção, o campo *Data* e o FBP. O campo de contenção é composto de *m minislots*, onde são enviadas as requisições de acessos (RTS). O campo *Data* é a parte do quadro dedicada à transmissão do pacote, contendo a carga útil do quadro, e junto

com ele é enviado um *bit* de final de mensagem (*bfm*). O FBP, que é transmitido pelo AP no sentido *downlink*, contém os subcampos *ACK*, para confirmação do recebimento do pacote, as CTS's correspondentes a cada RTS enviada no campo contenção, o subcampo FC (*Frame control* – controle de quadro) e o subcampo FCS (*Frame Control Sequence* – Controle de seqüência de quadro). Junto com o FBP é enviado ainda um *bit* de final de mensagem e nos esquemas que consideram os estados dos enlaces nas decisões de escalonamento é incluído o subcampo *OverheadCrossLayer*, cujo tamanho é 2 x *TQ bits*, arredondado para o próximo número inteiro de *bytes* [5], [31], onde *TQ* representa o número total de terminais ativos. Os parâmetros considerados para a configuração do quadro são apresentados na Tabela 5 e foram fixados de acordo com as referências [5], [6], [34] e [7].

Tabela 5: Parâmetros de configuração do Quadro

Número de *minislots*	3
Ld – Tamanho dos pacotes de Dados	1000 *bytes*
Lv – Tamanho dos pacotes de Voz	100 *bytes*
Tp – Atraso de propagação	1 μs
Duração de um *minislot*	2 μs
Intervalo SIFS	10 μs
Cabeçalho da camada Física	96 μs
Cabeçalho da camada MAC	34 *bytes*
CTS	6 *bytes*
FC	2 *bytes*
ACK	1 *byte*
FCS	4 *bytes*
OverheadCrossLayer	2*bits* x *TQ* (arredondado para múltiplo de 8)
Taxa de transmissão das informações de sinalização e controle	1 *Mbps* (taxa mínima)

4.2.4. Parâmetros das simulações

Para análise e comparação dos desempenhos dos esquemas estudados, foram realizadas baterias de simulações, variando a carga total de dados submetida. Cada combinação dos parâmetros a serem considerados em uma simulação, que compreendem carga submetida, protocolo executado pelos terminais, número de terminais de voz, etc., constitui um cenário de simulação e para cada cenário foi realizada uma simulação com

duração de 1000 segundos. Em todos os cenários, foram considerados 20 terminais de dados gerando pacotes de 1000 *bytes*, conforme descrito na seção 4.2.2.1, e um número variável de terminais de voz gerando pacotes de 100 *bytes*. Assumiu-se também que independente das condições do canal, todos os pacotes são recebidos sem erros de transmissão com as taxas selecionadas e também que não há perdas devido a estouros nos limites dos buffers dos terminais; perdas ocorrem somente em pacotes de voz que são descartados ao excederem o limite de atraso permitido. Os parâmetros empregados nas simulações estão resumidos na Tabela 6.

Tabela 6: Parâmetros empregados nas simulações dos esquemas DQCA

Tempo de simulação	1000 segundos ou até violar a QoS dos terminais de voz
Número de terminais de dados, *Nd*	20
Número de terminais de Voz, *Nv*	Variando em 10, 20 e 30 terminais de voz
Carga de dados submetida	Variando de 0,25 *Mbps* até 5 *Mbps*, em saltos de 0,25 *Mbps*.

Exceto nas simulações para obtenção dos resultados da Figura 24, todas as outras simulações são encerradas se o requisito de perdas de pacote de voz (1%) for violado, entretanto, esta verificação inicia-se somente depois de decorrido 20% do tempo total da simulação. Da mesma forma que em [6], a simulação não é encerrada com a violação do requisito de QoS dos terminais de dados, indo até o final, ocorrendo ou não a violação.

4.3. Resultados obtidos

Dentre as métricas de desempenho mais consideradas na literatura, destacam-se a vazão, o atraso, o *jitter* e a taxa de perdas de pacotes. No entanto, em se tratando de classes de tráfego distintas, é necessário considerar a natureza e os requisitos de cada classe para se definir os parâmetros de comparação adequados a uma delas. Por exemplo, enquanto o tráfego de dados é sensível à taxa de perdas, tráfego multimídia como o de voz é mais sensível ao atraso e *jitter* do que propriamente à perdas de pacotes [47]. De acordo com [48], neste tipo de tráfego, perdas de pacotes dentro de certos limites chegam a ser imperceptíveis para quem está recebendo as informações, ao passo que níveis de atraso e *jitter* muito altos podem comprometer ou até inviabilizar a comunicação.

Nas próximas seções são apresentados os resultados da avaliação do desempenho do esquema proposto, comparando-o com os dos outros esquemas baseados no DQCA descritos na Seção 4.1.

4.3.1. Vazão de dados

A vazão, ou *throughput*, é um importante parâmetro a ser considerado quando se está comparando o desempenho de diferentes protocolos ou algoritmos. Ela define a taxa de transferência dos dados e no contexto deste trabalho representa o número de *bits* de dados que é transmitido por segundo, dado pela Eq.(7):

$$S = \frac{P}{T} \tag{7}$$

onde P é o número de *bits* de informação enviados e T o tempo total da simulação.

Nos protocolos baseados no DQCA, a vazão cresce linearmente com o aumento da carga submetida e se estabiliza ao atingir o valor máximo alcançável, dentro do cenário definido. A partir deste ponto a vazão se mantém constante, apesar do aumento da carga de dados. Este comportamento é ilustrado no gráfico da Figura 24 e demonstra a estabilidade destes protocolos para altas cargas de tráfego, principalmente devido aos processos de tratamento de colisão e de transmissão ocorrerem de forma independente. Na Figura 24 são apresentados os resultados de vazão de dados em função da carga de dados submetida de dados, obtidos nas simulações para os esquemas DQCA testados, considerando 10 terminais de voz, gerando tráfego conforme descrito na seção 4.2.2.2.

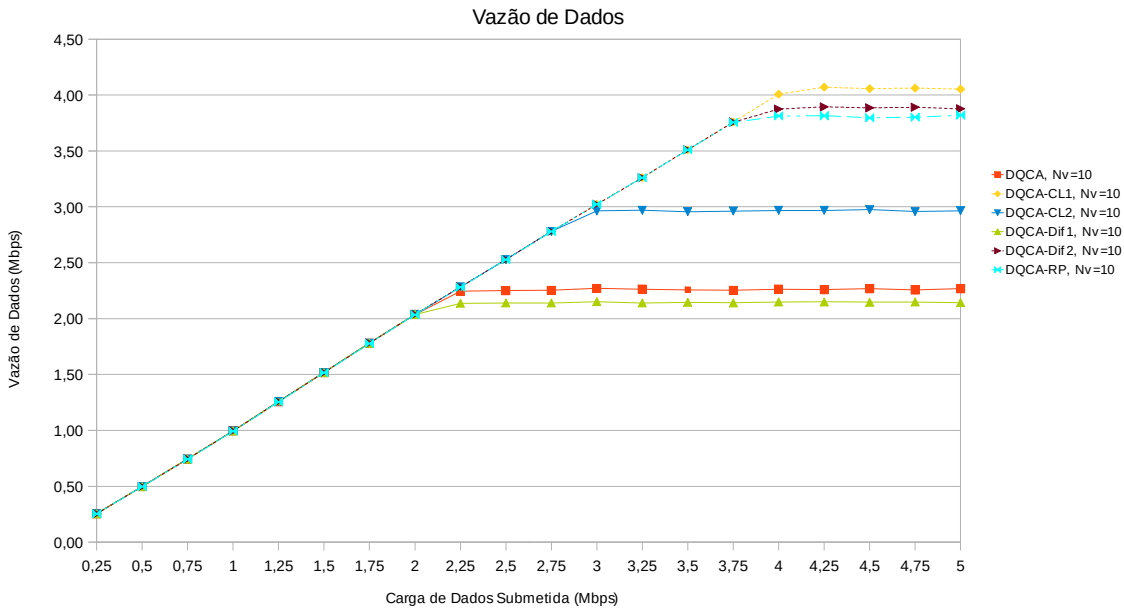

Figura 24: Vazão de dados com 10 terminais de voz, em função da carga de dados submetida – sem considerar violação de QoS de voz.

Observa-se pela Figura que, de maneira geral, os valores de vazão alcançam níveis significativamente mais altos nos esquemas que empregam o conceito de escalonamento oportunista, DQCA-CL1, DQCA-Dif2 e DQCA-RP, os quais priorizam de alguma forma a transmissão de terminais com melhores condições nos enlaces. Esta técnica possibilita o aumento no número de pacotes transmitidos a taxas maiores, o que implica em menores durações dos quadros e conseqüentemente uma utilização mais eficiente do canal. Observa-se ainda que o esquema DQCA-CL2 alcança uma vazão inferior aos três primeiros, no entanto, ele apresenta um resultado melhor em justiça entre os terminais, visto que os tempos de espera dos terminais na fila são considerados nas decisões de escalonamento. Com o objetivo de oferecer uma visão mais completa dos comportamentos dos esquemas estudados, o gráfico da Fi**gura 24** foi traçado para todos os valores de carga submetida, independente da violação da QoS dos terminais de voz, ou seja, a simulação prosseguiu para todos os níveis de carga submetida, mesmo quando as perdas de pacotes de voz ultrapassaram 1% dos pacotes gerados, caracterizando uma violação da QoS. Assim, devemos ressaltar que, se considerarmos requisitos de QoS, a vazão máxima obtida em alguns esquemas é inferior àquela que aparece na Fi**gura 24**, como veremos a seguir.

A Figura 25 dá uma visão mais realista dos resultados de vazão dos esquemas testados quando os requisitos de QoS são considerados. Observa-se que, a partir do momento que a QoS dos terminais de voz não é satisfeita a simulação não prossegue com cargas de dados mais altas. Na figura são apresentados os resultados de vazão em função das cargas submetidas de dados, com 10 e 20 terminais de voz.

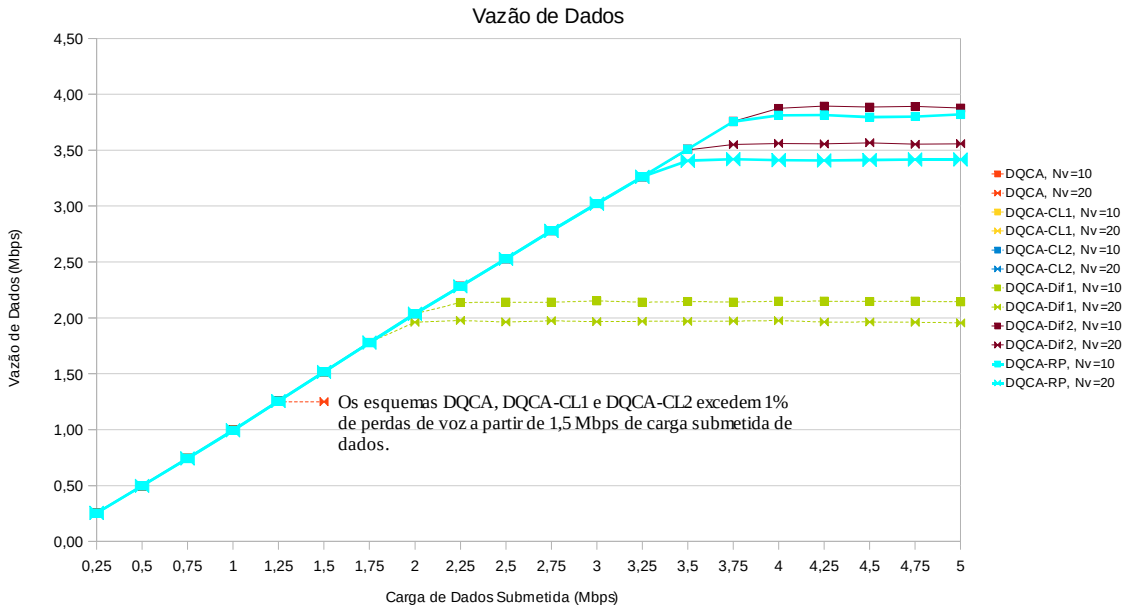

Figura 25: Vazão de dados em função da carga de dados submetida com 10 e 20 terminais de voz.

Observa-se pela figura que nos três esquemas que não diferenciam terminais de voz e de dados na decisão de escalonamento, DQCA, DQCA-CL1 e DQCA-CL2, com cargas de dados a partir de 1,5 *Mbps* os pacotes de voz sofrem perdas acima de 1% do total gerado, violando assim o requisito de QoS desta classe de tráfego. Como o esquema DQCA-DIF1 prioriza os terminais de voz sobre os de dados, ele não viola o requisito de QoS desses terminais, entretanto, como ele não emprega o conceito de escalonamento oportunista, a vazão de dados não ultrapassa 2,15 e 1,97 *Mbps*, com 10 e 20 terminais de voz, respectivamente. Nota-se ainda, que o esquema DQCA-DIF2 apresenta o melhor desempenho em vazão de dados. O DQCA-RP apresenta uma vazão 1,8% inferior ao DQCA-DIF2, o que representa um custo baixo frente aos ganhos obtidos em atraso médio, *jitter* e perdas dos pacotes de voz, comentados nas próximas seções.

Observa-se ainda pela Figura 25, que aumentando de 10 para 20 o número de terminais de voz a vazão máxima alcançável com o DQCA-Dif2 cai de 3,89 *Mbps* para 3,57 *Mbps*, enquanto com o DQCA-RP cai de 3,82 *Mbps* para 3,42 *Mbps*.

4.3.2. Atraso médio dos pacotes de voz

O atraso de um pacote é o tempo decorrido desde a sua geração no terminal até o instante em que ele foi recebido corretamente no terminal de destino. Assim, o atraso médio de pacotes de voz é calculado pela média dos atrasos sofridos por todos os pacotes transmitidos pelos terminais de voz e pode ser dado pela Eq.(8).

$$\bar{d} = \frac{\sum_{i=1}^{N} d_i}{N} \tag{8}$$

Onde $\bar{d}$ é o atraso médio de pacotes de voz, d_i é o atraso sofrido pelo pacote i e N o número total de pacotes de voz transmitidos.

Conforme comentado na introdução e em outras seções desta dissertação, o atraso e *jitter* sobre pacotes de voz são parâmetros muito importantes a serem considerados em um protocolo de acesso multi-serviço. Altos índices de atraso e de *jitter* modificam o padrão da fala, prejudicando a qualidade da comunicação. Além disso, pacotes de voz que sofrem atrasos superiores a 300 *ms* são descartados. As perdas até o limite de 1% são toleráveis e contornadas no processo de decodificação, entretanto, acima deste limite constituem violação da QoS do tráfego de voz [5], [6].

Os resultados de atraso médio de pacotes de voz em função das cargas submetidas, considerando 10 terminais de voz compartilhando o canal, são mostrados na Figura 26. Com os esquemas DQCA, DQCA-CL1 e DQCA-CL2 o índice de perdas de pacotes de voz excede o limite de 1% com cargas submetidas a partir de 1,5 *Mbps*. Observa-se pela figura que o esquema proposto, DQCA-RP, provê o menor índice de atraso médio entre

os protocolos estudados, com redução de 88,3% e 82,3% sobre o DQCA-DIF1 e o DQCA-DIF2, respectivamente, para altas cargas de tráfego de dados.

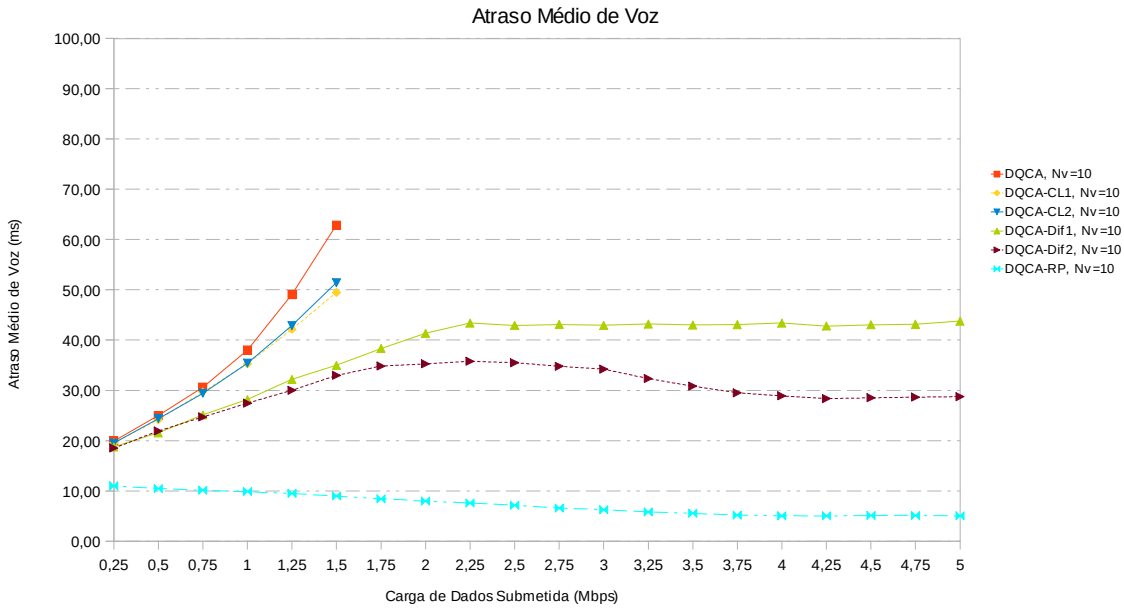

Figura 26: Atraso médio de voz em função da carga de dados submetida

Ainda de acordo com a Figura 26, observa-se que com os esquemas DQCA-DIF2 e DQCA-RP, o atraso médio cai com o aumento da carga de dados. Este comportamento se deve ao dimensionamento do quadro quando as filas de transmissão estão vazias. Conforme comentado na Seção 2.5, quando as filas de transmissão e de tratamento de colisões estiverem vazias, o terminal que enviar uma requisição pode transmitir imediatamente o primeiro pacote no mesmo quadro. Assim, para permitir que terminais de dados e de voz compartilhem o canal, quando as filas estão vazias, o quadro é dimensionado de acordo com o maior tamanho de pacote (1000 *bytes* de dados) sendo transmitido à taxa mínima de 1 *Mbps*, de forma que nesta situação um terminal de voz transmite seu pacote de 100 *bytes* em um *time-slot* dimensionado para transmitir um pacote de 1000 *bytes* e isto ocorre com mais freqüência quando o tráfego é baixo. Diante disso, à medida que o tráfego aumenta, são maiores as probabilidades de os pacotes serem transmitidos sob taxas mais altas e de os terminais de voz transmitirem seus pacotes em *time-slots* com durações menores, explicando a redução do atraso de voz com o aumento da carga de dados.

4.3.3. Variação dos atrasos de pacotes de voz

Uma medida para estimar a variação dos atrasos dos pacotes de voz é o desvio padrão do atraso, que representa a dispersão com relação ao valor médio e é calculado pela Eq.(9).

$$Dp = \sqrt{\frac{\sum_{i}^{N} \left(D_i - \overline{D}\right)^2}{N}} \tag{9}$$

onde *Dp* é o desvio padrão dos atrasos de *N* pacotes, D_i é o atraso do pacote *i* e $\overline{D}$ é a média dos atrasos.

Quanto maior o desvio padrão, mais disperso em relação à média está o atraso e, conseqüentemente, maior será o *jitter* na rede. Na seção anterior comentamos sobre a importância de um *jitter* reduzido para a qualidade da comunicação de voz. Logo, um menor desvio padrão do atraso é um indicativo de uma melhor qualidade para a comunicação de voz.

Na Figura 27, apresenta-se o resultado da variação do atraso médio de pacotes de voz em função da carga de dados submetida. Observa-se que no DQCA-RP, como acontece com os resultados relativos ao atraso médio de pacotes de voz, a redução do desvio padrão do atraso destes pacotes chega a 81,4% e 74,6% sobre os valores obtidos com os esquemas DQCA-DIF1 e DQCA-DIF2 respectivamente, mantendo-se sempre abaixo de 10 *ms* independente da carga de dados submetida.

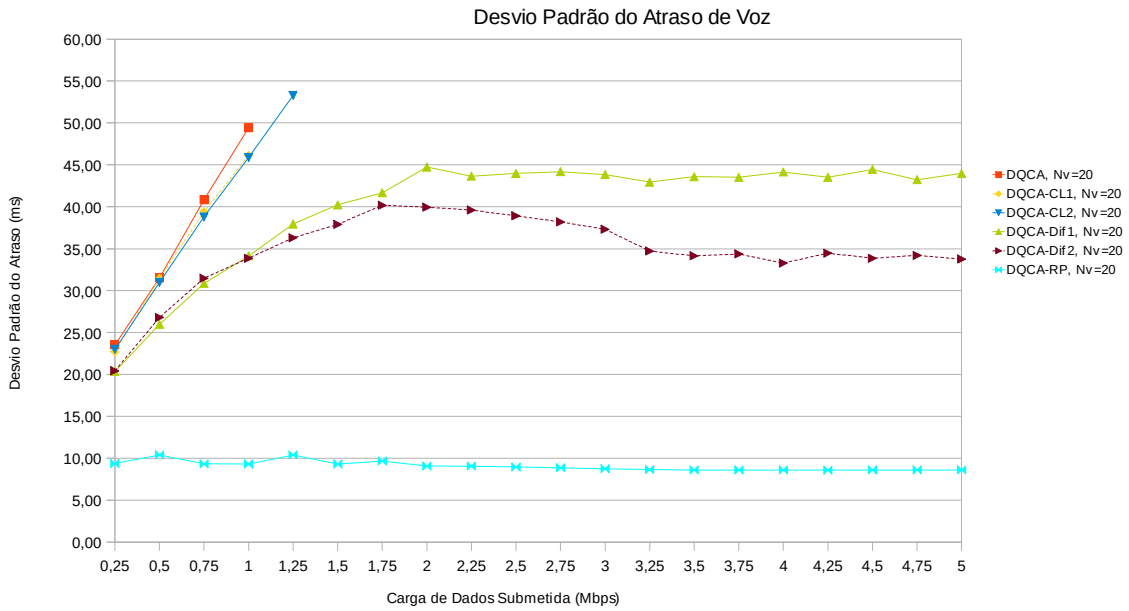

Figura 27: Desvio padrão do atraso de voz em função da carga de dados submetida

Observa-se na Figura 27, que da mesma forma que acontece com os valores de atraso médio dos pacotes de voz, com os esquemas DQCA-DIF2 e DQCA-RP a variação do atraso médio dos pacotes de voz cai com o aumento da carga de dados. A explicação para este comportamento é a mesma comentada na seção 4.3.2 para as curvas do atraso médio.

As Figuras 28 e 29 ilustram a distribuição dos valores de atrasos sofridos pelos pacotes de voz, permitindo uma melhor visualização do *jitter* na rede. As simulações foram realizadas com 20 terminais de voz, com cargas submetida de dados de 3 *Mbps* (Figura 28) e 5 *Mbps* (Figura 29) e foram considerados somente os dois protocolos que apresentaram os melhores resultados em atraso e *jitter*, o DQCA-Dif2 e o DQCA-RP. Com base nas figuras, observa-se que no protocolo DQCA-RP há uma menor dispersão no atraso. Como pode ser observado na Figura 28, com a carga submetida de dados de 3 *Mbps*, no DQCA-RP, 32,32% dos pacotes tiveram atrasos entre 2 e 4 milisegundos; neste mesmo cenário, 70,34% dos pacotes tiveram atrasos entre 0 e 6 milisegundos, contra 16,64% com o DQCA-Dif2.

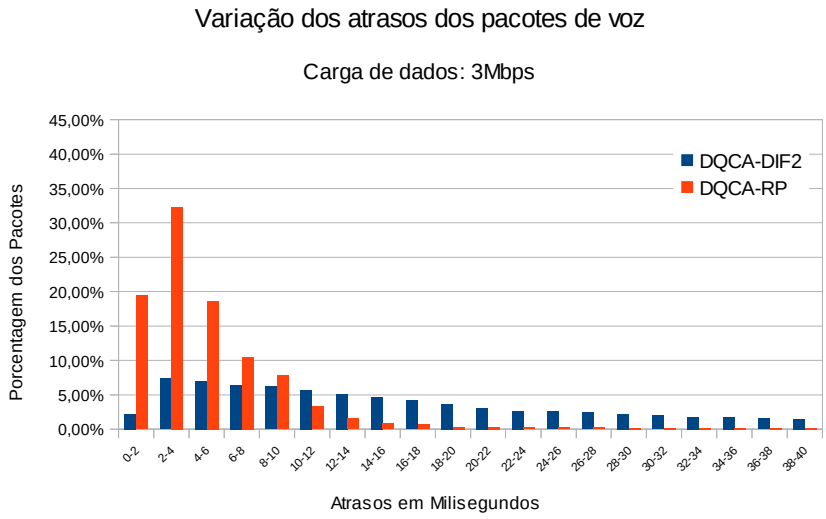

Figura 28: Histograma dos atrasos de voz obtidos com carga de dados submetida de 3 Mbps

De acordo com a Figura 29, com a carga submetida de dados de 5 *Mbps*, no DQCA-RP, 35,2% dos pacotes tiveram atrasos entre 0 e 2 milisegundos e 39,68 tiveram atrasos entre 2 e 4 milisegundos; neste mesmo cenário, 87,84% dos pacotes tiveram atrasos entre 0 e 6 milisegundos, contra 23,29% com o DQCA-Dif2.

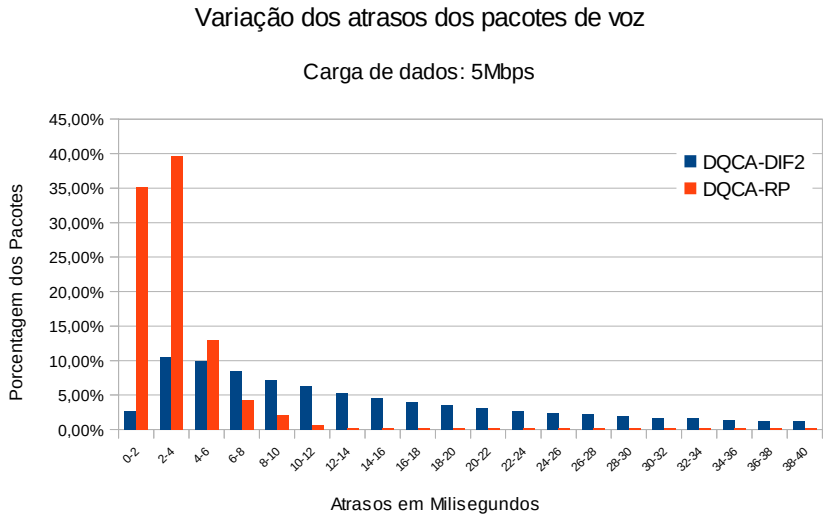

Figura 29: Histograma dos atrasos de voz obtidos com carga de dados submetida de 5 Mbps

Na Tabela 7 são apresentados os valores mínimo, médio e máximo dos atrasos dos pacotes de voz e o *jitter* máximo, para os esquemas DQCA-Dif2 e DQCA-RP, com 20

terminais de voz e variando a carga submetida de dados em 1, 2, 3, 4 e 5 *Mbps*,. A partir destes dados foi possível determinar o *jitter* máximo para cada cenário, subtraindo-se o atraso mínimo do máximo. De acordo com a tabela, entre os cenários analisados, o maior valor de *jitter* máximo obtido com o DQCA-RP foi de 87,129 milisegundos, quando a carga de dados submetida é de 1 *Mbps*. O *jitter* máximo para o esquema DQCA-Dif2 sempre se aproxima de 300 milisegundos, que é o limite de atraso para que os pacotes sejam descartados.

Tabela 7: Atraso mínimo, máximo e médio de pacotes de voz

Carga de Dados	Protocolo	Atraso mínimo	Atraso Médio	Atraso Máximo	*Jitter* máximo
1	DQCA-DIF2	0,740	25,873	299,583	298,844
	DQCA-RP	0,275	9,388	87,404	87,129
2	DQCA-DIF2	0,735	32,796	298,453	297,718
	DQCA-RP	0,206	7,832	81,783	81,577
3	DQCA-DIF2	0,732	32,240	299,414	298,682
	DQCA-RP	0,233	6,359	76,615	76,382
4	DQCA-DIF2	0,742	27,364	296,734	295,992
	DQCA-RP	0,209	5,048	88,350	88,142
5	DQCA-DIF2	0,777	24,642	295,488	294,711
	DQCA-RP	0,211	4,691	72,590	72,379

4.3.4. Porcentagem de perdas de pacotes de voz

O nível de perda de pacotes é um parâmetro especialmente importante para tráfego de voz. Em nossas análises consideramos que pacotes de voz são descartados (perdidos) quando sofrem atrasos superiores a 300 *ms* e que esta classe de tráfego não tolera perdas acima de 1% do total dos pacotes gerados. A porcentagem de perdas de pacotes de voz é dada pela Eq.(10):

$$L = \frac{P}{G} \times 100 \quad (10)$$

Onde, L é o percentual de perdas, P o total de pacotes perdidos e G o total de pacotes gerados.

Na Figura 30 são mostrados os resultados de perdas de pacotes de voz obtidos nas simulações, em função da carga de dados submetida, considerando 10 terminais de voz compartilhando o canal. De acordo com o gráfico observa-se que, com os esquemas que não priorizam os terminais de voz nas decisões de escalonamento, DQCA, DQCA-CL1 e DQCA-CL2, o limite de perdas de pacotes de voz (1%) é alcançado antes da carga de dados submetida chegar a 1,5 *Mbps*. Os esquemas DQCA-DIF1 e DQCA-DIF2 apresentam baixos índices de perdas, com o segundo apresentando um desempenho melhor devido à utilização da técnica de escalonamento oportunista. De todos os protocolos estudados, claramente o DQCA-RP é o que apresenta o melhor desempenho com relação a perdas de pacotes de voz. Como pode ser observado no gráfico, com o DQCA-RP praticamente não há perdas de pacotes de voz com qualquer carga de dados submetida. Isto se deve à reserva de *time-slots* para os pacotes de voz à medida que eles são gerados nos terminais, o que reduz os atrasos e conseqüentemente os descartes de pacotes.

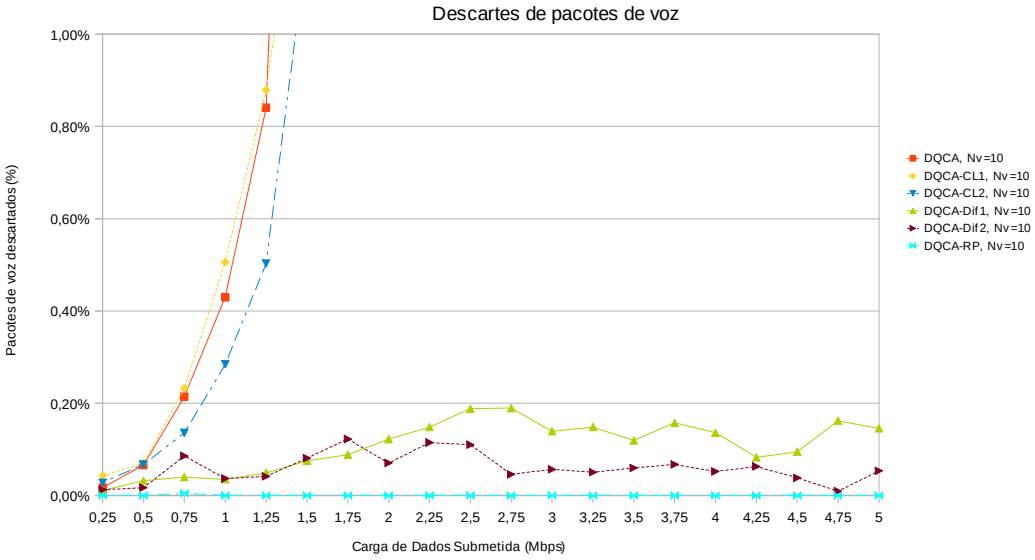

Figura 30: Descartes de pacotes de voz em função da carga de dados submetida

Capítulo 5 - Conclusões e trabalhos futuros

Nesta dissertação foram apresentados e analisados os protocolos PRMA e DQCA, que serviram de base para a idealização do esquema proposto neste trabalho. Foram apresentadas algumas variantes destes protocolos, que visam melhorar alguns aspectos de desempenho ou resolver limitações. Também foram introduzidas algumas soluções *cross-layer* encontradas na literatura, apresentando uma proposta de classificação para estas soluções em quatro tipos diferentes.

Um novo protocolo foi proposto, baseado no DQCA, chamado DQCA-RP, que visa reduzir o atraso médio e a variação do atraso de pacotes de voz, em um canal compartilhado por terminais de voz e de dados. O esquema proposto mantém a estabilidade e a alta vazão alcançável com o DQCA, através da execução em paralelo dos processos de resolução de colisões e de transmissão de dados, do uso de *minislots* na contenção e de técnicas *cross-layer* nas decisões de escalonamento. Para melhorar o atraso médio e *jitter* do DQCA, no DQCA-RP foi implementado um esquema de reserva baseado no protocolo PRMA, que atribui *time-slots* reservados aos terminais de voz à medida que seus pacotes são gerados, enquanto os terminais de dados transmitem seus pacotes em *time-slots* que não estiverem reservados, de modo semelhante ao do DQCA básico. Para melhorar a vazão, entre os terminais de dados é empregada uma solução *cross-layer*, através da qual os terminais de dados com melhores condições nos enlaces têm prioridade na decisão de escalonamento.

Para testar o desempenho do esquema proposto foi implementado um simulador orientado a eventos discretos, escrito em linguagem Java, projetado especificamente para simular o funcionamento do DQCA e suas variantes, coletando parâmetros que podem ser mapeados em métricas de desempenho a serem analisadas. O desempenho do esquema proposto foi comparado através de simulações por computador com os de

outras variantes do DQCA encontradas na literatura e os resultados obtidos foram bastante promissores. Foram considerados nas comparações de desempenho os esquemas DQCA, DQCA-CL1, DQCA-CL2, DQCA-DIF1, DQCA-DIF2 e DQCA-RP apresentados na Seção 4.1. Os três primeiros esquemas tratam pacotes de voz e de dados da mesma forma, sem priorizar um ou outro tipo de tráfego. Apesar do DQCA-CL1 prover uma vazão de dados alta, nenhum dos três protocolos mantém os requisitos de QoS dos terminais de voz, com carga de dados igual ou superior a 1,5 *Mbps*. A diferença entre os esquemas DQCA-DIF1 e DQCA-DIF2 é que o segundo emprega o conceito de escalonamento oportunista e, portanto, o seu desempenho é sempre superior ao do primeiro.

Analisando os resultados obtidos e discutidos no Capítulo 4, podemos concluir que o esquema proposto nesta dissertação cumpre os objetivos definidos e almejados durante a sua concepção. Além de manter os benefícios de desempenho geral do sistema, já alcançados com as outras variantes do DQCA apresentadas, o esquema proposto apresenta os níveis de atraso e *jitter* de voz consideravelmente mais baixos, comparados aos do DQCA e as outras variantes analisadas. De acordo com os resultados obtidos nas simulações, para os cenários definidos, apesar de uma pequena perda em vazão de 1,8% com relação ao DQCA-DIF2, com o DQCA-RP há uma redução no atraso médio de cerca de 88,3% sobre o DQCA-DIF1 e de 82,3% sobre o DQCA-DIF2. A redução do desvio padrão do atraso de pacotes de voz chega a 81,4% e 74,6% sobre os esquemas DQCA-DIF1 e DQCA-DIF2 respectivamente, além de manter os níveis de perdas de voz próximos a zero.

Portanto, em relação à meta inicial de melhoria de QoS para os terminais de voz, pode-se dizer que o objetivo foi amplamente atingido. As sugestões para trabalhos futuros incluem:

Analisar a coexistência e intercomunicação de terminais executando o DQCA e/ou suas variantes com terminais executando outros protocolos, como o legado 802.11.

Criação de Modelos analíticos para protocolos baseados no DQCA.

Analisar e implementar o DQCA-RP com outras classes de serviço diferentes. Testar outros parâmetros e dimensionamentos dos quadros e outros cenários de simulação.

Implementar nova proposta para viabilizar que o protocolo DQCA-RP trabalhe simultaneamente com terminais empregando codificadores de voz com taxas variadas. Uma possível solução poderia ser baseada no AP informar as taxas dos codificadores de voz dos terminais, de forma que os incrementos dos *timer's* dos terminais sejam realizados de acordo com estes valores.

Quando um terminal de voz no DQCA-RP demora a conseguir o acesso, o *timer* já se inicia com uma defasagem, gerando atraso extra. Uma solução que poderia ser proposta seria criar um mecanismo que permita ao terminal solicitar a redução do seu *timer*.

Também poderia se realizar adaptações no DQCA-RP para redes *ad-hoc* ou testar outros parâmetros ou modelos da camada física.

Referências Bibliográficas

[1] CHEN, T. M.; WALRAND, J. and MESSERSCHMITT, D. G. **Dynamic Priority Protocols for Packet Voice**, IEEE Journal on Selected Areas in Communications, Vol. 7, no. 5, June 1989.

[2] **IEEE Std. 802.11-1999**, Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. IEEE Std. 802.11, 1999 edition.

[3] TANENBAUM, A. **Computer Networks**. 4. Ed., Nova Jersey: Prentice Hall, 2003.

[4] SHAKKOTTAI, S.; RAPPAPORT, T.; KARLSSON, P. **Cross-Layer Design for Wireless Networks**. IEEE Communications Magazine, Vol. 41, no. 10, pp. 74-80, Oct. 2003.

[5] Kartsakli, E.; Cateura, A.; Alonso, J.; Verikoukis, C. and Alonso L. **Cross-Layer Enhancement for WLAN Systems with Heterogeneous Traffic based on DQCA**, IEEE Communications Magazine, vol. 46, no. 6, pp. 60–66, 2008.

[6] Kartsakli, E.; Cateura, A.; Verikoukis, C. and Alonso, L. **A Cross-Layer Scheduling Algorithm for DQCA-based WLAN Systems with Heterogeneous Voice-Data Traffic**, The 14th IEEE Workshop on Local and Metropolitan Area Networks, 2005. LANMAN 2005, pp. 1–6. doi: 10.1109/LANMAN. 2005.

[7] Alonso, L; Ferrús, R. and Agustí, R. **WLAN Throughput Improvement via Distributed Queuing MAC**, IEEE Commun. Lett., vol. 9, no. 4, pp. 310–12, Apr. 2005.

[8] **IEEE Std. 802.11n-2009**, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 5: Enhancements for Higher Throughput, IEEE, 2009.

[9] **IEEE Std. 802.11i-2004**, Part 11. Wireless LAN Medium Access Control (MAC) and Physical *Layer* (PHY) Specifications. Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Std., 2004.

[10] **IEEE Std. 802.11e-2005**, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, IEEE, 2005.

[11] SRIVASTAVA, V. and MOTANI, M. **Cross-Layer Design:** A Survey and the Road Ahead, IEEE Communications Magazine, vol. 43, pp. 112-119, December 2005.

[12] JI, Z.; YANG, Y.; ZHOU, J.; Takai, M. and Bagrodia, R. **Exploiting medium access diversity in rate adaptive wireless LANs**, in Proc. ACM Annual International Symposium on Mobile Computing and Networks, Philadelphia, PA, Oct. 2004.

[13] WU, D.; CI, S. and WANG, H. **Cross-Layer Design for Multimedia Delivery over Wireless Networks**. Proc. IEEE International Symposium on Multimedia, 407-414, 2006.

[14] ABD EL AL, A.; SAADAWI, T. and LEE, M. **A Cross-Layer Optimized Error Recovery Mechanism for Real-Time Vídeo in ad-hoc Networks**. Proc. International Conference on Parallel and Distributed Systems, Vol. 2, 2006.

[15] ELBATT, T. and EPHREMIDES, A. **Joint Scheduling and Power Control for Wireless Ad Hoc Networks**, IEEE Trans. Wireless Commun., vol. 3, no. 1, pp. 74–85, Jan. 2004.

[16] TONG, L.; NAWARE, V. and VENKITASUBRAMANIAM, P. **Signal Processing in Random Access**. IEEE Signal Processing, 21 (5): 29-39, Set. 2004.

[17] LIU, Q.; ZHOU, S. and GIANNAKIS, G. **Cross-Layer Combining of Adaptive Modulation and Coding with Truncated ARQ over Wireless Links**. IEEE Transactions on Wireless Communications, 3 (5): 1746-1755, Set. 2004.

[18] LIN, X.; SHROFF, N. B. and SRIKANT, R. **A tutorial on cross-layer optimization in wireless networks**. IEEE J. Sel. Area Commun., vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[19] GYASI-AGYEI, A. and KIM, S. L., **Cross-Layer Multiservice Opportunistic Scheduling for Wireless Networks**. IEEE Communications Magazine, vol. 44, no. 6, pp. 50–57, Jun. 2006.

[20] GOODMAN, D. J.; VALENZUELA, R. A.; GAYLIARD, K. T. and RAMAMURTHI, B. **Packet reservation multiple access for local wireless communications**. IEEE Transactions on Communications, pp. 885-890, Aug. 1989.

[21] HANZO, L.; CHEUNG, J. C. S.; STEELE, R. and WEBB, W. T. **Performance of PRMA Schemes via Fading Channels**, IEEE 43rd Vehicular Technology Conference, (1993)

[22] BIANCHI, G.; BORGONOVO, F.; FRATTA, L.; MUSUMECI, L. and ZORZI, M. **CPRMA: The centralized packet reservation multiple access for local wireless communications**, in: Proc. GLOBECOM ’94, Vol. 3 (1994) pp. 1340–1345, 1994.

[23] JANI, L. **Packet Reservation Multiple Access (PRMA) for Joint Speech and Data Systems**, Nokia Research, NOKIA Group Finland, November 23, 2002.

[24] JALLOUL, L. M. A.; NANDA S. and GOODMAN, D. J. **Packet Reservation Multiple Access over Slow and Fast Fading Channels**, In proceedings of the IEEE 40th Vehicular Technology Conference, Orlando, FL, USA, pp. 354-359, May 1990.

[25] CHUA, K. C. and TAN, W. M. **Modified Packet Reservation Multiple Access Protocol**, IEEE Electronics Letters, Vol. 29, Issue 8, pp. 682-684, April 1993.

[26] DYSON, D. A. and HAAS, Z. J. **A Dynamic Packet Reservation Multiple Access Scheme for Wireless ATM**, in: Proc. MILCOM ’97 (1997).

[27] DYSON, D. A. and HAAS, Z. J. **A Dynamic Packet Reservation Multiple Access Scheme for Wireless ATM**, Mobile Networks Applications, vol. 4, pp. 87–99, 1999.

[28] DYSON, D. A. and HAAS, Z. J. **The Dynamic Packet Reservation Multiple Access Scheme**, IEEE Wireless Communications and Networking Conference 1999, vol.2, 555 – 559, New Orleans, LA , USA, 1999.

[29] DYSON, D. A. and HAAS, Z. J. **On the Performance of the Dynamic Packet Reservation Multiple Access Scheme**. MILCOM ‘99, vol.1, 256 – 260, 1999.

[30] ELNOUBI, S. and ALSAYH, A. M. **Packet Reservation Multiple Access (PRMA)-Based Algorithm for Multimedia Wireless Sys**tem, IEEE Transactions on Vehicular Technology, Vol. 53, pp.215–222, 2004.

[31] VERIKOUKIS, C.; ALONSO, J.; KARTSAKLI, E.; CATEURA, A. and ALONSO, L. **Cross-Layer Enhancement for WLAN Systems Based on a Distributed Queuing MAC Protocol**. Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd , vol.3, no.pp. 1293- 1297, 07-10 May 2006.

[32] CATEURA, A.; VERIKOUKIS, C. and ALONSO, L. **Opportunistic Scheduling for WLAN Systems using Cross-Layer Techniques and Distributed MAC**.

Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd, vol.1, no.pp. 221- 224, 28-25 Sept., 2005.

[33] BIANCHI, G. **Performance Analysis of the IEEE 802.11 Distributed Coordination Function**. IEEE JSAC, vol. 18, Mar. 2000, pp. 535–47.

[34] ZÁRATE, J. A.; KARTSAKLI, E.; CATEURA, A.; VERIKOUKIS, C. and ALONSO, L. **A near-optimum cross-layered distributed queuing protocol for wireless LAN**. IEEE Wireless Communication Magazine, Special Issue on MAC protocols for WLAN, vol. 15, no. 1, pp. 48-55, 2008.

[35] ALONSO, L.; AGUSTÍ, R. and SALLENT, O. **A Near-Optimum MAC Protocol Based on the Distributed Queuing Random Access Protocol (DQRAP) for a CDMA Mobile Communication System**. IEEE JSAC, vol. 18, no. 9, pp. 1701–18, Sept. 2000.

[36] XU, W. and CAMPBELL, G. **A near perfect stable random access protocol for a broadcast channel**, in IEEE Proc. ICC'92, vol. 1, pp. 370–374, 1992.

[37] XU W. and CAMPBELL, G. **DQRAP, A distributed queueing random access protocol for a broadcast channel**, Computer Commun. Rev., vol. 23, no. 4, pp. 270–278, Oct. 1993.

[38] SIMOENS, S.; VERIKOUKIS, C.; CATEURA, A; RODRIGUEZ, J.; ET. AL., **Physical Aware Link Adaptation Including Innovative Optimization MAC/PHY Frame Architectures**, CELTIC Telecommunication Solutions, 2007. Obtido em http://www.wisquas.org/Public/publications/D31b.pdf. Acesso em 3 de Novembro de 2008).

[39] DYSON, D. A. and HAAS, Z. J. **The Dynamic Packet Reservation Multiple Access Scheme for Multimedia Traffic**, ACM/Baltzer Mobile Networks & Applications, vol. 3, pp.1640 – 1644, 1999.

[40] KARTSAKLI, E.; CATEURA, A.; ALONSO, J.; VERIKOUKIS, C. and ALONSO, L. **Opportunistic Scheduling using and Enhanced Channel State Information Update Scheme for WLAN Systems with DQCA**. IEEE VTC 2007, Dublin, Ireland.

[41] LIN, H. J. and CAMPBELL, G. **Using DQRAP (Distributed Queueing Random Access Protocol) for Local Wireless Communications**, Proceedings of Wireless '93, Calgary, Canada, July 1993, pp. 625-635.

[42] GOODMAN, D. J. **Trends in Cellular and Cordless Communications**, IEEE Communications Magazine, Vol 29, No. 6, pp 31-40, 1991.

[43] ZHANG, X. and CAMPBELL, G. **Performance analysis of distributed queueing random access protocol-DQRAP**. DQRAP Research Group Rep. 93-1, Illinois Institute of Technology, 1993.

[44] WU, C. T. and CAMPBELL, G. **Extended DQRAP (XDQRAP). A Cable TV Protocol Functioning as a Distributed Switch**, Proc. IEEE 1st Int'l. Wksp. Community Networking Integrated Multimedia Services to the Home, 1994, pp. 191–98.

[45] BRADY, P. T. **A Model for Generating On-Off Speech Patterns in Two-way Conversation**, The Bell System Technical Journal, Vol 48, No. 2, pp 2445-2472, 1969.

[46] ITU-T Recommendation P.59 Artificial Conversational Speech. Disponível em http://www.itu.int/rec/T-REC-P.59-199303-I. Acesso em 22 de Agosto de 2011.

[47] COVERDALE, P. Itu-T, **Study Group 12: Multimedia QoS requirements from a user perspective**, Workshop on QoS and user perceived transmission quality in evolving networks, Oct 2001.

[48] FERRARI, D. **Client requirements for real-time communication services**, IEEE Communications Magazine, vol. 28, no. 11, pp. 65-72, Nov. 1990.

[49] LAW, A. M. and KELTON, W. D. **Simulation Modeling and Analysis**. Third edition, ed. McGraw-Hill, 2000.

[50] DEITEL, H M.; DEITEL, P. J. **Java: como programar**. Tradutor: Edson Furmankiewicz. 6 ed. São Paulo: Pearson Prentice Hall, 2006. 1110 p.

Anexo A – O Simulador DQCA-Sim

A.1. Introdução

As ferramentas de simulação por computador permitem analisar e comparar os desempenhos de diferentes protocolos e algoritmos, segundo critérios predeterminados, auxiliando na escolha da melhor alternativa a ser empregada em equipamentos comerciais. Elas são uma importante ferramenta para estudar o desempenho dos protocolos, visto que a implementação de equipamentos reais para testes normalmente demandam altos custos em termos de dinheiro, tempo e pessoal técnico, e também devido à escassez de modelos analíticos que representem fielmente o comportamento de novas tecnologias e protocolos de comunicação. Assim, simulações por computador constituem uma solução rápida, eficiente e barata, mas para que os resultados se aproximem ao máximo das situações reais, os modelos empregados nas simulações têm que ser os mais realistas possíveis.

Existem no mercado várias ferramentas para simulação de protocolos MAC. Algumas são gratuitas e de código aberto e outras comerciais, que muitas vezes exigem o pagamento de licenças, normalmente válidas por tempo limitado. Vários fatores devem ser considerados na escolha de uma ferramenta para simulação, mas qualquer que seja o protocolo a ser estudado, a ferramenta de simulação tem que oferecer os modelos necessários, além de atender às especificidades da tecnologia. Vale ressaltar que, para um mesmo cenário os resultados obtidos com diferentes simuladores podem variar significativamente, devido a vários fatores como os diferentes critérios empregados e simplificações que são introduzidas em maior ou menor grau pelos simuladores. Assim, não existe uma ferramenta de simulação melhor para todos os casos, além disso, algumas disponíveis no mercado são mais adequadas a uma determinada camada da pilha de protocolos e outras são apropriadas para simular protocolos baseados no IEEE

802.11 com pequenas variações, não sendo adequadas para abordagens completamente novas, com mecanismos que envolvem diferentes camadas da pilha de protocolos.

Para analisar o desempenho do esquema proposto neste trabalho, foi implementado um simulador, chamado DQCA-Sim, projetado especificamente para simular o DQCA e algumas variantes deste protocolo já citadas anteriormente, incluindo-se o DQCA-RP. Outras adaptações no DQCA que vierem a ser implementadas também poderiam ser simuladas no DQCA-Sim com um esforço relativamente pequeno em programação, principalmente se as mudanças propostas afetarem basicamente as disciplinas de fila. Ele simula o funcionamento dos protocolos com tráfego de voz e/ou dados, permitindo a definição e combinação de parâmetros variáveis, configurando os cenários a serem considerados nas baterias de simulações. Os resultados obtidos para cada cenário durante a simulação, relativos a diversas métricas de desempenho, como vazão, atrasos, *jitter*, perdas, etc., são acumulados, permitindo-se comparações entre eles.

Neste capítulo são apresentadas as principais características e a estrutura básica do DQCA-Sim. Visto que o simulador foi desenvolvido dentro do contexto deste trabalho, vale ressaltar que ele não se propõe a simular outros protocolos, como o padrão 802.11, ou o PRMA, assumindo-se pois que comparações do DQCA com estes protocolos já foram feitas em trabalhos já citados anteriormente, como em [5], [6], [34], [31] e [32]. Portanto, os estudos serão focados na comparação de desempenhos do DQCA-RP com o DQCA e algumas das suas variantes, com relação às métricas de desempenho a serem analisadas.

A.2. Principais características

O DQCA-Sim é um simulador orientado a eventos discretos [49], implementado na linguagem Java e projetado especificamente para simular o DQCA e suas variantes. Os conceitos de orientação a objetos próprios da linguagem Java que foram explorados no projeto do simulador, como herança, polimorfismo e encapsulamento, proporcionam inúmeras facilidades que auxiliam na implementação de um sistema parametrizável e de fácil ampliação e adaptação a novas demandas. Informações completas sobre classes, objetos e os conceitos de orientação a objetos em Java podem ser encontrados em [50].

Nos modelos de simulação por eventos discretos, geralmente, existe uma lista de eventos a serem tratados ou executados, sempre ordenada pelo tempo em que eles ocorrem. A execução da simulação é baseada em um *loop* através do qual são feitas as leituras e tratamentos dos eventos presentes na lista. A cada iteração no *loop*, é feita a captura do próximo evento da lista e é executada uma ação ou rotina, dependendo do evento capturado. Cada evento pode ocasionar a criação de novos eventos, que são incluídos na lista de acordo com os instantes em que ocorrerão, de modo que a lista se mantenha sempre ordenada.

Na versão atual do DQCA-Sim não foi criada uma interface gráfica para a configuração dos parâmetros de simulação, para a geração de saídas ou gráficos prontos, ou para ilustrar o andamento do processo. Entretanto, é possível programar uma série de simulações consecutivas, que podem ser configuradas com cenários diferentes, para permitir comparações entre eles, ou repetindo o mesmo cenário, a fim de garantir independência estatística dos resultados. Para cada cenário simulado, são acumulados os dados sobre todo tráfego gerado e transmitido pelos terminais de cada tipo de tráfego, assim como os dados relacionados às métricas de desempenho, como vazão, atraso médio, *jitter*, etc. Estes parâmetros podem ser gravados em arquivos texto, permitindo que sejam abertos e analisados através de vários aplicativos capazes de fazer a leitura e conversão de arquivos neste formato, como os *softwares* de planilha eletrônica.

A.3. Alguns detalhes da implementação

Nesta seção serão apresentados alguns detalhes sobre a implementação do simulador, necessários para entendimento das classes criadas, dos relacionamentos entre elas e alguns aspectos do simulador comentados nas seções subseqüentes.

No DQCA-Sim, a escolha sobre qual terminal terá a permissão para transmitir a cada quadro é feita com base em uma função que calcula os valores de prioridades virtuais de todos os terminais na fila de transmissão (DTQ) e os terminais inativos, que não estiverem na fila (pTQ=0), terão prioridade virtual (PV) igual a zero. A fila é ordenada de modo que o terminal que tiver o maior valor de PV é dito estar na cabeça da fila. Para cada um dos esquemas baseados no DQCA testados, foi implementada uma versão

diferente da função de prioridade virtual de acordo com a disciplina de fila proposta, por exemplo, no DQCA padrão que utiliza a disciplina FIFO, a fórmula que calcula a prioridade virtual dos terminais ativos é 1/*pTQ*; assim, o terminal com *pTQ* igual a um, que é o mais antigo na fila, terá o maior valor de PV e portanto ocupará a cabeça da fila. Na prática, foi criada uma classe em Java chamada EsquemaDQCA.java para a implementação das funcionalidades do DQCA padrão, que inclui um método (função) chamado getPrioridadeVirtual, o qual calcula as prioridades virtuais dos terminais de acordo com a disciplina FIFO. Depois, para cada novo esquema a ser implementado foi necessário somente criar uma classe derivada de EsquemaDQCA, escrevendo nela uma nova versão do método getPrioridadeVirtual. De acordo com o conceito de herança em Orientação a Objetos [50], uma classe derivada (subclasse) herda todos os atributos e funcionalidades da classe base (superclasse), podendo substituí-los (sobrepor). Assim, a implementação dos outros esquemas baseados no DQCA foi feita sem a necessidade de modificações profundas no restante do código.

Apesar de alguns esquemas estudados utilizarem duas filas para transmissão, a DTQ e a VTQ, para terminais de dados e de voz respectivamente, internamente no simulador é implementada uma única fila, controlada nos terminais pelos atributos *TQ* e *pTQ* já comentados. O simulador emula o comportamento, como se fossem duas filas, através da função que calcula as PVs, getPrioridadeVirtual. Nos esquemas que utilizam duas filas a fórmula da função é elaborada de modo que as prioridades virtuais dos terminais de um tipo (voz) sejam sempre maiores que as dos terminais do outro tipo (dados), quando for o caso.

A.4. Estrutura do simulador DQCA-Sim

A estrutura do simulador é composta por quatro classes consideradas principais e outras que servem para dar suporte a estas ou para empacotar algum tipo de informação ou funcionalidade. As classes que juntas formam a estrutura principal do simulador são SimuladorDQCA, TerminalDQCA, QuadroDQCA e EsquemaDQCA. Para facilitar o entendimento e a documentação, as classes implementadas no simulador são agrupadas de acordo com estas classes principais, em quatro grupos, apresentados nas próximas seções.

A.4.1. Classes dedicadas à implementação do programa principal do simulador

A.4.1.1. Classe SimuladorDQCA.java

A classe SimuladorDQCA realiza uma simulação de acordo com o cenário definido pelo usuário. Ela se baseia no método simulaCenario, que é o método principal do simulador, responsável por configurar o cenário da simulação, acionar os tratamentos dos eventos à medida que os mesmos ocorrem e preparar os resultados finais a serem retornados. O método simulaCenario recebe como argumento um objeto da classe ParametrosSimulacao, apresentada na Seção A.4.1.4, que contém os parâmetros que compõem o cenário a ser considerado na simulação e, com base nestes parâmetros, realiza inicialmente as seguintes ações:

Configura e dimensiona o quadro, através de um objeto da classe QuadroDQCA;

Configura as características do canal sem fio, através da classe Canal;

Define o esquema DQCA a ser simulado, através de um objeto da classe EsquemaDQCA ou de uma de suas subclasses apresentadas na Seção A.4.4, as quais implementam os outros esquemas baseados no DQCA;

Cria os tipos de terminais, de voz e dados, através de objetos da classe TipoTerminal. Estes objetos serão utilizados na configuração dos terminais e no armazenamento dos parâmetros de tráfego de cada tipo separadamente;

Cria as instâncias dos terminas de voz e de dados, através das classes TerminalDQCA_Voz e TerminalDQCA_Dados, tendo como atributos um dos objetos da classe TipoTerminal, armazenando-as em uma lista de terminais;

Agenda os primeiros eventos a serem tratados, como as chegadas dos primeiros pacotes dos terminais, a finalização do primeiro quadro, a finalização da simulação, a

atualização das taxas de transmissão, de acordo com o período de coerência, etc. Estes eventos são instâncias da classe Evento, que será apresentada na próxima Seção, e são armazenados em uma lista de eventos através de um vetor dinâmico.

Depois das inicializações e configuração do cenário da simulação, os eventos são tratados através de um *loop* que dura até a ocorrência do evento finaliza simulação ou opcionalmente até a quebra de um determinado requisito de QoS. A cada iteração do *loop*, um dos eventos apresentados na Tabela 8 é capturado da lista e são executadas as respectivas ações, de acordo com a classe responsável por tratá-lo. O evento que será capturado é o que possui o menor valor no atributo tempo e no caso de dois eventos no mesmo instante, o de menor valor de *token* é tratado primeiro. O *token* é um atributo da classe Evento que permite a priorização, quando dois eventos ocorrem no mesmo instante.

Tabela 8: Eventos tratados no DQCA-Sim

Tipo	Descrição	Token	Classe que trata o evento
‘q’	Finaliza o quadro	1	EsquemaDQCA ou uma de suas subclasses
‘s’	Mudança de status do terminal de voz	2	TerminalDQCA_Voz
‘m’	Nova mensagem de um terminal	3	TerminalDQCA
‘r’	Transmissão de uma requisição de acesso	4	TerminalDQCA
‘p’	Transmissão de um pacote	5	TerminalDQCA
‘t’	Atualização das taxas de transferência	6	TerminalDQCA
‘f’	Finaliza simulação	7	SimuladorDQCA

No final da simulação, ainda no método simulaCenario, os parâmetros são formatados e retornados para o programa que o chamou, através do mesmo objeto da classe ParametrosSimulacao recebido como argumento.

A.4.1.2. Classe Evento.java

A classe evento é utilizada para encapsular os eventos que ocorrem e devem ser tratados no simulador. A classe possui construtores para a criação dos eventos e métodos para acesso aos atributos da classe, apresentados na Tabela 9.

Tabela 9: Atributos da classe Evento

Atributo	Tipo	Descrição
tipo	char	É um caractere que identifica o tipo do evento. Pode ser um dos tipos listados na Tabela 8.
tempo	double	Indica o instante em que o evento deve ocorrer.
descricao	String	Descrição do evento.
terminal	TerminalDQCA	Terminal relacionado ao evento, ou *null* se nenhum.
token	int	Ficha de desempate, para o caso de tempos iguais. Pode ser um dos listados na Tabela 8.

A.4.1.3. Classe Canal.java

A classe Canal é utilizada para configurar as características do canal sem fio, a serem considerados pelo simulador. Os seus atributos são mostrados na Tabela 10.

Tabela 10: Atributos da classe Canal

Atributo	Tipo	Descrição
tempoDeCoerencia	double	Tempo de coerência do canal.
atrasoDePropagacao	double	Atraso de propagação no canal.
taxasDisponiveis	double[]	Vetor com as taxas disponíveis na camada física. Por exemplo {1, 2, 5.5, 11}
matrizTransicao	double[][]	Matriz com as probabilidades de transição entre as taxas, usada para calcular as próximas taxas dos terminais, como a matriz *P* em [5].
probabilidadesRegimePerm	double[]	Probabilidades em regime permanente de o terminal estar em cada uma das taxas. Utilizada para calcular as taxas iniciais.

A classe Canal disponibiliza alguns métodos para acessar os valores dos seus atributos e outros métodos úteis, que permitem obter aleatoriamente as taxas iniciais e as próximas taxas dos terminais e obter as taxas máximas e mínimas entre as disponíveis na camada física.

A.4.1.4. Classe ParametrosSimulacao.java

A classe ParametrosSimulacao tem dupla função, de acordo com os seus atributos, de configurar o cenário para a execução de uma simulação e obter os resultados calculados durante a simulação. Alguns campos da classe são destinados a configurar o cenário e devem ser definidos antes do início da simulação, enquanto outros têm a função de

acumular os resultados da simulação, que poderão ser utilizados nas análises e comparações dos esquemas estudados. A utilização desta classe permite a realização de seqüências de simulações, com o mesmo cenário, ou com cenários diferentes, somente modificando os parâmetros desejados. Por exemplo, para realizar várias simulações variando as cargas submetidas ou os tamanhos dos pacotes basta criar *loops* variando estes atributos e chamando o método simulaCenario da classe SimuladorDQCA, como no código da Figura 31. Os atributos da classe destinados a configuração do cenário, que são os parâmetros de entrada no simulador, são apresentados na Tabela 11.

Tabela 11: Atributos da classe ParametrosSimulacao para configurar o cenario

Atributo	Tipo	Descrição
tempoSimulacao	double	Define a duração da simulação, em segundos.
esquemaDQCA	EsquemaDQCA	Esquema a ser simulado. Pode ser um objeto da classe EsquemaDQCA ou de uma de suas subclasses, de acordo com o esquema que se deseja simular.
cargaDeDadosSubmetida	double	Carga de dados a ser gerada durante a simulação, em *bps*.
tamPacoteDados	int	Tamanho dos pacotes de dados.
tamPacoteVoz	int	Tamanho dos pacotes de voz.
taxaPacotesVoz	int	Taxa de geração dos pacotes de voz, quando no estado ON.
taxaBase	double	Taxa de transmissão dos campos de controle, como *ACK*, *CTS*, *RTS*, etc. (É utilizada a taxa mínima).
n_TerminaisDados	int	Número de terminais de dados.
n_TerminaisVoz	int	Número de terminais de voz.
limiteAtrasoDados	double	Limite de atraso de pacotes de dados.
limiteAtrasoVoz	double	Limite de atraso de pacotes de voz.
limitePerdasDados	double	Limite de perdas de pacotes de dados.
limitePerdasVoz	double	Limite de perdas de pacotes de voz.

Na Tabela 12 são apresentados os atributos de saída, relativos ao tráfego de dados. No final da simulação, estes campos armazenam alguns resultados obtidos durante a simulação para este tipo de tráfego.

Tabela 12: Atributos da classe ParametrosSimulacao para retornar os resultados dos terminais de dados

Atributo	Tipo	Descrição
throughputAlcancadoDeDados	double	Vazão de dados alcançada durante a simulação.
atrasoMedioDeDados	double	Atraso médio dos pacotes de dados.
desvioPadraoAtrasoDeDados	double	Desvio padrão dos atrasos dos pacotes de dados.
trafegoGeradoDeDados	double	Tráfego total gerado de dados.

trafegoTransmitidoDeDados	double	Tráfego total transmitido de dados.
pacotesDescartadosDeDados	long	Número de pacotes descartados de dados.

Na Tabela 13 são apresentados os atributos de saída, relativos ao tráfego de voz. No final da simulação, estes campos armazenam alguns resultados obtidos durante a simulação para este tipo de tráfego.

Tabela 13: Atributos da classe ParametrosSimulacao para retornar os resultados dos terminais de Voz

Atributo	Tipo	Descrição
throughputAlcancadoDeVoz	double	Vazão de Voz alcançada durante a simulação.
atrasoMedioDeVoz	double	Atraso médio dos pacotes de voz.
desvioPadraoAtrasoDeVoz	double	Desvio padrão dos atrasos dos pacotes de voz.
trafegoGeradoDeVoz	double	Tráfego total gerado de voz.
trafegoTransmitidoDeVoz	double	Tráfego total transmitido de voz.
pacotesDescartadosDeVoz	long	Número de pacotes descartados de voz.

A classe ParametrosSimulacao disponibiliza vários métodos para obter e definir os valores de todos os seus campos, além dos métodos somaResultados e calculaMediaDosResultados, que podem ser utilizados respectivamente para somar os parâmetros de saída obtidos em duas simulações distintas ou calcular a média destes parâmetros.

Optou-se pela utilização de uma única classe para configurar os parâmetros de entrada do simulador e para obter os resultados da simulação, em vez de duas classes separadas, para manter juntos em uma única estrutura os resultados da simulação e o cenário empregado para atingi-los, facilitando assim a análise dos dados.

A.4.1.5. Classe Main.java

A classe Main inicializa os parâmetros para configurar o cenário de cada simulação e executa o simulador. Na realidade poderia ser substituída por qualquer outro programa feito em Java. A implementação desta classe permite a automatização do processo de várias simulações em seqüência, através de chamadas ao método simulaCenario da classe SimuladorDQCA, com diferentes combinações de parâmetros.

No código mostrado na Figura 31, o programa configura os parâmetros iniciais e realiza uma bateria de simulações, variando os esquemas a serem simulados, os números de terminais de voz e as cargas submetidas de dados. Conforme pode ser observado nas três estruturas *for*, são realizadas simulações para cada um dos seis esquemas baseados no DQCA, com cinco, dez e quinze terminais de voz e variando as cargas submetidas de dados em 1, 2, 3, 4 e 5 *Mbps*. Um objeto da classe ParametrosSimulacao é utilizado para definir os parâmetros que compõem o cenário da simulação e nele ficam armazenados os resultados obtidos e acumulados durante a execução. As classes que definem os esquemas baseados no DQCA são apresentadas na Seção A.4.4.

```
public static void main(String[] args) {
  ParametrosSimulacao parametros = new ParametrosSimulacao();
  parametros.setTempoSimulacao(100);  // Tempo total de Simulação
  parametros.setTamPacoteDados(1000); // Tamanho do pacote de dados
  parametros.setTamPacoteVoz(100);    // Tamanho do pacote de Voz
  parametros.setN_TerminaisDados(20); // Nº de terminais de dados
  // Cria um vetor com os esquemas a serem simulados
  EsquemaDQCA esquemas [] = {new EsquemaDQCA     ("DQCA     "),
                             new EsquemaDQCA_CL1 ("DQCA-CL1 "),
                             new EsquemaDQCA_CL2 ("DQCA-CL2 "),
                             new EsquemaDQCA_Dif1("DQCA-DIF1"),
                             new EsquemaDQCA_Dif2("DQCA-DIF2"),
                             new EsquemaDQCA_RP  ("DQCA-RP  ")};
  System.out.println("Esquema\tCarga\tnVoz\tVazãoDados\tAtrasoVoz");
  // Loop para simular cada esquema baseado no DQCA
  for(int e = 0; e < esquemas.length ; e++){
    parametros.setEsquemaDQCA(esquemas[e]);
    // Loop para variar a carga submetida de dados de 1 a 5 Mbps
    for(double c = 1E6; c <= 5E6; c += 1E6){
      parametros.setCargaDeDadosSubmetida(c);
      // Loop para simular com 5, 10 e 15 terminais de voz
      for (int v = 5; v <= 15; v+=5) {
        parametros.setN_TerminaisVoz(v);
        // Realiza uma simulação e imprime os resultados
        SimuladorDQCA.simulaCenario(parametros);
        System.out.println(
            parametros.getEsquemaDQCA().getDescricao()+"\t"+
            parametros.getCargaDeDadosSubmetida()/1000000+"\t"+
            parametros.getN_TerminaisVoz()+"\t"+
            parametros.getThroughputAlcancadoDeDados()+"\t"+
            parametros.getAtrasoMedioDeVoz());
      }
    }
  }
}
```

Figura 31: Código para realizar uma seqüência de simulações variando alguns parâmetros

A.4.1.6. Classe NumeroAleatorio.java

A classe NumeroAleatorio é uma subclasse da classe java.util.Random da biblioteca do Java, que é responsável pela geração dos números aleatórios, de acordo com as distribuições utilizadas no simulador. Os principais métodos, que são utilizados no simulador são nextExponential, que retorna uma variável aleatória exponencial cuja média é especificada como argumento, nextDouble que retorna uma variável aleatória uniforme em ponto flutuante maior ou igual a zero e menor que um e nextInt(), que retorna um valor inteiro que é uma variável aleatória uniforme maior ou igual a zero e menor que o número especificado como argumento.

A.4.2. Classes para criar as instâncias dos terminais

Alguns comportamentos e características dos terminais são comuns tanto para terminais de dados quanto de voz, funcionando do mesmo modo para ambos. Por exemplo, todos os terminais, independente do tipo, devem ter uma lista de mensagens, uma taxa de transmissão e variáveis que representam as filas; o processo de resolução de colisões também é o mesmo para todos eles. Entretanto, alguns tipos de terminais podem possuir atributos e comportamentos específicos. Por exemplo, o modo como os pacotes são gerados e transmitidos é diferente para terminais de dados e de voz. Assim, seria inviável implementar uma única classe que representasse tanto os terminais de dados quanto os de voz. Por outro lado, implementar duas classes independentes para os dois tipos de terminais iria trazer outros problemas, pois muitos atributos e funcionalidades teriam implementações duplicadas. Além disso, teria o problema de tratar dois tipos de objetos em uma única lista (fila). A solução foi criar uma classe mais geral, e nela implementar todas as características comuns aos terminais de dados e de voz, e depois criar subclasses dela, onde seriam implementados os atributos e comportamentos específicos para cada tipo de terminal. Uma classe abstrata em Java, permite que algumas funcionalidades sejam implementadas nela e que outras funcionalidades sejam somente declaradas, de forma que as subclasses ficam obrigadas a implementá-las. Então foram criadas a classe abstrata TerminalDQCA, com a implementação dos elementos comuns e as subclasses concretas TerminalDQDA_Dados e TerminalDQCA_Voz, com as implementações específicas para os respectivos tipos de

terminais. Em implementações futuras, no caso da necessidade de um terceiro tipo de terminal como de vídeo, por exemplo, deveria ser criada uma terceira subclasse de TerminalDQCA com as implementações específicas para este tipo de tráfego. Esta solução envolve conceitos de polimorfismo, herança e classes abstratas em Java que podem ser encontrados em [50]. Nas próximas seções são mostrados os atributos e apresentados resumidamente alguns detalhes considerados mais relevantes das implementações das classes TerminalDQCA, TerminalDQCA_Dados e TerminalDQCA_Voz.

A.4.2.1. Classe TerminalDQCA.java

Os principais atributos da classe TerminalDQCA são apresentados na Tabela 14. Detalhes relativos às classes Mensagem, TipoTerminal e EsquemaDQCA que compõem alguns dos atributos, são apresentados nas seções subseqüentes.

Tabela 14: Atributos da classe TerminalDQCA

Atributo	Tipo	Descrição
TQ	int	Campo estático[1] que representa o número de terminais na fila de transmissão (DTQ).
RQ	int	Campo estático que representa o número de terminais na fila de resolução de colisões (CRQ).
pTQ	int	Representa a posição do terminal na DTQ.
pRQ	int	Representa a posição do terminal na CRQ.
taxaDisp	double	Armazena a taxa de transmissão do terminal.
listaDeMensagens	Mensagem[]	Um vetor dinâmico, contendo objetos da classe Mensagem. Cada mensagem armazena os pacotes que foram gerados no terminal.
tipo	TipoTerminal	É um objeto da classe TipoTerminal. Através deste campo são configurados alguns atributos dos terminais e são acumulados durante a simulação parâmetros de tráfego, como atraso, perdas e tráfego gerado e transmitido.
esquemaDQCA	EsquemaDQCA	Pode ser um objeto da classe EsquemaDQCA ou de uma de suas subclasses. Este campo é estático, para que todos os terminais executem o mesmo protocolo.

[1] Campo estático é um campo da classe; é um atributo que tem o mesmo valor para todas as instâncias da classe[50]. Neste contexto, o atributo TQ terá o mesmo valor para todos os terminais.

A classe TerminalDQCA disponibiliza métodos para acesso e manutenção dos seus atributos, das suas mensagens e das filas DTQ e CRQ. Nela são implementados também os métodos para tratamento dos eventos dos tipos ‘r’ e ‘t’, mostrados da Tabela 8; para os eventos dos tipos ‘s’, ‘m’ e ‘p’ foram declarados métodos abstratos, que são implementados nas subclasses TerminalDQCA_Dados e TerminalDQCA_Voz, por terem funcionamentos diferentes para terminais de dados e de voz.

A.4.2.2. Classe TerminalDQCA_Dados.java

A classe TerminalDQCA_Dados é uma subclasse da classe abstrata TerminalDQCA. Portanto, ela herda todos os seus atributos e funcionalidades e implementa todos os seus métodos abstratos, de acordo com o comportamento dos terminais de dados. Os principais métodos abstratos implementados nesta classe são trataEventoNovaMsg e trataEventoTransmitePacote, que tratam os eventos dos tipos ‘m’ e ‘p’, apresentados na Tabela 8.

A.4.2.3. Classe TerminalDQCA_Voz.java

A classe TerminalDQCA_Voz é uma subclasse da classe abstrata TerminalDQCA. Portanto, ela herda todos os atributos e funcionalidades dessa classe e implementa todos os seus métodos abstratos de acordo com o comportamento dos terminais de voz. Os principais métodos abstratos implementados nesta classe são trataEventoNovaMsg, trataEventoTransmitePacote e trataEventoNovoStatus, que tratam os eventos dos tipos ‘m’, ‘p’ e ‘s’, mostrados na Tabela 8.

Nesta classe foi necessário implementar novos atributos e funcionalidades adequados às características do tráfego de voz, que funcionam de forma diferente do tráfego de dados. Cada Terminal de voz, em um determinado instante, pode estar no estado *ON* (ativo) ou *OFF* (em silêncio). No estado *OFF* não são gerados pacotes no terminal, enquanto no estado *ON* são gerados pacotes a uma taxa constante de 13 *Kbps*, configurável no simulador. Além dos campos herdados da classe base TerminalDQCA, foram incluídos os atributos mostrados na Tabela 15.

Tabela 15: Atributos da classe TerminalDQCA_Voz

Atributo	Tipo	Descrição
status	int	Indica se o terminal está ativo (*ON*=1) ou em silêncio (*OFF*=0).
taxaCBR	double	Define a taxa em que os pacotes são gerados, quando no estado *ON*.
P_on	double	Constante estática, que define a duração média dos períodos *ON*.
P_off	double	Constante estática, que define a duração média dos períodos *OFF*.

Nesta classe, além dos métodos abstratos da classe TerminalDQCA que foram implementados, foram criados outros métodos para implantação do modelo de tráfego ON/OF dos terminais de voz.

A.4.2.4. Classe TipoTerminal.java

A classe TipoTerminal empacota os tipos de terminais considerados nas simulações. O simulador cria uma instância desta classe para cada tipo de terminal, por exemplo, um objeto para configurar terminais de voz e outro para terminais de dados. Depois, na criação dos terminais, um desses objetos é configurado no atributo tipo da classe TerminalDQCA.

Esta classe possui alguns campos para configuração dos terminais de cada tipo. Outros atributos são dedicados a acumular dados que irão compor os resultados da simulação. Os atributos para configuração dos terminais são apresentados na Tabela 16.

Tabela 16: Atributos da classe TipoTerminal para configuração dos terminais

Atributo	Tipo	Descrição
nivelDePrioridade	int	Define o nível de prioridade, usado no cálculo da prioridade virtual, a ser utilizada nas decisões de escalonamento. No simulador foi definido o valor 1 para voz e 0 para dados
charTipo	char	Caractere que identifica o tipo, utilizado no simulador para verificações (‘D’ para dados e ‘V’ para voz).
tamanhoPacote	int	Tamanho do pacote (em *bytes*).
limiteAtraso	double	Limite de atraso de pacotes.
limitePerdas	double	Taxa de perdas tolerável.
tamanhoMedioDasMensagens	int	Tamanho médio das mensagens que chegam nos terminais. Em terminais de voz é o tamanho do pacote.
intervaloMedioEntreMensagens	double	Intervalo médio entre as chegadas de mensagens, ou de pacotes, no caso de voz. Para terminais de voz deve ser fixo (*CBR*). Para terminais de dados deve ser calculado de acordo com a carga de dados desejada no sistema.

Os campos da classe TipoTerminal dedicados a acumular os dados estatísticos são apresentados na Tabela 17.

Tabela 17: Atributos da classe TipoTerminal para acumular dados estatísticos da simulação

Atributo	Tipo	Descrição
trafegoGeradoAcumulado	long	Tráfego gerado pelos terminais do tipo, durante a simulação.
trafegoTransmitidoAcumulado	long	Tráfego transmitido pelos terminais do tipo, durante a simulação.
numPacotesDescartadosAcumulado	long	Número de pacotes descartados durante a simulação.
listaDeAtrasosDePacotes	double[]	Vetor dinâmico contendo os atrasos de todos os pacotes transmitidos. Utilizado no cálculo do desvio padrão.

As funcionalidades da classe são disponibilizadas através de vários métodos, para se obter e alterar os valores dos seus atributos e outros que permitem calcular, com base na lista de atrasos, os valores de atraso médio e desvio padrão dos atrasos.

A.4.2.5. Classe Mensagem.java

A classe Mensagem permite armazenar e manipular as mensagens dos terminais. Os campos que compõem a classe são apresentados na Tabela 18.

Tabela 18: Atributos da classe Mensagem

Atributo	Tipo	Descrição
listaDePacotes	Pacote[]	Contém um vetor dinâmico de objetos do tipo Pacote, que é uma classe interna de Mensagem, comentada a seguir. Representa os pacotes que compõem cada mensagem de um terminal.
msgStatusFechada	boolean	Atributo criado especificamente para tratar mensagens de terminais de voz. Valor *false* indica que novos pacotes que chegarem entrarão nesta mensagem; valor *true* indica que esta mensagem não recebe mais pacotes.

Além dos métodos para obter informações sobre os pacotes da mensagem, a classe disponibiliza os métodos push() e pop(), para inserir e remover pacotes da mensagem, respectivamente, e descartaPacotes() para descartar os pacotes da mensagem cujo atraso exceder o limite máximo permitido.

A classe interna Pacote configura cada pacote presente nas mensagens dos terminais. Esta classe possui os atributos apresentados na Tabela 19 e os métodos para obter os valores destes campos.

Tabela 19: Atributos da classe interna Pacote

Atributo	Tipo	Descrição
tamanho	int	Representa o tamanho do pacote (em *bytes*).
instanteGeracao	double	Representa o instante em que o pacote foi gerado. No momento da transmissão, este valor é utilizado no cálculo do atraso do pacote.

A.4.3. Classes para empacotar o quadro DQCA

A.4.3.1. QuadroDQCA.java

Conforme comentado no Capítulo 2, o quadro DQCA é composto pelos campos de contenção, onde são enviadas as requisições de acesso (RTS), o campo *Data* para transmissão do pacote e o campo de realimentação (FBP) que envia informações aos terminais no sentido *downlink*. O quadro é implementado no simulador através da classe QuadroDQCA, que utiliza as classes internas, CampoContencao, CampoData e CampoFBP para implementar os seus três campos. A classe QuadroDQCA possui os atributos mostrados na Tabela 20 e disponibiliza diversos métodos que permitem acesso ao estado atual do quadro. Tem ainda outros métodos que provêem funcionalidades úteis, como obter a duração do quadro, configurar o quadro seguinte, de acordo com o terminal que irá transmitir, e montar o campo FBP no final do quadro, de acordo com os estados dos campos de contenção e de dados.

Tabela 20: Atributos da classe QuadroDQCA

Atributo	Tipo	Descrição
campoContencao	CampoContencao	É um objeto da classe interna, CampoContencao, que empacota os *minislots* de contenção.
campoData	CampoData	É um objeto da classe interna CampoData, que representa a parte do quadro dedicada a transmissão dos pacotes. É o campo de dados.
campoFBP	CampoFBP	É um objeto da classe interna CampoFBP, que representa o pacote de

		realimentação, transmitido no sentido *downlink*.
instanteDoInicioDoQuadro	double	Armazena o instante do início do quadro.
numQuadro	int	É um número seqüencial, que enumera o quadro, para controle do simulador.
M_MINISLOTS=3	int	Indica o número de *minislots* de contenção.
tamanhoPacotePadrao	int	Define o tamanho de pacote a ser considerado, quando a fila de transmissão estiver vazia. Neste caso, não tem como saber qual tipo de terminal irá transmitir um pacote.
tmpMinislot	double	Define a duração de um *minislot*.
tmpSifs	double	Define a duração de cada intervalo SIFS.
tmpCabecPHY	double	É o tempo necessário para transmissão do cabeçalho da camada física.
tamCabecMAC	int	Define o tamanho do cabeçalho MAC.

A.4.3.2. Classe interna: CampoContencao

A classe interna CampoContencao controla a parte do quadro dedicada à contenção, ou seja, os *minislots*. Os atributos da classe são mostrados na Tabela 21 e são implementados métodos para incluir novas requisições, para obter a duração da janela de contenção e métodos para verificar se houve sucesso ou colisão em um determinado *minislot*.

Tabela 21: Atributos da classe interna CampoContencao

Atributo	Tipo	Descrição
vetMinislots	int[]	É um vetor que mantém o número de terminais que enviaram requisições em cada *minislot*. Cada posição do vetor representa um *minislot*. Então, valor 0 indica nenhuma requisição, valor 1 indica sucesso e valores maiores que 1 indicam colisão no *minislot*.
tmpMinislot	double	Representa a duração de um *minislot*.

A.4.3.3. Classe interna: CampoData

A classe interna CampoData controla a parte do quadro dedicada à transmissão dos pacotes. Os atributos da classe CampoData são mostrados na Tabela 22.

Tabela 22: Atributos da classe interna CampoData

Atributo	Tipo	Descrição
terminal	TerminalDQCA	Objeto que representa o terminal que irá transmitir o pacote no quadro, ou *null* se nenhum.
tamanhoPacote	int	Tamanho do pacote a ser transmitido. É configurado de acordo com o terminal que irá transmitir ou, se nenhum terminal recebe o valor do atributo tamanhoPacotePadrao da classe QuadroDQCA.
taxaTransm	double	Taxa de transmissão com a qual o pacote será enviado.
bitFinalMsg	int	*Bit* de final de mensagem, transmitido junto com o campo *data*.
Ack	boolean	Confirmação de recebimento do pacote. Não está sendo utilizado na implementação atual, mas permitiria retransmissões, por exemplo.

Além dos métodos básicos para obter e modificar os atributos da classe, a classe CampoData tem o método getTempoTransmissao, que calcula o tempo de transmissão do campo Data, considerando o Pacote, os cabeçalhos PHY e MAC e o atraso de propagação.

A.4.3.4. Classe interna: CampoFBP

A classe interna CampoFBP implementa o campo de realimentação (*feedback*) enviado pelo AP no final do quadro. Os atributos desta classe, apresentados na Tabela 23, são preenchidos de acordo com os estados dos campos de contenção e *data*.

Tabela 23: Atributos da classe interna CampoFBP

Atributo	Tipo	Descrição
Cts	char[]	Vetor que contém as informações dos *feedbacks* das requisições enviadas nos *minislots* do campo contenção. Cada elemento pode ter um dos valores, ‘s’, ‘c’ ou ‘i’, conforme o estado do *minislot* seja sucesso, colisão ou vazio.
Ack	boolean	É a replicação da confirmação do recebimento do pacote no campo *data*.
bitFinalMsg	boolean	É a replicação do *Bit* de final de mensagem recebido com o campo *data*.
tamCTS	int	Representa o tamanho de cada CTS transmitida no FBP.
tamFC	int	Tamanho do controle de quadro (FC).

tamACK	int	Tamanho do *ACK*.
tamFCS	int	Tamanho do controle de seqüência de quadros (FCS).
overheadCrossLayer	int	Tamanho do overhead. Somente se o esquema DQCA considera as taxas dos terminais no escalonamento. Sua configuração depende do campo usaCrossLayer da classe EsquemaDQCA.

A classe possui, além de métodos para obter e modificar os atributos e para calcular o tempo de transmissão do campo, outros para obter informações dos *minislots*, como o número de requisições, o número de colisões e de sucessos, etc.

A.4.4. Classes para implementação dos esquemas baseados no DQCA

Como comentado na Seção A.3, o que diferencia a maioria dos esquemas baseados no DQCA testados é a disciplina de fila empregada. A classe EsquemaDQCA implementa o funcionamento do protocolo DQCA, sendo que a disciplina de fila, ou seja, a ordem que os terminais ocupam na fila de transmissão, é definida através do método getPrioridadeVirtual, que nessa classe resulta em uma fila FIFO. Então, para cada esquema a ser testado foi criada uma subclasse de EsquemaDQCA, que herda todo o comportamento do protocolo DQCA já implementado e sobrepõe o método getPrioridadeVirtual por uma versão que utiliza uma fórmula diferente, resultando em outra regra de fila. Como será observado, dos esquemas testados, somente o DQCA-RP exigiu outras modificações além desta para a implementação.

Na próxima seção será descrita de forma resumida a classe EsquemaDQCA que implementa o DQCA básico e nas seções subseqüentes serão apresentadas as classes que implementam os outros protocolos baseados no DQCA, comentando a disciplina de fila empregada em cada uma delas e mostrando as fórmulas utilizadas nos respectivos métodos getPrioridadeVirtual. Os detalhes relativos aos esquemas testados são abordados no Capítulo 4.

A.4.4.1. Classe EsquemaDQCA.java

A classe EsquemaDQCA possui um único campo do tipo *boolean* chamado usaCrossLayer, que indica se o esquema utiliza ou não as informações sobre os estados dos enlaces dos terminais nas decisões de escalonamento. O simulador utiliza esta informação para decidir sobre a inclusão ou não de *overhead* extra no quadro. Os métodos implementados são comentados a seguir:

getPrioridadeVirtual: calcula a prioridade virtual (PV) do terminal recebido como argumento. Para os terminais que não estão na fila de transmissão, a PV retorna sempre zero, enquanto os terminais ativos (com $pTQ > 0$) terão o valor da PV calculado pela Eq.(11),

$$PV = 1.0 / pTQ. \tag{11}$$

Esta fórmula é empregada no protocolo DQCA básico, no qual a fila de transmissão emprega a disciplina FIFO. Como resultado, o terminal que tiver o valor de *pTQ* igual a um, ou seja, o terminal que está a mais tempo na fila terá o maior valor de VP, ocupando assim a cabeça da fila.

Esta classe possui ainda métodos dedicados ao tratamento do evento Finaliza quadro, mostrado na Tabela 8. Implementa as ações que cada terminal deve executar no final do quadro, como a atualização das filas, confirmação de transmissão do pacote, caso tenha transmitido, removendo-o do buffer. Cria eventos para transmissão de requisições e finalmente define qual terminal irá transmitir um pacote no próximo quadro, configurando-o de acordo.

A.4.4.2. Classe EsquemaDQCA_CL1.java

No esquema DQCA-CL1 apresentado no Capítulo 4, os terminais que ocupam a fila de transmissão são ordenados pela suas taxas de transmissão. No caso de terminais com a mesma taxa, o algoritmo de ordenação considera o tempo de permanência do terminal

na fila, indicado pelo seu valor de *pTQ*. Como este esquema só modifica a disciplina de fila, em relação ao DQCA básico, foi necessário sobrepor somente o método getPrioridadeVirtual, substituindo a Equação que calcula a PV pela Eq.(12):

$$PV = taxaDisp; \tag{12}$$

A.4.4.3. Classe EsquemaDQCA_CL2.java

Considerando que no esquema DQCA-CL2 a prioridade virtual deve ser calculada pela razão da taxa pela posição na fila, a fórmula no método getPrioridadeVirtual foi escrita como a Eq.(13):

$$PV = taxaDisp / pTQ; \tag{13}$$

A.4.4.4. Classe EsquemaDQCA_Dif1.java

A proposta do esquema DQCA-DIF1 é dar prioridade aos terminais de voz utilizando duas filas FIFO, uma para terminais de dados e outra para terminais de voz, mais prioritária. Como comentado anteriormente, na Seção A.3, os esquemas que propõem duas filas devem ser implementados internamente no simulador através de uma única fila; para isto, a fórmula da função de prioridade virtual deve ser elaborada de forma que a prioridade seja dada para diferentes terminais como se estivessem em duas filas. Então, a solução encontrada foi utilizar a função de prioridade virtual como no esquema DQCA básico e somar o nível de prioridade ao resultado. Como o nível de prioridade de terminais de dados é 0 (zero) e o de terminais de voz é 1 (um), os terminais de voz sempre terão valores de prioridade virtual maiores que os terminais de dados. A fórmula elaborada é apresentada na Eq.(14):

$$PV = 1.0 / pTQ + nivelDePrioridade; \tag{14}$$

A.4.4.5. Classe EsquemaDQCA_Dif2.java

O esquema DQCA-DIF2 também utiliza duas filas, com prioridade para terminais de voz, entretanto, a diferença deste esquema com relação ao DQCA-DIF1 é que a fila de terminais de dados deve ser ordenada pelas taxas dos terminais. A solução empregada foi utilizar uma fórmula para cada tipo de terminal e somar a taxa máxima na prioridade dos terminais de voz, para torná-los prioritários sobre os de dados:

```
if(tipoTerminal == 'D'){

    PV = taxaDisp;

}else if(tipoTerminal == 'V'){

    PV = (1.0 / pTQ + taxaMaxima);

}
```

A.4.4.6. Classe EsquemaDQCA_RP.java

A classe EsquemaDQCA_RP implementa as modificações necessárias para a simulação do esquema DQCA-RP. Todos os esquemas apresentados nas seções anteriores exigiram modificações somente na função de prioridade virtual, onde é definida a disciplina de fila. Entretanto, no DQCA-RP os terminais de dados podem seguir uma disciplina de fila qualquer, mas conforme comentado no Capítulo 3, sempre que um terminal de voz tem pacote para transmitir, identificado pelo valor do *timer* menor que o instante atual, o terminal de dados adia a transmissão de um pacote, passando a vez para o terminal de voz. Este comportamento diferenciado exigiu modificações em outros métodos da classe, além da criação do campo estático, comentado a seguir:

terminalDeDadosDaVez. Este campo foi criado para armazenar o terminal de dados que está transmitindo e assim permitir que ele possa recuperar a transmissão depois de

passar a vez para um terminal de voz. Ele é mantido de acordo com os seguintes critérios: no início ele vale *null* (indefinido); ao entrar o primeiro terminal de dados, a lista de terminais é reordenada e o terminal na cabeça da fila será o terminalDeDadosDaVez. A partir daí, sempre que um terminal de dados transmitir seu último pacote, o terminalDeDadosDaVez recebe *null* e sempre que a lista for reordenada ele recebe o terminal da cabeça da fila.

Além da criação deste campo e das rotinas de atualização comentadas, foram realizadas modificações em outros métodos:

getPrioridadeVirtual: este método foi modificado para que, na classificação da lista de terminais primeiro apareçam os terminais de dados, com os terminais com melhores taxas primeiro, e depois os terminais de voz, ordenados do menor valor do campo *timer* ao maior. Então a fórmula que calcula a PV ficou como a seguir:

```
if(tipo == 'D'){   // Terminais de dados ordenados pelas taxas

PV = taxaDisp;

}else if(timer > 0){ // Terminais de voz pelos timer's

PV = (1 / (timer+1));

}
```

Foram implementadas modificações em outros métodos da classe, para inicializar o *timer* do terminal de voz quando ele ganha o acesso e atualizá-lo após a transmissão de cada pacote. Outras modificações foram realizadas para permitir que o terminal de dados na cabeça da fila passe a vez para os terminais de voz com *timer's* expirados e volte a transmitir em seguida.