GILSON NORBERTO HORSTMANN

AVALIAÇÃO DE MECANISMOS PARA GERENCIAMENTO DA FILA DA INTERFACE DO HOST CONTROLLER BLUETOOTH

Florianópolis
2002

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GILSON NORBERTO HORSTMANN

AVALIAÇÃO DE MECANISMOS PARA GERENCIAMENTO DA FILA DA INTERFACE DO HOST CONTROLLER BLUETOOTH

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Carlos Becker Westphall

Florianópolis, outubro de 2002

Avaliação de mecanismos para gerenciamento da fila da interface do Host Controller do Bluetooth

Gilson Norberto Horstmann

Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Ciência da Computação, Área de Concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Fernando A. Ostuni Gauthier
Coordenador do Curso

Banca Examinadora:

_____

Prof. Dr. Carlos Becker Westphall
Orientador

_____

Profa Dra. Carla Merkle Westphall

_____

Prof. Dr. Paulo José Freitas Filho

_____

Prof. Dr. Vitório Bruno Mazzola

À

Simone, Gabriela e Yuri

Agradecimentos

Pesquisar, simular, analisar, escrever e pensar, a rotina destes últimos meses. Ações que requerem determinação, disciplina, otimismo, perseverança e acima de tudo, fé. Fé no Deus do impossível, nAquele que sonda nossas intenções e que nos guia pelo Seu caminho. Este Deus maravilhoso, que sempre esteve presente, guiando-me por novos caminhos quando os conhecidos pareciam dar em lugar nenhum. O Deus que me mostrava as oportunidades, alimentava meus pensamentos, resgatava minha saúde física e espiritual. Agradeço ao meu Deus Maravilhoso, Pai Forte, Príncipe da Paz.

E nesta caminhada, na alegria incontida de ter alcançado o objetivo, as palavras de agradecimento que brotam do fundo do coração às pessoas que mais sentiram minha ausência, mas compreenderam, estimularam e me amaram, numa demonstração inequívoca de despreendimento e doação, minha amada esposa Simone Schreiber Horstmann, meus queridos e amados filhos, Gabriela Carolina e Yuri Alexandre.

Enio Alterman Blay, as nossas discussões dos mais diversos assuntos foram de extrema importância para o desenvolvimento do meu espírito crítico. Ficou visível a mútua admiração. Sou imensamente grato a você. E neste mesmo contexto, não poderia deixar de incluir Carlos Henrique Beuter, pelo estímulo, pelo tempo que dedicava para debater o trabalho, por auxiliar-me em momentos importantíssimos de consolidação de hipóteses e conceitos.

Foram fundamentais na jornada as pessoas com quem estudei ao longo deste período, ao quarteto que formamos, Guilherme Huth, Josué Michelutti e Sidney Vicente Mendes. Vocês são incomparáveis, meus amigos.

E para poder chegar ao fim desta aventura, agradeço ao meu orientador, Prof. Dr. Carlos Becker Westphall, pelo conhecimento e orientação para elaboração de todo o trabalho de pesquisa, simulação e apoio, e pela confiança e reconhecimento ao colocar-me em contato com pessoas maravilhosas, que jamais teria a oportunidade de conhecer sem esta indicação. E principalmente, por resgatar o sonho de prosseguir nos estudos.

Aos colegas da Embraco, em especial ao gestor de Tecnologia, Raul Moreira, por ter possibilitado de várias formas minha dedicação nos momentos mais importantes do trabalho; à minha equipe de trabalho, Beatris Xavier, Elaine C. Rocha, Carlos A. Hoepers, Monica E. Heinzelmann, Rodrigo A. Machado e Ronaldo Hornburg.

Por último, e mais importante, a meus pais Norberto e Geralda. Seus esforços para que os filhos estudassem, aprendessem a viver e tomassem suas decisões estão acima de qualquer descrição possível. Ensinaram-me a respeitar e amar as pessoas, a seguir em frente, a levantar tantas vezes quantos fossem os tropeços, a encarar as dificuldades com coragem e humildade e a viver sempre em função do próximo. Pai e mãe, eu os agradeço pela oportunidade de viver e pelo que me ensinaram com suas lutas.

Resumo

Este trabalho aborda o conceito de qualidade de serviços aplicada em dispositivos que utilizam a tecnologia Bluetooth. O Bluetooth é uma tecnologia de comunicação de rádio-freqüência, baixo consumo de energia, baixo alcance e alta capacidade de transferência de sinais.

Bluetooth forma redes chamadas *ad hoc*, onde o dispositivo *master* é o que inicia o processo de descobrimento de outros dispositivos sob sua área de cobertura de sinal. Os dispositivos que respondem ao *master* são chamados de *slaves*, e assim formam *piconets*. Quando há sobreposição de áreas de cobertura entre *piconets*, e há troca de sinalização entre elas, formam-se as *scatternets*.

Na pilha de protocolos do Bluetooth estão os protocolos L2CAP e *Baseband*, e entre eles está a interface do *Host Controller*. Através desta interface, o L2CAP envia comandos para o *Baseband* e, no sentido contrário, o *Baseband* envia eventos para o *host*.

Num dispositivo *slave* pode haver uma aplicação gerando tráfego melhor esforço em taxas superiores à outra aplicação gerando tráfego com requisitos de qualidade de serviços. Pela diferença das taxas de geração de tráfego, a fila do *Host Controller* pode ficar congestionada com tráfego melhor esforço.

Este problema pode ser extendido para *scatternets*, quando o tráfego de uma *piconet* pode gerar congestionamento sobre o dispositivo *gateway* que faz a conexão com outra *piconet*.

A fila do *Host Controller* implementa uma fila do tipo *First In First Out* (FIFO), que não faz o gerenciamento do conteúdo da fila. Para solucionar o problema, esquemas de fila alternativos podem ser implementados, como *Fair Queuing, Stochastic Fair Queuing e Deficit Round Robin*.

A contribuição deste trabalho está na realização de simulações com o software *Network Simulator* – versão 2. Os esquemas de fila citados foram simulados, e, o melhor desempenho foi obtido com o esquema *Deficit Round Robin*. Foram comparados os resultados de perda de pacotes, latência e *jitter*, considerando seis tamanhos de pacotes diferentes.

Abstract

The approach of this work is quality of services on devices with Bluetooth technology. Bluetooth is a radio frequency technology, with low power consumption, low range and high signal transmission capability.

Bluetooth form ad hoc networks, where the master unit starts the discovery service of other units in the area covered by its radio-frequency signal. All units in this area that answer to the discovery process are called slaves, and they form piconets.

When there are overlapping of covered areas between differents piconets and recognition of signals, the piconets form a scatternet.

The L2CAP and Baseband protocols are parts of the Bluetooth protocol stack. Among them is the Host Controller Interface that sends commands from the L2CAP to the Baseband, and, on the other way, the Baseband sends events to the host.

A slave unit can host one application generating best effort traffic in rates higher then another application that is generating traffic with quality of service requirements. Due the difference between the traffic generation rates, the queue on the Host Controller can become congested by best effort traffic.

This problem can be assigned also to scatternets, when the traffic of a piconet can put the gateway unit between the piconets in a congested state.

The queue of the Host Controller implements a First In First Out (FIFO) scheme. This scheme does not work on the content of the queue.

To solve this problem other queue shemes can be implemented, like Fair Queuing, Stochastic Fair Queuing and Deficit Round Robin.

The contribution of this work was the realization of simulations with the Network Simulation software, version 2. The queue schemes mentioned above were simulated and the best performance was obtained with the Deficit Round Robin scheme. The results were compared in terms of packet losses, latency and jitter, considering six differents packets sizes.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACL	-	Asynchronous Connection-Less
AODV	-	Adhoc On-demand Distance Vector
ARQ	-	Automatic Repeat Request
BE	-	Best Effort
BER	-	Bit Error Rate
BTCP	-	Bluetooth Topology Construction Protocol
CAC	-	Channel Access Code
CDMA	-	Code Division Multiple Access
CRC	-	Cyclic Redundancy Check
CVSD	-	Continuous Variable Slope Delta Modulation
DAC	-	Device Access Code
DRR	-	Deficit Round Robin
DSDV	-	Destination Sequence Distance Vector
DSR	-	Dynamic Source Routing
DSSS	-	Direct Sequence Spread Spectrum
ESS	-	Extended Service Set
ESSID	-	Extended Service Set Identification
FCC	-	Federal Communications Commission
FEC	-	Forward Error Correction
FHSS	-	Frequency Hopping Spread Spectrum
FIFO	-	First In First Out
FQ	-	Fair Queuing
GFSK	-	Gaussian Frequency Shift Keying
GHz	-	GigaHertz
HCI	-	Host Controller Interface
IAC	-	Inquiry Access Code
IBSS	-	Independent Basic Service Set
IEEE	-	Institute of Electrical and Electronics Engineers
IP	-	Internet Protocol
IPMoA	-	Integrated Personal Mobility Architecture
IR	-	InfraRed
ISA	-	Industry Standard Architecture
ISM	-	Industrial, Scientific and Medical
ISO	-	International Organization for Standardization
ITU	-	International Telecommunication Unit
kHz	-	kiloHertz
L2CAP	-	Logical Link Control and Adoption Protocol
LAN	-	Local Area Network
LC	-	Link Channel
LM	-	Link Manager
LMP	-	Link Manager Protocol
MAC	-	Media Access Control
MH	-	Mobile Host
MHz	-	MegaHertz
MPA	-	Mobile People Architecture
NIC	-	Network Interface Card
OSI	-	Open Source Initiative
PC	-	Personal Computer

PCI	-	Peripheral Component Interconnect
PCR	-	Prediction Confidence Ratio
PHY	-	Physical Level
PN	-	Pseudo Noise
QoS	-	Quality of Service
RF	-	Radio Frequency
RSVP	-	Resource ReSerVation Protocol
SCO	-	Synchronous Connection-Oriented
SDP	-	Service Discovery Protocol
SFQ	-	Stochastic Fair Queuing
STA	-	Station
TCP	-	Transport Control Protocol
TDD	-	Time Division Duplexing
TDMA	-	Time Division Multiple Access
TORA	-	Temporally Ordered Routing Algorithm
UA	-	User Assynchronous
UI	-	User Isosynchronous
US	-	User Synchronous
USA	-	User Services Assistant
UV	-	UltraViolet
WAN	-	Wide Area Network
WLAN	-	Wireless Local Area Network

1 - INTRODUÇÃO

Nos últimos anos, as redes sem fio *ad hoc* tem sido uma crescente área de pesquisas. A miniaturização dos dispositivos de computação móvel e o extraordinário crescimento da capacidade de processamento têm estimulado a utilização de equipamentos móveis.

Da mesma forma, a pesquisa e a produção de tecnologia para suportar a mobilidade computacional têm crescido. Comunicações que se estabelecem quando em contato com outro equipamento sem a necessidade de infra-estrutura fizeram surgir as redes *ad hoc*. Várias propostas surgiram abrangendo estabelecimento automático e manutenção dinâmica de topologias.

Surgem também novas aplicações visando dar mobilidade e comodidade ao ser humano. Atualmente, estas facilidades são obtidas através da conexão de computadores em casas, escolas, empresas, mas todas baseadas na utilização de cabos, infravermelho, meios que requerem determinados procedimentos muitas vezes dificultados pela ausência de padrões elétricos ou mecânicos no conjunto de tecnologias utilizadas. Individualmente apresentam grandes facilidades, mas quando inter-relacionadas apresentam desempenhos e comportamentos muitas vezes imprevisíveis.

Em 1994, a Ericsson iniciou um projeto procurando eliminar o incômodo causado pela quantidade de cabos existentes em praticamente todas instalações de computadores, necessários para ligar teclado, *mouse* e monitor, dentre outros periféricos. Com este objetivo inicial surgiu o Bluetooth.

Bluetooth designa um padrão de tecnologia que substitui os cabos de conexões de periféricos eletrônicos fixos ou portáteis por conexões de rádio de curto alcance. O padrão Bluetooth define uma estrutura uniforme para periféricos comunicarem entre si de forma robusta, com baixa complexidade, baixa potência e baixo custo. Esta tecnologia define também a comunicação com redes locais e com tecnologias de comunicação por voz, num conjunto que envolve equipamentos domésticos e interfaces de equipamentos de computação pessoal.

Alguns inconvenientes com a utilização de cabos podem ser relacionados:

- O emaranhado de cabos,
- Vários padrões de cabos e conectores,
- Conexões não confiáveis,
- Necessidade de manter cabos e conectores para reposição,
- Dificuldade na movimentação de unidades computadorizadas para diferentes locais devido à restrição dos tamanhos dos cabos,
- Necessidade de intervenção manual quando o número de portas físicas não é suficiente para novas conexões,
- Necessidade de reconfigurar as unidades no sistema operacional quando há movimentação das unidades.

Tecnologias como Bluetooth não significam a substituição do cabeamento por ondas de rádios. O cabeamento tem as suas vantagens, como a de estar menos suscetível a interferência de outras fontes de sinal, além de poder ser utilizado em ambientes com proteção contra ondas de rádio.

O objetivo maior é obter a especificação de uma tecnologia que otimiza o modo de uso de todos equipamentos de computação móvel e comunicação, provendo:

- Utilização global, transmissão de dados e voz,
- Capacidade de estabelecer conexões *ad hoc*,
- Capacidade de perceber a interferência de outras fontes na banda aberta,
- Tamanho reduzido para ser integrado em grande variedade de equipamentos,
- Baixo consumo de energia,
- Padrão aberto de interface,
- Baixo custo das unidades.

O Bluetooth não é uma tecnologia para altos volumes de transmissão, nem para substituição de cabeamento de redes locais (LAN) ou geográficas (WAN), nem de infraestruturas físicas centrais de redes (*Backbones*). A versão um do Bluetooth tem a velocidade nominal de 1 Mbps, e a versão dois prevê velocidade nominal de 11 Mbps. Redes locais e backbones requerem velocidades maiores do que as oferecidas pelo Bluetooth.

A tecnologia Bluetooth também não é adequada para aplicações baseadas em cliente-servidor. O foco do Bluetooth é mobilidade em unidades computadorizadas reconfiguráveis e que necessitam de conexões esporádicas. Aplicações cliente-servidor requerem maior estabilidade e conexões permanentes.

Bluetooth é uma tecnologia que possibilita a dispositivos portáteis formarem redes de curto alcance, sem fios, empregando saltos de freqüência no nível físico. As redes são formadas conforme a necessidade, sem a necessidade de uma infraestrutura de Internet (roteadores, *switches, hubs*, ...), denominadas redes *ad hoc*.

Estas características implicam em que os dispositivos não podem estabelecer canais de comunicação, a não ser que tenham se identificado previamente, sincronizando o padrão de suas freqüências. Praticamente todos dispositivos que estiverem na mesma área de cobertura podem comunicar-se. Porém, somente os dispositivos sincronizados com o transmissor podem ouvir a transmissão.

1.1 Objetivos

O *Host Controller Interface* (HCI) é uma interface padrão que pode ser usada por todos os módulos Bluetooth e por todos os dispositivos que incorporem a tecnologia Bluetooth.

O *Host Controller*, que incorpora o HCI é responsável pela interpretação dos dados recebidos e por direcioná-los ao(s) componente(s) apropriado(s) do Bluetooth.

Esta dissertação, visando agregar qualidade de serviços em dispositivos que utilizam tecnologia Bluetooth, tem como principais objetivos:

- Avaliar o problema de controle de fluxo no nível de *Host Controller* Interface com Baseband.
- Identificar alternativas para o problema de controle de fluxo.
- Estabelecer um modelo de simulação.

- Exercitar o modelo utilizando uma ferramenta apropriada para simulações de redes de comunicação.
- Analisar os resultados através de comparações com o modelo atual.
- Sugerir trabalhos futuros sobre o tema.

1.2 Justificativas

As pesquisas com redes *ad hoc*, onde está inserido o Bluetooth, procuram desenvolver soluções para superar desafios como mobilidade, inconstância de rotas e qualide de serviços (QoS).

Os desafios a serem analisados envolvem, principalmente, processos como:

- Controles de admissão, como parte de mecanismos para gerenciamento de recursos, envolvendo priorização de pacotes.
- Estabelecimento de rotas móveis, para comunicação entre dispositivos fora da área de cobertura em determinados instantes e para manutenção ou estabelecimento de novas rotas entre os dispositivos móveis;
- Mecanismos de controle de banda, atuando em conjunto com o controle de admissão, estabelecendo bandas fixas e pré-determinadas ou bandas variáveis conforme a demanda das aplicações.

1.3 Trabalhos Paralelos

Vários trabalhos foram consultados visando a formação de conceitos e verificação dos rumos de pesquisas. Alguns deles estão relacionados a seguir.

1.3.1 Redes *Ad hoc*

Há inúmeros trabalhos que suportam mobilidade pessoal na área de comunicação pessoal. Propostas como *Mobile People Architecture* (MPA), que proporciona aos usuários a capacidade de conexão sem restrições quanto ao dispositivo ou rede que estiver usando [35]. Um *proxy* é fornecido para cada usuário e toda comunicação acontece através deles, assim como a conversão de protocolos e adaptação de conteúdos. *Integrated Personal Mobility Architecture* (IPMoA), provê as mesmas funcionalidades do MPA de forma otimizada [35,36].

Nas redes *ad hoc*, assim como em redes de comunicação celulares, a grande dificuldade está relacionada com previsão da mobilidade. Chan *et al.* [37] propuseram

uma forma de melhorar a predição, aplicando um conceito chamado *Prediction Confidence Ratio* (PCR). Este conceito está baseado em diferentes valores usados estatisticamente pela rede fixa para fazer pré-configuração nas localidades adjacentes, pelas quais, provavelmente, o usuário móvel passará. Desta forma, procura evitar um padrão de predição de movimento aleatório.

Outra forma é exploração da não-aleatoriedade de comportamento, considerando os padrões de mobilidade que um usuário móvel apresenta. Os defensores desta técnica, Su *et al.* [38], afirmam que o estado futuro de uma topologia de rede pode ser previsto, consequentemente a reconstrução proativa de rotas pode ser executada. Esta previsibilidade melhora o desempenho de protocolos de roteamento.

No sentido oposto, Ardon *et al.* [39] consideraram um esquema reativo para reserva de recursos. A proposta é estruturada sobre um esquema de agentes de QoS reativos, chamado de *User Services Assistant* (USA). O principal requisito para disponibilizar este serviço é chamado de *Dynamic Resource-Reservation Protocol*, o qual reserva recursos ao nível de rede através de um esquema *sender-initiated*.

Um esquema reativo de reserva de recursos supera alguns problemas associados com o gerenciamento de qualidade de serviço (QoS), que tem como base o mapeamento de parâmetros de qualidade para sistemas relacionados e a pré-determinação da rota ótima da relação degradação/melhoria (*degrading/upgrading*) de qualidade de serviço (QoS).

As comparações entre quatro protocolos que gerenciam múltiplos saltos no roteamento em redes *ad hoc*, *Destination Sequence Distance Vector* (DSDV), *Temporally Ordered Routing Algorithm* (TORA), *Dynamic Source Routing* (DSR) e *Ad hoc On-demand Distance Vector* (AODV), foram realizadas por Broch *et al.* [40], usando o simulador ns-2.

Este trabalho identifica maior previsibilidade no protocolo DSDV, que liberou, virtualmente, todos pacotes em ambiente com baixa taxa de mobilidade e baixa velocidade de movimento. TORA apresentou o pior desempenho. DSR apresentou bom desempenho, embora tenha aumentado a sobrecarga de trabalho de roteamento pelo uso do protocolo. Finalmente, AODV, que apresentou um desempenho semelhante ao DSR, embora requeira a transmissão de pacotes de roteamento, tornando-o mais caro que o DSR.

Informações adicionais sobre protocolos de roteamento para redes *ad hoc* podem ser encontrados no livro "*Ad hoc Networking*", de Charles Perkins [03]. O objetivo do livro é esclarecer aos leitores os diversos algoritmos e protocolos do estado da arte em redes *ad hoc*.

1.3.2 Bluetooth

Há um interesse crescente na tecnologia Bluetooth, o que pode ser constatado pelo aumento do número de trabalhos com datas recentes. Entre serviços de descoberta de dispositivos na área de cobertura (*Service Discovery*), protocolos de rádio, mecanismos de reserva de *polling*, há também vários trabalhos voltados para a formação de redes Bluetooth, chamadas de *scatternets*.

Pravin Bhagwat e Adrian Segall [41] exploraram a questão de protocolos de roteamento sobre *scatternets*, fazendo analogias com os trabalhos desenvolvidos pelo grupo de trabalho de redes móveis (Manet). Embora não tenham implementado simulações e experimentos, descreveram com muitos detalhes e explicaram o funcionamento teórico do protocolo de roteamento proposto.

Comunicação entre *piconets*, dispositivos Bluetooth conectados, implica na formação de *scatternets*. Uma *piconet* é formada por um dispositivo controlador (*master*) e dispositivos controlados (*slaves*).

Podem ser formadas *piconets* contendo até 255 estações. Para isto, a estratégia é de ativar o *slave* mais antigo após a desativação do *slave* mais antigo. Outras estratégias analisadas incluem a formação de *scatternets* hierárquicas e a utilização de um slave como gateway entre *piconets*. Os desempenhos das estratégias de comunicação citadas, analisando os atrasos médios, foram avaliados por Kalia *et al.* [42]. O estudo está incompleto devido à omissão de detalhes do processo de modelagem, e por não citar a ferramenta utilizada para avaliar os tempos

Um trabalho esclarecedor, desenvolvido por Miklos *et al* [43], avaliou as implicações de desempenho na formação de *scatternets* a partir das *piconets*. Primeiro estabeleceu um modelo de rede e definiu métricas de desempenho para *scatternets*. As métricas foram definidas a partir das restrições da tecnologia Bluetooth, mas abstraídas a ponto de servirem para uma análise mais genérica de redes *ad hoc*. No segundo passo, ao realizar as simulações e relacionar os parâmetros das *scatternets* com as métricas de desempenho identificou correlações entre as regras de formação de *scatternets* e desempenho.

Seguindo a linha de análise das propriedades e restrições da tecnologia Bluetooth, Law *et al* [44] propuseram um protocolo novo, randômico e distribuído, para formação de *scatternets*, incorporando o problema de descobrimento de dispositivos para múltiplas conexões em paralelo com vários dispositivos *master* e *slaves*. O desempenho do algoritmo proposto foi avaliado através de simulações [22].

Utilizando um protocolo distribuído assíncrono, chamado de Bluetooth *Topology Construction Protocol* (BTCP), uma *scatternet* pode ser formada em duas fases: a primeira com a seleção de um líder que conheça todo os demais nós e a segunda fase, quando o líder determina aos demais nós como será formada a *scatternet* [45]. Basicamente, o protocolo inicia com nós que não conhecem os demais nós na área de cobertura e termina com a formação de uma rede conectada atendendo todas as restrições de conectividade impostas pela tecnologia Bluetooth.

Na mesma linha, relaciona-se o trabalho de Aggarwal *et al.* [46], que também apresentaram um algoritmo para formação de *scatternets*, dividindo inicialmente a rede em *piconets* independentes e elegendo um "super-*master*" que conhece todos os nós da rede. Em seguida ocorre uma reorganização de todas as *piconets* formadas e ainda não interconectadas.

1.4 Motivações

A premissa básica de suporte à este trabalho está nas intensas mudanças em computação e equipamentos de redes. Equipamentos móveis, de baixo custo, com larguras de bandas apropriadas, difundidas pelo mundo com acessos fáceis às redes com fios.

O desenvolvimento deste trabalho tem como principais motivações:

- Desenvolvimento acelerado de tecnologias de redes *ad hoc*, estimuladas pela crescente necessidade de estar conectado em meios para troca de informações;
- Uso intensivo de recursos multimídia, como forma de apresentar informações que tem exigido qualidade de serviço nos meios de transmissão de dados;
- A necessidade de identificar parâmetros para assegurar qualidade de serviços IP – QoS – em redes *ad hoc* como tecnologia Bluetooth.

- A constatação feita através das leituras e análises dos trabalhos correlatos sobre a necessidade de mecanismos apropriados para gerenciamento das cargas nos dispositivos *slaves* de *piconets* no Bluetooth.

1.5 Organização do trabalho

O trabalho procura inicialmente conceituar redes *ad hoc*. Estabelece comparações de topologias com outras infra-estruturas de rede sem fio e faz uma introdução ao conceito de rádio-freqüência e de tecnologias do espectro magnético.

São apresentados os conceitos de *Direct Sequence Spread Spectrum* (DSSS) e *Freqüency Hopping Spread Spectrum* (FHSS), sendo esta utilizada pelo Bluetooth, que também é o assunto seguinte.

Sobre o Bluetooth, são apresentados os princípios desta tecnologia, características físicas, pacotes e canais de comunicação. Pontos importantes para assegurar o bom entendimento do trabalho, a pilha de protocolos, o estabelecimento de conexões e os tipos de redes *piconets* e *scatternets* são descritos e quando possível, enriquecidos com figuras.

Os conceitos de qualidade de serviços – QoS – em redes também são apresentados. Os principais requisitos de uma rede com qualidade de serviços, os mecanismos de tratamento de tráfego e as categorias de mecanismos são esclarecidos. A ênfase maior está sobre esquemas de filas.

Um capítulo especial é dedicado à questão de qualidade de serviços na tecnologia Bluetooth. Os principais problemas estão relacionados neste capítulo seguinte. Os problemas apresentados estão relacionados com QoS e os níveis de protocolos envolvidos com o aspecto de qualidade. O problema do controle de fluxo é tratado com maior nível de detalhamento.

Na sequência são descritos os experimentos e resultados. O simulador ns-2 é apresentado de forma suscinta, e prossegue-se com a descrição do modelo e os resultados observados nas simulações. Os resultados são comentados sob os diferentes aspectos observados, perda de pacotes, latência e *jitter* em cada esquema de fila. Especificamente para o esquema FIFO são apresentadas duas variações nos tamanhos máximos de fila.

A conclusão é o último capítulo do trabalho, sendo seguido pelos anexos contendo gráficos e tabelas resultantes das várias simulações realizadas. Os gráficos demonstram

o nível de enfileiramento e perda de pacotes por cada esquema de fila, a latência e o *jitter*. Além disso, os comportamentos de cada tipo de pacote nos diferentes esquemas de filas também são comparados. As tabelas dos anexos mostram os valores finais das simulações, tanto na *piconet* quanto na scatternet.

2. REDES SEM FIOS - *WIRELESS*

2.1 *Introdução*

As redes sem fio surgiram e adquiriram grande popularidade nos últimos tempos. Na medida em que a miniaturização dos dispositivos utilizados em computação móvel tem se desenvolvido, em que o poder de processamento tem aumentado, mais acessíveis se tornaram equipamentos de comunicação móvel.

Em paralelo, a Internet também tem estimulado a necessidade de estar sempre conectado a rede. A possibilidade de receber informações novas e de forma constante, uma certa ansiedade de informações, em qualquer lugar, tornou-se um sinônimo de competitividade individual e organizacional.

Executivos, vendedores, consultores tornaram-se os grandes usuários dos meios de comunicação móvel. Decisões mais rápidas, acesso mais rápido à informação e mais viagens do que em anos anteriores demandou maior acesso às informações.

O desenvolvimento e a massificação da comunicação celular estimularam o consumo de produtos de comunicação móvel. A figura 2.1 apresenta os números esperados de usuários da comunicação móvel (celular), que devem convergir para os mesmos níveis da comunicação fixa em 2004 [01].

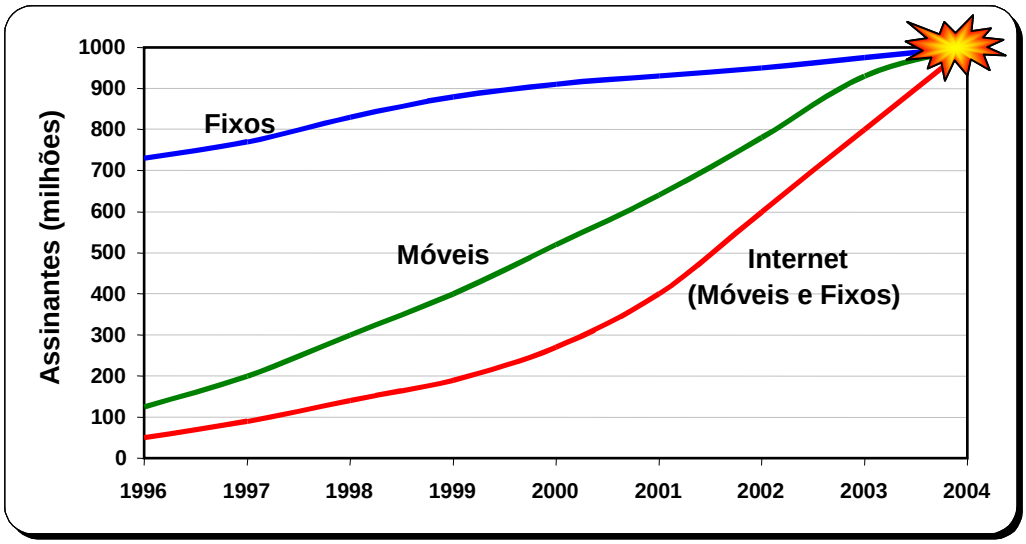

Figura 2.1 – Crescimento estimado de assinantes

Os benefícios das comunicações móveis - sem fio, podem ser extrapolados para lugares em que é necessária a comunicação e não se dispõe de infra-estrutura, a situações de emergência, catástrofes e campos de batalhas, onde a necessidade de comunicar-se é fundamental para o sucesso das operações e coordenação de esforços.

Neste capítulo, investigaremos o funcionamento de redes sem fio, seus componentes e as configurações possíveis de acordo com as áreas de cobertura. Também são descritas tecnologias do espectro magnético, especificamente as técnicas DSSS e FHSS.

2.2 *Funcionamento de redes sem fio*

O propósito de redes sem fio em organizações é estender a cobertura da rede para permitir, ao nível de prédios, construções e campus a comunicação de usuários móveis.

Há quatro componentes numa rede sem fio:

- Pontos de acesso (PA)
- *Wireless LAN Network Interface Card* (NIC)
- *Wired* LAN
- Equipamento móvel / computador de mesa (PC *Desktop*)

Os pontos de acesso (PA) convertem o sinal captado do ar em dado a ser transmitido pelos cabos (pode ser Ethernet ou Token Ring). Atua como uma ponte entre os usuários móveis e a rede física (formada por cabos). O ponto de acesso recebe,

armazena e transmite dados entre a rede sem fio e a rede física. Cada cliente tem um endereço MAC que é reconhecido pelo ponto de acesso (figura 2.2) [01].

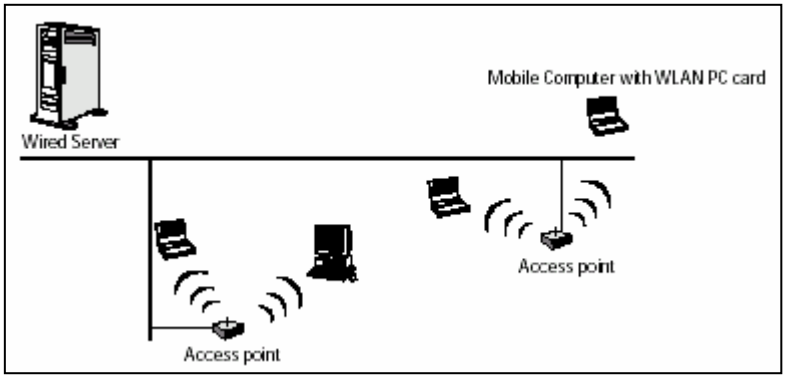

Figura 2.2 – Configuração típica de uma WLAN

As ondas de rádio são as portadoras, pois elas simplesmente executam a função de liberar energia para um receptor remoto. O dado a ser transmitido é modulado sobre a portadora de tal forma que possa ser extraído pelo receptor.

Utilizando diferentes freqüências de rádio, múltiplas ondas podem existir no mesmo espaço ao mesmo tempo sem que uma interfira em outra. As estações receptoras precisam estar sintonizadas na freqüência de rádio apropriada para capturar o sinal enquanto rejeita outras freqüências.

Os pontos de acesso são definidos pelo *Extended Service Set Identification* (ESSID), parte do padrão IEEE 802.11, com o qual os clientes são configurados para manter a comunicação. Quando há múltiplos pontos de acesso na mesma sub-rede então devem ter o mesmo ESSID e permitir o roaming. Ponto de acesso de servidor com o mesmo ESSID formam um *Extended Service Set* (ESS) [02].

Através de adaptadores ISA ou PCI em computadores ou handhelds ou através de LAN NICs em computadores móveis (*notebooks*, *laptops*), usuários tem acesso à rede. Usando ondas eletromagnéticas, rádio ou infravermelho, os dados são transmitidos aos pontos de acesso sem a utilização de conexões físicas entre eles.

Os pontos de acesso suportam pequenos grupos de usuários por área de cobertura, sendo que a maioria pode suportar simultaneamente de 60 a 70 usuários.

Os pontos de acesso:

- Quanto maior o número de pontos de acesso, melhor desempenho,
- Quanto maior o nível de encriptação, menor desempenho,
- Quanto mais alto instalado, melhor.

Se uma rede sem fios não tiver nenhuma conexão com redes físicas, os usuários desta rede somente poderão estabelecer comunicação entre si. Este tipo de rede é altamente segura e portátil, podendo ser utilizada em diferentes pontos sem necessidade de instalação de infra-estrutura física.

2.3 Configurações de redes sem fios (wireless)

As redes sem fios podem ser configuradas de acordo com a área de cobertura alcançada [01].

2.3.1 Redes independentes

Computadores podem ser interligados entre si usando adaptadores sem fios. Quando dois ou mais computadores com adaptadores wireless estiverem na mesma área de abrangência poderão formar uma rede independente. É a forma mais simples de configuração de uma rede, ponto-a-ponto, sem necessidades de administração ou pré-configuração (figura 2.3 [01]).

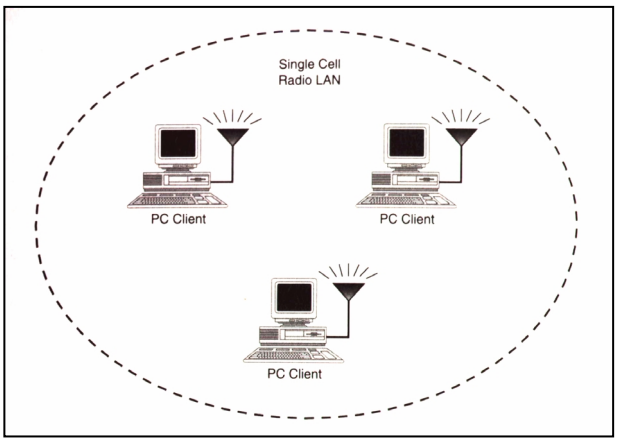

Figura 2.3 – WLAN Independente

2.3.2 Redes com infra-estrutura

Redes com infra-estrutura fazem a conexão entre os usuários móveis e as redes com fios. Usuários móveis podem acessar os recursos da rede através dos vários pontos de acesso instalados no prédio ou campus.

Os pontos de acesso servem também como mediadores entre os vários pontos de acesso instalados próximos, mediando o processo de comunicação sem fios (figura 2.4 [01]).

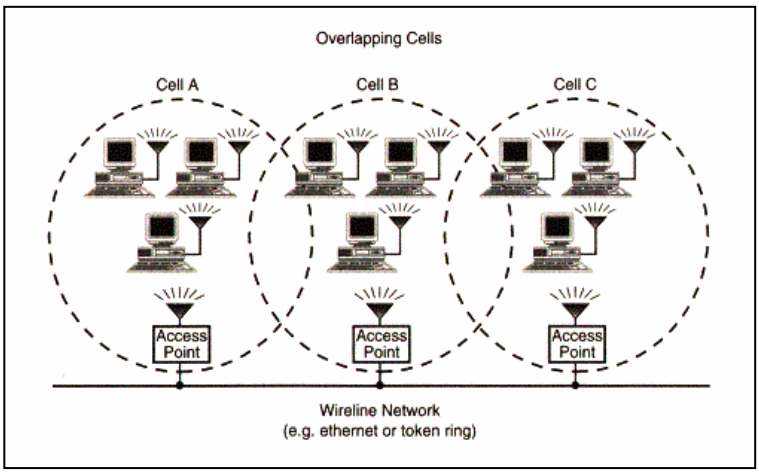

Figura 2.4 – WLAN com infra-estrutura

2.3.3 Microcélulas e roaming

A rede de telefonia celular usa o conceito de células para estender sua área de cobertura. De forma similar WLAN usam microcélulas para estender sua conectividade. A área de cobertura é definida pelo alcance do sinal para determinada potência. Um usuário móvel estará conectado com um ponto de acesso, e enquanto vai se deslocando de uma área para outra mantém sua conexão com a rede (figura 2.5 [01]).

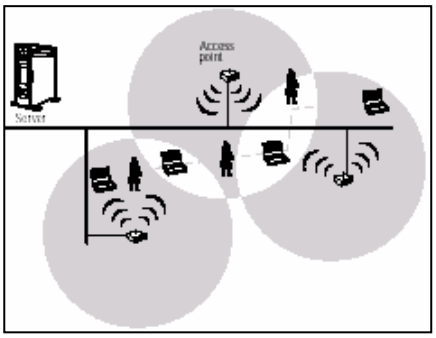

Figura 2.5 – Manutenção da conexão entre pontos de acesso

2.4 Redes ad hoc

Redes *ad hoc* são formadas conforme a necessidade e oportunidade, sem a necessidade da disponibilidade de infra-estrutura. Algumas técnicas foram propostas em relação a estas redes. Perkins[1], *apud* [Bagrodia+1996] cita técnicas para redes *ad hoc* com infra-estruturas instantâneas e sobre redes "*mobile-mesh*", *apud* [SDT 1995].

[1] Ver Bibliografia [03].

A visão de redes móveis *ad hoc* é suportar operações robustas e eficientes em redes móveis sem fio incorporando funcionalidades de roteamento nos nós móveis. Estas redes são vistas como sendo dinâmicas, mudando rapidamente, randômicas, com topologias multihop e compostas por restrições de banda em canais sem fio [04,05]. Na comunidade Internet o suporte para roteamento de dispositivos móveis é formulado como tecnologia IP móvel, suportando "*roaming*".

A tecnologia de redes móveis *ad hoc* é praticamente um sinônimo de redes móveis de pacotes de rádios, "*mobile-mesh*", e também de redes "*mobile, multihop and wireless*" [04].

A topologia básica (IBSS – *Independent Basic Service Set*) de uma rede *ad hoc* consiste de dois ou mais nós, ou estações (STA), que estando na mesma área de cobertura de sinal, podem se reconhecer, estabelecer uma comunicação, formando-se espontaneamente (figura 2.6). Os *hosts* (MH – *Mobile Hosts*) podem estar em movimento e conectando-se com diferentes *host*s ao longo do caminho.

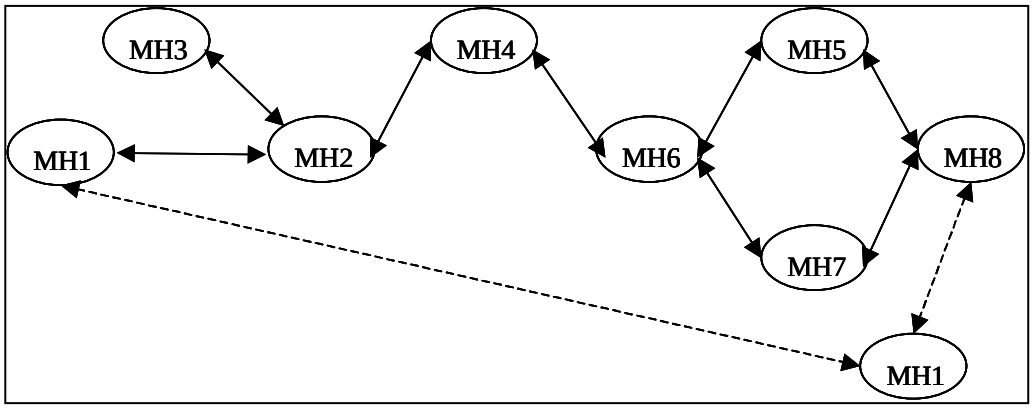

Figura 2.6 – Rede *Ad hoc*

2.5 Rádio-freqüência

Rádio freqüência é o campo eletromagnético, que também pode ser chamado de onda de rádio, gerado quando uma corrente alternada é enviada para uma antena. Este campo pode ser usado para comunicações sobre partes do espectro de radiação eletromagnético, de 9 kilohertz (kHz) para milhares de gigahertz (GHz). A porção usada é chamada de espectro de rádio freqüência (RF *Spectrum*), e a energia eletromagnética pode ter a forma de infravermelho (IR), luz visível, ultravioleta (UV), raios X e raios gama.

O espectro de rádio freqüência é dividido em vários intervalos, onde cada banda representa um incremento de freqüência correspondente a uma ordem de magnitude (potência de dez). A figura 2.7 [06] ilustra oito bandas do espectro de rádio freqüência e o intervalo destas bandas.

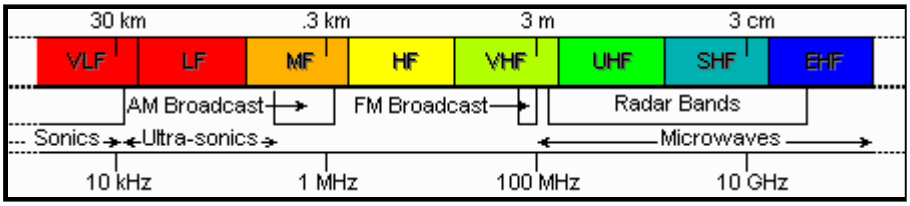

Figura 2.7 – Espectro de rádio freqüência

O ITU, International Telecommunication Union, é responsável pela alocação de bandas para várias classes de serviços de acordo com as diferentes regiões do mundo.

2.6 Tecnologias do espectro magnético

Existem três técnicas de espectro espalhado: seqüência direta (*direct sequence*), salto de freqüência (*frequency hopping*), salto de tempo (*time hopping*). O uso combinado destas tecnologias também é empregado.

Os métodos da tecnologia de espectro espalhado (*spread spectrum*) mais usados são seqüência direta (*direct sequence*) e salto de freqüência (*frequency hopping*). Embora tenham o mesmo objetivo, apresentam desempenhos diferentes pelas características distintas que apresentam.

A maioria das redes sem fio utiliza uma destas tecnologias para transporte das ondas de rádio. Foi desenvolvida inicialmente para fins militares e utilizada em sistemas de comunicação de missão crítica por sua confiabilidade, segurança, resistente ao congestionamento e com comunicação de missão crítica por ser confiável e segura, resistente ao congestionamento e difícil de ser interceptada por dispositivos não autorizados [07, 08].

A tecnologia de espectro espalhado implementa segurança e integridade em detrimento da eficiência de alocação de banda. Isto significa que mais banda é utilizada para transmissões ao mesmo tempo em que o sinal é mais alto e mais fácil de ser detectado.

Os sinais são indistinguíveis e praticamente transparentes, distribuídos sobre um amplo intervalo de freqüências e depois coletados pelo receptor na sua freqüência

original. Por serem praticamente não interceptáveis pelos receptores não desejados, não interferem com outros sinais, mesmo que transmitidos nas mesmas freqüências.

Para que um sinal seja qualificado como sinal de espectro espalhado deve atender a dois critérios: [08]

1. A sinalização da banda deve ser muito maior que a banda da informação,
2. Alguma função diferente da informação que está sendo transmitida é empregada para determinar o resultado transmitido por toda a banda.

Modulação de sinal é o processo de sobrepor uma onda de baixa freqüência a uma onda de maior freqüência que é fixa e constante, chamada de portadora.

A sinalização é modulada em banda maior do que o sinal modulado da informação transmitida, gerando a impressão de ruído. A maioria dos sistemas de espectro espalhado transmite sinal de rádio freqüência em bandas 20 a 254 vezes maiores do que a banda em que a informação está sendo enviada. Alguns sistemas empregam bandas de rádio 1000 vezes superior à banda da informação transmitida.

A tecnologia de espectro espalhado permite que vários usuários compartilhem a mesma freqüência de banda, principalmente quando acompanhada pela tecnologia de acesso CDMA, que permite a transmissão simultânea de sinais na mesma freqüência. Funciona alocando para cada usuário uma seqüência única para identificação pelo receptor. A técnica de multiplexação TDMA também pode ser usada, embora seja menos empregada.

Há duas abordagens que podem ser empregadas na tecnologia de espectro espalhado, ambas aprovadas para uso pelo FCC na banda não licenciada ISM [02]:

1. Espectro espalhado por salto de freqüência – FHSS,
2. Espectro espalhado por seqüência direta – DSSS.

O comitê 802.11 WLAN ao estabelecer o padrão global para WLANs optou por deixar o usuário decidir qual das abordagens é mais adequada as suas necessidades [06]. Ambas opções do nível PHY foram especificadas em conformidade com as regulamentações do FCC (FCC 15.247).

2.6.1 DSSS – Direct Sequence Spread Spectrum

DSSS “espalha” o sinal sobre a larga banda de rádio freqüência e cria um padrão redundante chamado “chip” para transmitir. Este “chip” é o tempo para transmitir um

bit ou simples símbolo de um código PN (*Pseudo Noise*), um sinal digital com propriedades de ruído.

A portadora do rádio de seqüência direta permance em freqüência fixa. As informações em *narrowband* são espalhadas sobre uma banda muito maior (pelo menos 10 vezes) usando uma seqüência chamada "*pseudo-random chip*". Na figura 2.8 [08], DSSS *Spreading*, está ilustrado o sinal *narrowband* e o sinal de espectro espalhado, ambos usando a mesma quantidade de potência para transmissão das mesmas informações. A diferença está na densidade do sinal, muito menor do que no sinal narrowband, conseqüentemente muito mais difícil de detectar sua presença.

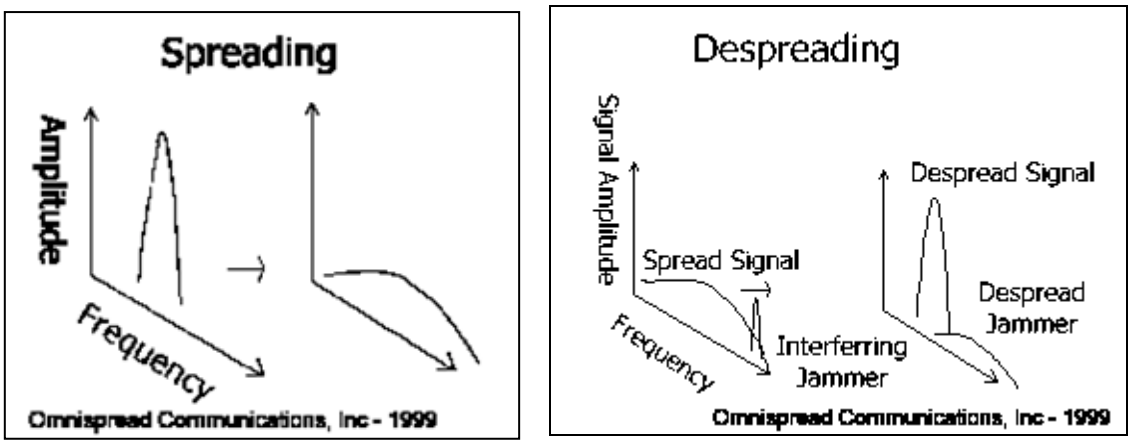

Figura 2.8 – DSSS *Spreading* e DSSS *Despreading*

O receptor agrupa o sinal recebido para regerar o sinal *narrowband* original – figura 2.8 (DSSS *Despreading*). As interferências na mesma banda serão eliminadas no agrupamento, resultando num sinal limpo.

Todos rádios operando em 11 Mbps são DSSS.

2.6.2 *FHSS – Frequency Hopping Spread Spectrum*

O mesmo resultado do DSSS é obtido pelo FHSS mas usando portadoras em diferentes freqüências em diferentes momentos. A portadora salta em freqüências diferentes evitando a interferência em determinadas freqüências. Esta é a única técnica realmente viável na freqüência de 2.45 GHz em função da interferência de fornos de microondas (2.4 a 2.5 GHz).

A figura 2.9 [09] representa a sinalização FHSS. Nela está representada uma sequência de transmissão C,A,B,E,D. Na abscissa (eixo x), estão representadas as

frequências e na ordenada (eixo y) está representado o tempo. Verificam-se então as diferentes frequências usadas ao longo do tempo.

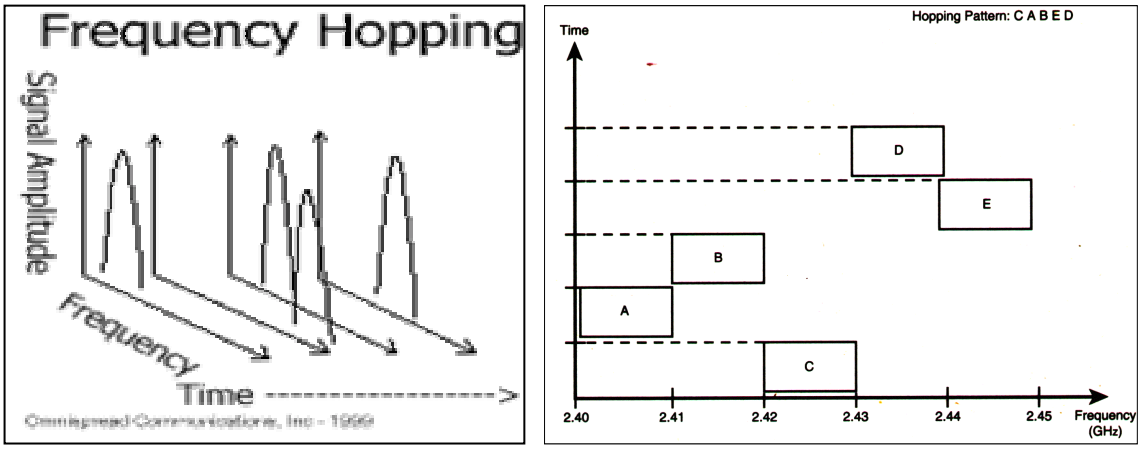

Figura 2.9 – FHSS – *Frequency Hopping Spread Spectrum.*

2.6.3 Comparando DSSS e FHSS

Embora tenham os mesmos objetivos, ambos métodos têm características distintas. A tabela 2.1 [09] compara algumas características das tecnologias e a tabela 2.2 [10] caracteriza as tecnologias. Ambas são tecnologias físicas e não são interoperáveis.

DSSS	Vs.	FHSS
Fácil e simples	vs.	Complicada
Baixa potência	vs.	Alta potência
Curto período de latência	vs.	Alto período de latência
Conexão de entrada rápida	vs.	Conexão de entrada lenta
Curto alcance interno	vs.	Alto alcance interno
Baixas taxa de transferência	vs.	Altas taxas de transferências.

Tabela 2.1 – Comparação entre DSSS e FHSS

CARACTERÍSTICAS	DSSS	FHSS
Banda	2.4 GHz	2.4 GHz
Padrão	IEEE 802.11b	IEEE 802.11
Técnica de modulação	Modulação de amplitude (AM)	Modulação de freqüência (FM)
Canal da portadora	Fixado em canal de 17 MHz	Envia dados sobre canais 1 MHz
Serviços suportados	Dado	Dado, vídeo e voz.
Máx. de canais independentes	3	15
Tecnologia da indústria	802.11b	HomeRF, Bluetooth

Tabela 2.2 – Características das tecnologias DSSS e FHSS

2.7 Conclusões do capítulo

Novas tecnologias têm surgido e estão sendo incorporadas nos equipamentos de comunicação móvel. As redes adequaram diferentes alternativas de infra-estrutura, cada vez com menores áreas de cobertura.

O espectro magnético, dividido em vários intervalos, é utilizado através de três métodos: seqüência direta, salto de freqüência e salto de tempo. Técnicas de sinalização foram desenvolvidas sobre espectro espalhado.

Este capítulo apresentou os conceitos básicos de redes sem fio, comentando as infra-estruturas possíveis baseadas em áreas de cobertura e sinalizações, conforme especificações do comitê IEEE.

Em seguida detalhou-se mais o espectro magnético, especificamente as técnicas de sinalização FHSS e DSSS, utilizadas em redes *ad hoc*.

A continuidade do trabalho abordará o Bluetooth, uma tecnologia para redes *ad hoc*, sem fios, empregando sinalização FHSS.

3. BLUETOOTH

3.1 *Introdução*

Neste capítulo investigaremos a tecnologia de redes sem fio Bluetooth. O grande desafio do Bluetooth é fazer uma conexão entre equipamentos de comunicação de forma simples e eficiente. Dispositivos Bluetooth formam redes quando entram em áreas de cobertura, permitindo que se estabeleça um canal de comunicação entre si, fornecendo mobilidade, não respeitando paredes, além de evitar a obstrução no sinal. O Bluetooth possui uma interface de rádio-freqüência de baixa potência, operando na faixa de 2,4 GHz, utilizando a técnica de sinalização FHSS, tendo como principal vantagem a baixa interferência entre as mensagens de diversas fontes. Dois tipos de canais podem ser estabelecidos, ACL e SCO, nos quais são enviados pacotes de dados e controles. Mecanismos de controle de erros e verificação da integridade da informação são incorporadas na estrutura da tecnologia Bluetooth. O sistema foi segmentado em níveis independentes permitindo melhor descrição conceitual, formando a pilha de protocolos da tecnologia Bluetooth. Conexões podem ser estabelecidas formando *piconets* ou *scatternets*, dependendo dos dispositivos na mesma área de cobertura.

3.2 História

Bluetooth teve origem em 1994 quando a Ericsson autorizou a execução de um estudo para investigar alternativas para substituir os cabos tradicionalmente usados para conectar telefones celulares a *headsets* e outros dispositivos. Os engenheiros alocados

neste projeto foram o sueco Dr. Sven Mattison e o holandês Dr. Jaap Haartsen. Inicialmente o projeto era chamado de *Multi-Communicator Link* – MC Link.

Quando os engenheiros descobriram que poderiam usar uma banda de rádio de baixa freqüência que não exigia licenciamento, portanto, disponível para ser usada sem nenhum custo, o projeto levou ao desenvolvimento de um pequeno rádio embutido em um chip de computador. Isto levou à identificação de outras aplicações que poderiam utilizar esta tecnologia.

Em 1997 a Ericsson decidiu doar a tecnologia. Sabendo que produtos baseados em tecnologias proprietárias raramente são bem sucedidos, a Ericsson conversou com outros fabricantes e estimulou-os a participar no desenvolvimento mais rápido desta tecnologia.

Em 20 de maio de 1998, a Ericsson, IBM, Intel, Nokia e Toshiba anunciaram para a imprensa em Londres – Inglaterra, São José – Califórnia, e Tóquio – Japão simultâneamente, que haviam se juntado para desenvolver uma especificação aberta, sem *royalties* para a conectividade sem fio entre os dispositivos de computação e os de telecomunicações.

Figura 3.1 – Logotipo Bluetooth

Neste dia a especificação foi nomeada com o código Bluetooth e a organização que daria suporte à especificação foi chamada de *Bluetooth Special Interest Group* (SIG).

Em dezembro de 1999 as empresas 3Com, Lucent Technologies, Microsoft e Motorola se juntaram como membros patrocinadores, e em seguida novas empresas se juntaram ao SIG ao nível de associada ou adotante.

O nome Bluetooth foi proposto pelo americano Jim Kardach da Intel, aficionado pelo estudo de história, referindo-se ao rei da Dinamarca Harald Blaatand “Bluetooth” II. Nascido no ano 911 A.D., governou a Dinamarca de 940 a 980 A.D., falecendo em 985 A.D.

O logotipo da Bluetooth (figura 3.1) é formado pelos caracteres rúnicos H e B, de Harald Bluetooth. A realização mais significativa do Rei Harald foi unificar e cristianizar a Dinamarca, além de ter conquistado a Noruega [11].

3.3 Características Físicas

3.3.1 Especificação de rádio

O esquema de modulação usado é *Gaussian Frequency Shift Keying* (GFSK), também usado em sistemas GSM. O número um representa um desvio de freqüência positivo e o zero para um desvio de freqüência negativo do canal de freqüência do dispositivo de RF. As vantagens do GFSK sobre outros esquemas de modulação são [12, 13]:

- Alta eficiência do amplificador de rádio freqüência;
- Espectro de potência menor, minimizando interferência em canais adjacentes;
- Bom desempenho da taxa *Bit Error* (BER).

O filtro BT é definido em 0,5 e o índice de modulação deve estar entre 0.28 e 0.35 [14]. As características de transmissão podem ser vistas na tabela 3.1 [14], mostrando as classificações de cada equipamento Bluetooth.

Power Class 1	Equipamentos de longa distância (~100m) Antena: potência máxima de saída de 20 dBm.
Power Class 2	Equipamentos de distâncias intermediárias (~10m) Antena: potência máxima de saída de 4 dBm.
Power Class 3	Equipamentos de distâncias curtas (~10cm) Antena: potência máxima de saída de 0 dBm.

Tabela 3.1 – Classes de potência de transmissão

3.3.2 Especificação canal Bluetooth

O termo canal pode ter três significados no contexto da tecnologia Bluetooth [15, 16]:

1. Referência aos 79 ou 23 canais de rádio-freqüência, com freqüências individuais de 1 MHz cada, operando na banda de 2.4 GHz Industrial, Científica e Médica (ISM) – (Figura 3.2 [05]). Cada canal é dividido em intervalos de tempo de 625 μs, o que equivale a 1.600 saltos por segundo. Esta freqüência é alterada em cada novo intervalo de tempo, exceto quando há pacotes de vários intervalos (*multi-slots*). Por exemplo, um simples

canal Bluetooth pode usar os canais de rádio freqüência 3, 74, 57, 64, 67..., com a freqüência mudando ao início de cada intervalo de tempo.

2. Também uma referência aos canais de comunicação, porém, consistindo de uma seqüência de saltos pseudo-randômica destes 79 (ou 23) canais de rádio-freqüência (equivale a uma “sessão” no modelo OSI);
3. E finalmente, pode estar se referindo aos cinco canais lógicos utilizados para controle.

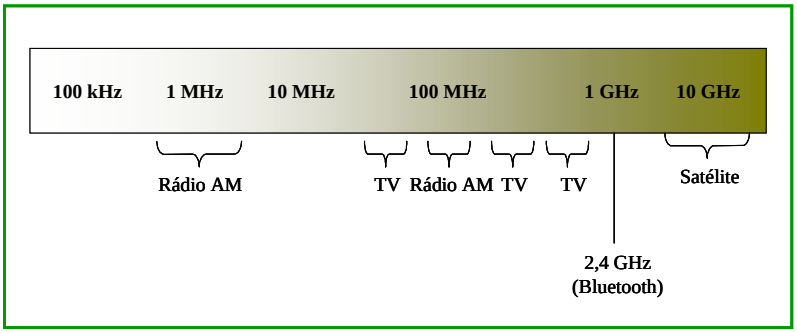

Figura 3.2 – Proximidades da banda 2,4 GHz

Por ser uma banda livre para qualquer uso, a interferência de freqüência em algum momento pode deixar a banda congestionada. No Bluetooth, a comunicação é feita usando espalhamento de freqüência (*spread spectrum*), e para otimizar o espalhamento, toda a banda de 2400 MHz a 2483.5 MHz é utilizada.

Saltos rápidos e pacotes pequenos de dados são usados para minimizar a possibilidade de interferências. Os cabeçalhos dos pacotes são protegidos com um esquema de alta redundância para correção de erros, e para transmissão de voz é adotada codificação *Continuous Variable Slope Delta Modulation* (CVSD)[2].

O rádio Bluetooth faz 79 saltos de 1 MHz, iniciando em 2,402 GHz e terminando em 2,480 GHz. Em alguns países esta freqüência é temporariamente reduzida e 23 saltos são utilizados, seguindo as normas que regulamentam a utilização de bandas de cada país (Tabela 3.2 -[11]). Em ambos sistemas mantêm-se bandas de segurança nos limites inferior e superior.

[2] O formato CVSD usa um algoritmo de modulação delta (esquema mais robusto para modulação de voz) adaptivo com reconhecimento silábico. A voz codificada na interface de linha deveria ter a qualidade igual ou melhor que a qualidade de 64 kb/s log PCM (padrão de compressão de voz – ITU-T recomendação G.711).

PAÍS	ALCANCE DA FREQÜÊNCIA (GHz)
Japão	2,471 – 2,497
França	2,4465 – 2,4835
Espanha	2,445 – 2,475
Outros locais (Europa e América)	2,400 – 2,483

Tabela 3.2 – Limitações regionais da freqüência Bluetooth

Um canal é formado pela seqüência pseudo-randômica de saltos (*pseudo-random hopping sequence*) variando entre os 79 ou 23 canais de rádio freqüência (tabela 3.3 – [17]). Este controle é mantido por chaveamento interno. A taxa de freqüência máxima é 1600 saltos/segundo, e o alcance nominal da conexão é de 10 centímetros a 10 metros, mas pode ser estendido para mais de 100 metros aumentando a potência de transmissão.

Intervalo de freqüência	Freqüência de rádio	Canais
2400 – 2483,5 MHz	f = 2402 + k MHz	k = 0,....., 78

Tabela 3.3 – Freqüências – (Europa e América)

A especificação Bluetooth SIG 1.0 considera uma taxa de transferência de dados total de 1 Mbps. Na transmissão *full duplex* a taxa é de 432,6 Kbps e na transmissão assimétrica os dados são enviados a 721 Kbps, correspondente a três canais de voz, e o retorno a 56 Kbps. A mesma especificação considera para transmissão de voz três canais de voz síncronos de 64 Kbps cada.

Com voz e dados simultaneamente um rádio Bluetooth poderá suportar três canais de voz síncronos simultâneos e um canal de dados assíncrono, ou um canal único que opere simultaneamente dados assíncronos e voz síncrona [11]. O Bluetooth pode manter até 8 dispositivos ativos num canal. Além disto, os dispositivos são acomodados temporariamente em estado de *parked*.

O Bluetooth usa a técnica FHSS tornando-o menos sensível às interferências de outros dispositivos. Um canal de RF interferirá no canal do Bluetooth somente quando operar na mesma freqüência, o que significa somente 1,3 % do tempo sobre um sistema com 79 canais [12].

3.4 Pacotes e Canais Bluetooth

3.4.1 Tipos de canais Bluetooth

Entre dois ou mais dispositivos Bluetooth podem ser estabelecidos dois tipos de canais:

- **SCO** – *Synchronous Connection-Oriented* – Canal simétrico ponto-a-ponto entre um *master* e um simples servidor numa *piconet.* O *master* mantém o canal SCO usando os intervalos de tempo reservados regularmente – comutação de circuitos (*circuit-switched*). Usado principalmente para transportar informação de voz, com o *master* suportando até três canais simultâneos e *slaves* suportando dois ou três canais. Pacotes SCO nunca são retransmitidos (64 Kbps) – Figura 3.3 [18].

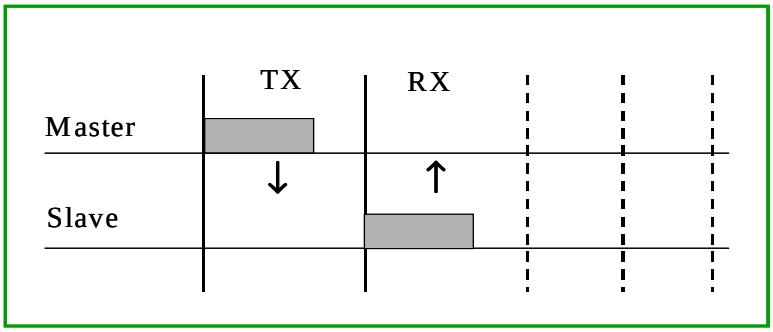

Figura 3.3 – Canal SCO

- **ACL** – *Asynchronous Connection-Less* – Suportam conexões simétricas ou assimétricas, ponto-a-multiponto, comutada por pacotes (*packet-switched*) entre o *master* e todos os *slaves* participando da *piconet.* Nos intervalos de tempo não reservados para os canais SCO o *master* pode estabelecer um canal ACL com base em intervalos para qualquer *slave*, incluindo os *slaves* que já estejam conectados em canais SCO (tipo comutado por pacote). Pode existir somente um canal ACL entre um *master* e um *slave* numa *piconet.* Praticamente todos pacotes ACL podem ser retransmitidos, e estes pacotes podem ter um, três ou cinco intervalos de tempo [19].

3.4.1.1 Formato do pacote Bluetooth

O pacote padrão do Bluetooth consiste de três partes [20]:

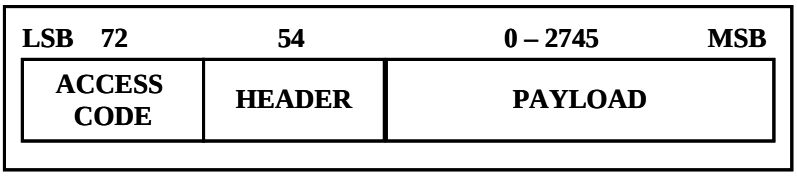

Figura 3.4 – Formato do pacote Bluetooth padrão

3.4.1.2 Access Code

Com tamanho entre 68 ou 72 bits, é usado para identificação dos pacotes transmitidos no canal Bluetooth. Faz sincronização temporal, *offset compensation* (até 6 erros de bits podem ser corrigidos), *paging* (verificar se determinado dispositivo está na área de cobertura) e *inquiry* (para descobrir novos dispositivos).

Um dispositivo Bluetooth monitora o código de acesso de cada pacote; se o dispositivo não é endereçado direta ou indiretamente, o resto do pacote é descartado. Há três tipos de código de acesso:

- *Channel Access Code* (CAC) – Identifica uma e somente uma *piconet*.
- *Device Access Code* (DAC) – Usado para *paging*.
- *Inquiry Access Code* (IAC) – Usado para *inquiry*.

3.4.1.3 Packet Header

Num tamanho de 18 bits o cabeçalho do pacote contém as seguintes informações:

- Endereço do dispositivo destino (slave) em 3 bits, endereçando via um canal Bluetooth o dispositivo ativo. Há também endereços de *broadcast*.
- Um código de 4 bits identificando o tipo de dado ou pacote de controle.
- Campos para controle de fluxo, sequenciamento e reconhecimento de pacotes.
- Um cabeçalho CRC de 8 bits.

O packet header conforme descrito tem somente 18 bits, porém, para proteger contra erros de transmissão o mesmo bit é transmitido três vezes em fila (codificação 1/3 FEC), totalizando um tamanho de 54 bits.

3.4.1.4 Packet Payload

Esta é parte de carrega os dados. Para canais ACL o *payload* inicia com um cabeçalho de 8 ou 16 bits, indicando o tamanho do pacote de dados e fornece campos

para canais lógicos e fluxo de controle, além de suportar fragmentação de pacotes de dados.

Em relação aos canais lógicos, o Bluetooth emprega cinco diferentes tipos de canais para controle da informação:

- Controle da conexão:
  - Gerenciamento do canal (LC – *Control Channel*).
  - Gerenciamento da conexão (LM – *Link Manager*).
- Informação do usuário:
  - Informação assíncrona – (UA – *User Asynchronous*).
  - Informação isosíncrona – (UI – *User Isosynchronous*).
  - Informação síncrona – (US – *User Synchronous*).

É importante observar que um intervalo de tempo para um dispositivo Bluetooth é de 625μs, podendo então transferir 625 bits. Para transmitir 2745 bits o pacote pode ser estendido em até 5 intervalos de tempo, período em que a freqüência do canal é mantida constante.

3.4.2 Tipos de pacotes

Dependendo do tipo de conexão o Bluetooth suporta vários tipos de pacotes, desempenho e *bit faul tolerance*. Em conexões SCO há pacotes para voz com baixa, média ou alta qualidade assim como é suportada a combinação de dados e voz. Para conexões ACL há pacotes para 1, 3 ou 5 intervalos usando taxas de dados médias (5 bits para verificação a cada 10 bits) e altas (sem bits de verificação adicionais).

Para cada tipo de conexão há 12 diferentes tipos de pacotes e 4 pacotes que são comuns às duas modalidades de conexão [20]. Ao nível do *Baseband* do Bluetooth estão definidos os seguintes tipos de pacotes:

- Para conexões ACL e SCO:
  - ID, NULL, POLL, FHS, DM1.
- Somente conexões ACL:
  - DH1, AUX1, DM3, DH3, DM5, DH5.
- Somente conexões SCO:
  - HV1, HV2, HV3, DV.

3.4.3 Correção de erros

Unidades Bluetooth operam freqüentemente em ambientes com ruídos eletromagnéticos, o que requer algum mecanismo para detecção e correção de erros. Para detecção de erros, Bluetooth usa vários cálculos de *checksum*. Quando são identificados erros, há três tipos de esquemas que podem ser aplicados:

- Repetir três vezes cada bit para redundância – 1/3 rate *Forward Error Correction* (FEC).
- Geração polinomial para codificar um código 10-bits num código 15-bits – 2/3 *rate* FEC.
- Retransmissão dos campos DM, DH e dados de pacotes DV até que seja recebida uma confirmação, ou por exceder tempo –esquema ARQ (*Automatic Repeat Request*).

3.4.3.1 FEC – *Forward Error Correction*

O objetivo de aplicar o esquema FEC sobre os dados no *payload* dos pacotes é reduzir o número de retransmissões. Porém, num ambiente com baixo nível de erros, esta abordagem aumenta desnecessariamente o tempo de processamento, reduzindo o desempenho. Com base nestas possibilidades, as definições de pacotes foram mantidas flexíveis no sentido de aplicar o FEC ou não, resultando em:

- pacotes DM e DH para conexões ACL,
- pacotes HV para conexões SCO.

O *header* dos pacotes sempre é protegido por uma taxa 1/3 FEC, pois contém informações de conexão.

3.4.3.2 ARQ – (Automatic Repeat Request)

A figura 3.5 [20] ilustra dois momentos com perda de informações, onde os blocos de dados ficam corrompidos. Isto é detectado pelo receptor. Na próxima oportunidade que o receptor tiver a oportunidade de comunicar com o emissor (no intervalo de tempo apropriado), o receptor enviará ao emissor um NAK (*Negative Acknowledgement*), informando que o bloco de dados terá de ser retransmitido.

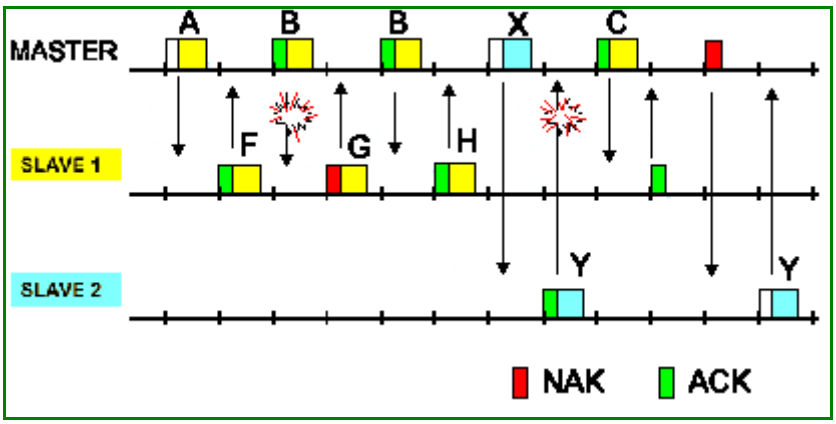

Figura 3.5 – Esquema ARQ

3.5 Pilha de protocolos Bluetooth

O Bluetooth implementa as camadas 1 e 2 da pilha de protocolos OSI, definido como o núcleo do Bluetooth. Na figura 3.6, o núcleo faz parte do conjunto do controlador do *host*. Entre as camadas *Link Manager* e L2CAP do Bluetooth está a interface entre o *host* e o módulo Bluetooth. Os protocolos superiores do Bluetooth formam uma camada de adaptação, chamada de *Network Adaptation Layer* (NAL). As camadas padrão do OSI, junto com a NAL, são funções residentes no *host*.

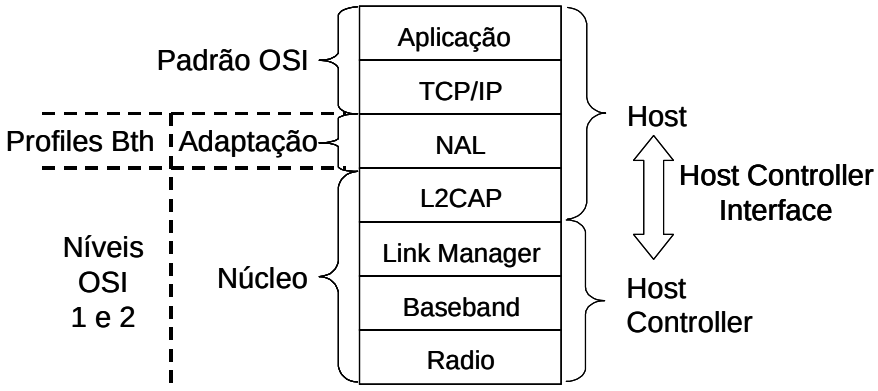

Figura 3.6 – Bluetooth vs Camada OSI

3.5.1 Núcleo de protocolos

Além das especificações físicas dos canais o Bluetooth também inclui um conjunto completo de *software* [21] e algumas recomendações de compatibilidade.

O Bluetooth é baseado principalmente no padrão IEEE 802.11, que define o padrão de comunicação para dois tipos de redes, *ad hoc* e cliente-servidor. Na rede *ad*

hoc as comunicações são estabelecidas entre múltiplas estações em determinada área de cobertura sem o uso de um ponto de acesso ou servidor. O padrão 802.11 especifica a etiqueta que cada estação deve observar para que todas estações tenham acesso ao meio. Métodos para arbitrar as requisições de acesso ao meio asseguram um desempenho maximizado para todos usuários no conjunto básico de serviços. O Bluetooth usa o padrão *ad hoc*.

Na rede cliente-servidor definida pelo padrão 802.11 é utilizado um ponto de acesso para controlar a alocação do tempo de transmissão de cada estação e para permitir o deslocamento das estações móveis de uma célula para outra. O ponto de acesso transfere o tráfego do rádio móvel para o *backbone* cabeado ou sem fio da rede cliente-servidor, além de rotear o tráfego entre as estações móveis. Normalmente a rede cliente-servidor tem melhor desempenho que redes *ad hoc*.

O projeto do sistema foi segmentado em vários níveis que são praticamente independentes, facilitando a descrição conceitual. Os diferentes protocolos seguem uma hierarquia predefinida, com a camada básica contendo os protocolos usados em todas as aplicações da tecnologia. As demais funções são empilhadas sobre a camada básica e cada aplicação individual tem sua pilha de protocolos exclusiva, que na tecnologia Bluetooth exigem uma pilha de protocolos separada para cada perfil Bluetooth (Figura 3.7 – [11]).

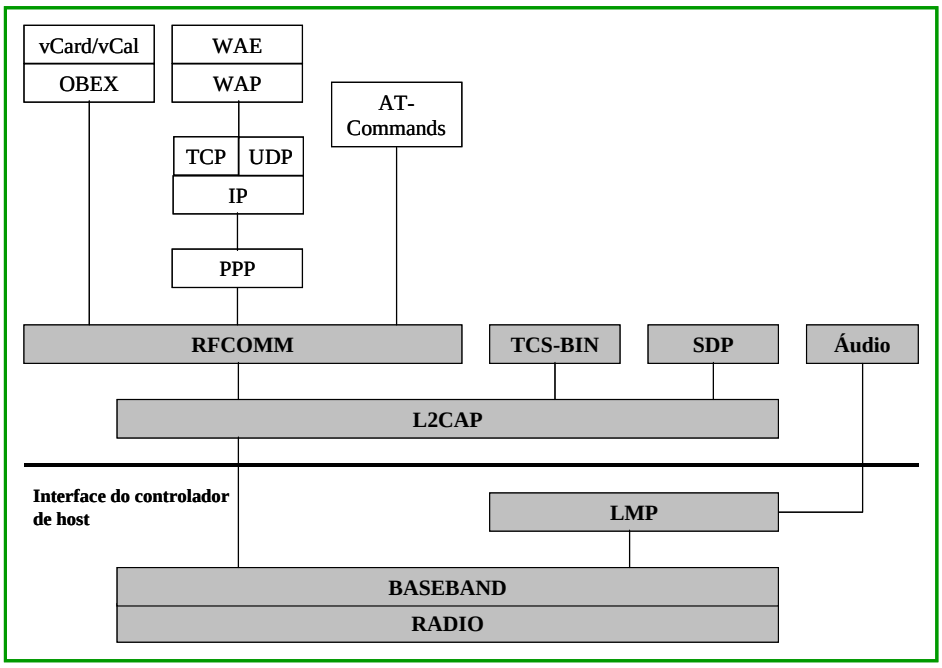

Figura 3.7 – Pilha de protocolos Bluetooth

Os níveis que são implementados em *hardware* ou *firmware* estão divididos em três, formando a base da pilha de protocolo. A base da pilha consiste de um nível chamado de Radio, responsável pela interface de conexão física. Os níveis seguintes, *Baseband* e LMP (*Link Manager Protocol*) estabelecem e controlam a conexão entre unidades Bluetooth.

A interface de controlador de *host* só é necessária quando o L2CAP estiver embutido no *software*. É possível, dependendo do *hardware*, que o L2CAP esteja incorporado no *hardware*, situação que também elimina a necessidade da interface de controlador de *host*. Neste caso, o L2CAP, o LMP e o *Baseband* comunicam-se diretamente.

A pilha de protocolos Bluetooth pode ser dividida em quatro camadas principais, de acordo com a função – Tabela 3.4 [11].

CAMADA DE PROTOCOLOS	**PROTOCOLOS UTILIZADOS**
Principais protocolos (*Core Protocols*)	*Baseband* *Link Manager Protocol* (LMP) *Logical Link Control and Adoption Protocol* (L2CAP) *Service Discovery Protocol* (SDP)
Protocolo de substituição de cabos (*Cable Replacement Protocols*)	RFCOMM
Protocolo de controle telefônico (*Telephony Control Protocols)*	*Telephony Control Specification – Binary* (TCS-BIN) AT-*Commands*
Protocolos adotados (*Adopted Protocols*)	*Point-to-Point Protocol* (PPP) *Transport Control Protocol/Intenet Protocol* (TCP/IP) *User Datagram Protocol* (UDP) *Object Exchange Protocol* (OBEX) *Infrared Mobile Communication* (IrMC) *Wireless Application Protocol* (WAP) *Wireless Application Environment* (WAE) vCard, vCalendar, vMessage e vNote (formatos de conteúdo)

Tabela 3.4 – Camadas da pilha de protocolos Bluetooth

Todos os níveis do Bluetooth Radio até o RFCOMM, também SDP, fazem parte da especificação Bluetooth. Para os demais, por exemplo, WAP, a especificação Bluetooth fornece as instruções que garantem a interoperabilidade.

3.5.2 Baseband, Link Manager Protocol (LMP) e Host Controller Interface (HCI)

Este é o conjunto de protocolos que integram o próprio módulo do dispositivo Bluetooth. Os demais são executados no *host* que faz o controle do módulo Bluetooth.

3.5.2.1 Baseband

O protocolo *Baseband* é responsável pela interface física do canal de rádio freqüência, provendo os saltos de freqüência na comunicação entre duas ou mais unidades Bluetooth. Os canais físicos são gerenciados independentemente de outros serviços como correção de erros, por exemplo. É o protocolo responsável pelo estabelecimento de conexões ACL e SCO entre os dispositivos Bluetooth.

3.5.2.2 *Link Manager Protocol* (LMP)

O LMP é o nível de estabelecimento de conexão entre dispositivos Bluetooth. Opera o controle e a negociação de tamanhos de pacotes quando há transmissão de dados. Opera também o gerenciamento dos modos de potência, consumo de potência, e o estado de um dispositivo Bluetooth numa *piconet.* Além disso, opera a geração, o intercâmbio e o controle de conexões e chaves para autenticação e encriptação.

Classificação das funções básicas do LMP:

1. Gerenciamento *Piconet*.
2. Configuração de conexão.
3. Funções de segurança.

No estado de baixa potência, denominado “*parked*”, muito mais dispositivos podem estar conectados. A comunicação entre os dispositivos de uma *piconet* pode ser através de conexões SCO ou ACL. O compartilhamento do canal é gerenciado pelo *master,* com o auxílio do *Link Manager* em cada unidade. Quaisquer dois ou mais dispositivos que precisam comunicar-se precisam estabelecer uma *piconet* entre elas, sendo que estes dispositivos podem fazer parte de várias outras *piconets* ao mesmo tempo.

A LMP dispõe das funcionalidades para anexar e desanexar dispositivos considerados *slaves,* regras de chaveamento entre um *master* e *slave* e para estabelecer conexões ACL / SCO, além de manter os modos de baixa potência – *hold, sniff e park,* definidos para economia de energia quando os dispositivos não transmitem dados.

Configuração de canal é a realização de uma série de tarefas, incluindo o estabelecimento de parâmetros de conexão, qualidade de serviço e controle de potência se os dispositivos suportarem, autenticação de dispositivos as serem conectados e chaves para gerenciamento de conexões [20, 21].

3.5.2.3 *Host Controller Interface*(HCI)

Para muitos dispositivos o componente Bluetooth pode ser adicionado como um cartão PCI ou um adaptador USB, que normalmente implementa os níveis mais baixos – Rádio, *Baseband* e LMP. O dado a ser enviado para o LMP e *Baseband* trafega sobre uma conexão física como USB. Um driver para esta conexão precisa estar instalado no *host* e um *host controller interface* é necessário para o *hardware* Bluetooth para aceitar dados sobre o meio físico. Desta forma, quando os níveis superiores do Bluetooth estão em *software* e os inferiores em *hardware*, os seguintes níveis adicionais são necessários:

HCI driver O *driver* para a interface do *host controller*. Reside no *host*, sobre o meio físico, e formata os dados para serem aceitos pelo *Host Controller* no *hardware* Bluetooth.

Host Controller Interface Reside no *hardware* Bluetooth e aceita comunicações sobre o meio físico.

O sinal enviado do módulo Bluetooth para o *host* é chamado de um “evento HCI”.

O período de tempo máximo em que o *Baseband* executa tentativas de retransmissão antes do pacote L2CAP ser descartado é chamado de *Flush Timeout*. Este período é estabelecido por canal ACL.

3.5.3 L2CAP

Uma vez que a conexão tenha sido estabelecida entra o protocolo L2CAP para servir às aplicações em níveis de protocolos superiores provendo algumas funções básicas:

Multiplexação

O protocolo permite que múltiplas aplicações utilizem o canal entre dois dispositivos simultaneamente.

Segmentação e Reagrupamento

O protocolo reduz o tamanho dos pacotes enviados pelas aplicações para o tamanho dos pacotes aceitos pelo nível *Baseband*. O L2CAP aceita pacotes com

tamanhos de até 64 kb, mas os pacotes do nível *Baseband* aceitam somente pacotes de 2.745 bits. O processo de reagrupamento é aplicado para os pacotes recebidos.

Qualidade de serviço (QoS)

O protocolo L2CAP permite aplicações que demandam QoS em certos parâmetros como pico de utilização de banda, latência e variação de atraso, verificando se o canal é capaz de prover os requisitos e de provê-lo se for possível.

Estas funções só não são desempenhadas pelo L2CAP se houver um *host controller* em uso. Além disso, o L2CAP pode lidar com pacotes de até 64 kbps em tamanho e suporta somente canais ACL.

3.5.4 RFCOMM

Este protocolo faz a emulação de um protocolo serial, emulando controles RS232 e sinais de dados, provendo recursos de transporte para níveis superiores. Vários protocolos que não são Bluetooth podem usar o RFCOMM como mecanismo de transporte, por exemplo, TCP/IP sobre PPP ou o protocolo OBEX [21].

3.5.5 Perfis de aplicações

Os perfis especificados no Bluetooth representam um importante papel no que concerne aos requisitos de interoperabilidade. Fabricantes e desenvolvedores de dispositivos precisam atender os requisitos dos perfis para habilitá-los para o uso do Bluetooth. Estes perfis atendem certos tipos de usos, como por exemplo o fax.

3.6 Estabelecimento de conexões

3.6.1 Piconet

Há quatro tipos de endereços, derivados dos padrões IEEE 802, que podem ser atribuídos para unidades Bluetooth:

BD_ADDR – Endereço da unidade Bluetooth	Endereço único de 48-bits dividido em: LAP (24-bits), NAP (16-bits), UAP (8 bits).
AM_ADDR – Endereço de membro ativo	Número de 3 bits válido somente enquanto for *slave* num canal ativo (MAC do Bluetooth).
PM_ADDR – Endereço de membro *parked*	Endereço de 8-bits que separa os parked *slaves*. Válido somente enquanto for um *slave parked*.

AR_ADDR – Endereço de solicitação de acesso	Usado pelo *parked slave* para determinar ao meio intervalo de tempo da janela de acesso que está apto a enviar mensagens. Válido somente enquanto for um *slave parked* e não necessariamente único.

Dispositivos Bluetooth que estão ao alcance de outros podem estabelecer conexões. Um ou mais dispositivos usando o mesmo canal, estabelecendo uma conexão, formam uma *piconet.* O canal usado pela *piconet* é identificado por sua seqüência única de saltos (*hop sequence*), cujo endereço é determinado pelo dispositivo *master* do Bluetooth (campo BD_ADDR). O *clock* do dispositivo *master* determina a fase na seqüência de saltos. *Master* é a denominação do dispositivo que inicia as comunicações e que pode ter até sete dispositivos adicionais conectados [20].

A figura 3.8 [20] demonstra os diagramas que ilustram os tipos de redes e as possibilidades dos dispositivos num ambiente Bluetooth:

a) dois ou mais dispositivos Bluetooth usando o mesmo canal formam uma piconet.
b) uma *piconet* entre vários dispositivos – um dispositivo é *master* e os demais *slave*(s).
c) uma *scatternet* – grupo de *piconets* com áreas de cobertura sobrepostas. Cada *piconet* é identificada por uma seqüência diferente de freqüência de salto.

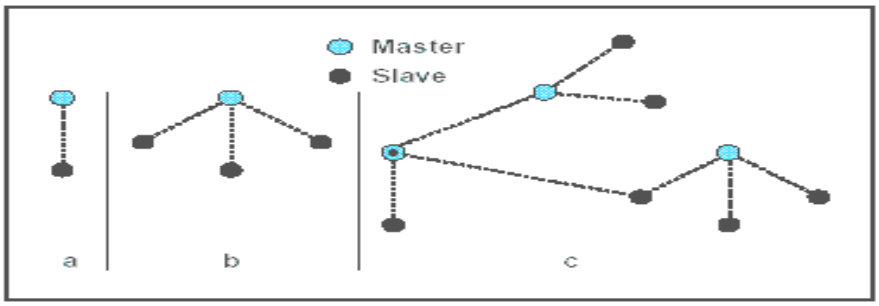

Figura 3.8 – *Piconets* e *Scatternets*

Cada dispositivo Bluetooth tem um sistema de relógio interno chamado de clock nativo (CLKN), cuja finalidade é determinar o tempo de transmissão deste dispositivo. O CLKN nunca é ajustado ou desligado [20].

O canal é dividido em intervalos de tempo (máximo 1600 saltos/seg.) com a duração de 625μs cada, e numerados de acordo com o *clock* do dispositivo master

(CLKN) da *piconet.* Estes intervalos correspondem a uma freqüência de salto de rádio, e saltos consecutivos correspondem a diferentes freqüências de saltos de rádio (Figura 3.9).

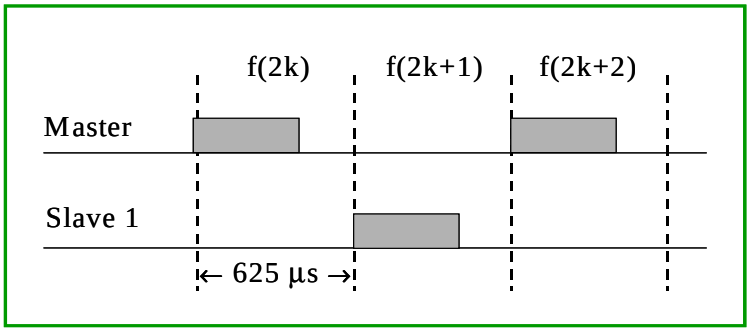

Figura 3.9 – Transmissão *Master – Slave*

O esquema *Time Division Duplex* é utilizada na comunicação *full duplex*, onde *master* e *slave* transmitem alternadamente. A transmissão é iniciada pelo *master* em intervalos de tempo pares, e o *slave* inicia suas transmissões em intervalos de tempo ímpares. O início do pacote é alinhado com o ínicio do intervalo de tempo [11,20].

As características de tráfego Bluetooth são [20]:

1. Taxa de transmissão de 1 Mb/seg., explorando a capacidade máxima disponível do canal.
2. Saltos de freqüência rápidos evitam interferência.
3. Potência de saída adaptiva minimiza interferência.
4. Pacotes de dados pequenos maximizam capacidade durante interferência.
5. Reconhecimento rápido minimiza codificação de controle para conexões.
6. Reconhecimento de voz em codificação *Continuous Variable Slope Delta Modulation* (CVSD) permite operações em altas taxas de erros de bit.
7. Tipos de pacotes flexíveis suportam uma ampla gama de aplicações.
8. Custo baixo de conexão suporta integração de baixo custo num único chip.
9. Interface de transmissão e recepção montada para minimizar consumo de energia elétrica.

Resumindo, numa *piconet*, a identidade do *master* (BD_ADDR) determina a seqüência da freqüência de saltos e o código de acesso ao canal, enquanto o *clock* do sistema *master* determina a fase na seqüência de saltos e determina o tempo [20].

A *piconet* é sincronizada pelo sistema de clock do *master*. Há três tipos de informações que são necessárias para transmitir sobre o canal da piconet:

1) *Channel Hopping Sequence* – O BD_ADDR do *master* é usado para formar a seqüência da freqüência de saltos.
2) *Phase* – O sistema de *clock* do *master* determina a fase na seqüência de saltos.
3) *Channel Access Code* (CAC) – Formado a partir do BD_ADDR do *master*.

3.6.2 Topologia de rede

Uma rede Bluetooth suporta conexões ponto-a-ponto e multiponto. Uma *piconet* é uma rede formada por no mínimo um e no máximo oito *slaves* ativos, sendo que cada *piconet* é diferenciada pelos canais de salto de freqüência. Todas as unidades participando da mesma *piconet* estão sincronizadas no mesmo canal.

3.6.3 Temporização na Piconet

Para determinar quando um novo intervalo de tempo inicia os dispositivos Bluetooth numa *piconet* devem estar sincronizados no mesmo *clock*. Por isto que cada dispositivo Bluetooth tem um *clock* independente, não relacionado a nenhuma fonte externa de tempo, que sinaliza cada 312.5μs e tem um período de 228-1 sinais. O *clock* do dispositivo *master* é usado como o *clock* da *piconet*.

Ao juntar-se numa *piconet* cada dispositivo *slave* determina o tempo de seu *clock* interno com o *clock* do *master*. O ajusto do *clock* interno ocorre com freqüência em função de ser independente de outro meio externo que o controle.

3.6.4 *Formação de piconet*

Uma *piconet* pode ser criada em uma de quatro formas:

1. Um *Page* (usado pelo *master* para conectar com *slave*),
2. Um *Page Scan* (uma unidade escuta seu código de acesso),
3. Através de um chaveamento *Master-Slave*,
4. Através de um *Unpark* de uma unidade (quando não houver *slaves* ativos).

Para estabelecer novas conexões são usados os procedimentos *Inquiry* e *Paging*. O procedimento *Inquiry* permite que uma unidade descubra quais unidades estão na área de cobertura, e quais são os endereços e *clocks*. Com o procedimento *Paging*, uma conexão real pode ser estabelecida, onde somente o endereço da unidade Bluetooth é necessário. Conhecer o *clock* somente irá acelerar o processo de conexão. Uma unidade

que estabelece uma conexão executará um procedimento *Page* e se tornará automaticamente o *master* da conexão.

Há um esquema de *Paging*, dentre vários possíveis, que deve ser suportado por todas unidades Bluetooth. Este esquema é utilizado quando unidades se contactam pela primeira vez, e no caso do processo de *Paging* seguir diretamente o processo de *Inquiry*. Duas unidades conectadas usando um esquema de *Paging/Scanning* mandatório podem combinar um esquema de *Paging/Scanning* opcional [20].

Há duas possibilidades na comunicação entre duas unidades:

- A unidade remota é desconhecida – segue-se então os procedimentos *Inquiry* e *Page*.
- Algumas informações da unidade remota são conhecidas – somente o procedimento de *Page* é necessário.

O procedimento para conexão de uma *piconet* que não existe é iniciado por qualquer uma das unidades, sendo que esta será a *master* da *piconet* criada. Uma conexão é feita por uma mensagem *page* enviada para um endereço já conhecido, ou por um *Inquiry* seguido de uma mensagem *Page* se o endereço for desconhecido.

No estado *Page* inicial, a unidade *master* enviará uma série de 16 mensagems *Page* idênticas em 16 saltos de freqüência diferentes definidos para a unidade a ser pesquisada (unidade *slave*). Se não houver resposta, o *master* transmite uma seqüência nos 16 saltos de freqüência restantes na forma de seqüência para despertar (*wake-up*). O atraso máximo antes de o *master* atingir o *slave* é duas vezes o período de despertar (2.56 segundos) enquanto o atraso médio é metade do período de despertar (0,64 segundo).

A mensagem *Inquiry* é normalmente usada para encontrar unidades Bluetooth. Esta mensagem é muito similar à mensagem *Page*, mas poder requerer uma série de períodos para coletar todas as respostas.

Para unidades conectadas numa *piconet* que não precisam transmitir dados pode ser usado o modo de economia de energia.

A formação de uma *piconet* consiste de três fases:

- *Inquiry* – Nesta fase o dispositivo que está iniciando a conexão pesquisa sua área de cobertura por dispositivos Bluetooth, descobrindo o endereço dos dispositivos a serem conectados.

- *Paging* – É quando acontece a negociação, que basicamente é informar ao slave qual canal Bluetooth deverá utilizar e sincronizar o *clock* para o canal.
- *Connection* – A conexão é estabelecida e a transmissão de dados pode iniciar.

O controlador Bluetooth opera em dois estados superiores (figura 3.10 – [17]):

- *Standby* – estado de baixa potência padrão na unidade Bluetooth. Não há conexões e somente o *clock* nativo está operando. Todos dispositivos Bluetooth estão neste modo antes de qualquer conexão numa *piconet.* Neste modo, uma unidade não conectada periodicamente procura por mensagens a cada 1.28 segundos. Cada vez em que uma unidade acorda, ela ouve um conjunto de 32 saltos de freqüência definidos para ela. Este número varia por região geográfica, mas 32 é o número adotado pela maioria dos países.
- *Connection* – o *master* pode trocar pacotes com o *slave*, usando o código de acesso de canal (*master*) e o *clock* Bluetooth do *master*. O esquema de saltos usado é o esquema de salto de canal.

3.6.4.1 Procedimento de *Inquiry*

O procedimento de *Inquiry* faz o dispositivo descobrir quais unidades estão na área, e determina os endereços e *clocks* para as unidades. Passos:

1. O dispositivo emissor dos pacotes de *inquiry* – estado de *Inquiry*, e então recebe as respostas dos *inquiries* enviados.
2. A unidade que recebe os pacotes de *Inquiry*, destinatária, deve estar no estado de *Inquiry Scan* para receber os pacotes de *Inquiry*.
3. A destinatária entra no estado de *Inquiry Response* e envia uma resposta do *Inquiry* recebido.

Desta forma está concluído o procedimento de *Inquiry*. Agora a conexão é estabelecida usando o procedimento de *Page*.

3.6.4.2 Procedimento de *Paging*

A conexão é de fato estabelecida. A unidade que estabelece a conexão e executará o procedimento *Page* será automaticamente a unidade *master* da conexão. Somente o endereço da unidade Bluetooth é necessário para estabelecer a conexão.

1. A unidade emissora procura outra unidade (o destino) – estado *Page*.
2. O destino recebe a procura, o *Page* – estado *Page Scan*.
3. O destino envia uma resposta ao emissor – estado *Slave Response* – passo 1.
4. O emissor envia um pacote FHS ao destino – estado *Master Response* – passo 1.
5. O destino envia sua segunda resposta ao emissor – estado *Slave Response* – passo 2.
6. O destino e o emissor chaveiam para os parâmetros do canal emissor – estado *Master Response* – passo 2 e estado *Slave Response* – passo 3.

O estado *Connection* inicia com o envio de um pacote POLL enviado pelo *master* para verificar se o *slave* alternou para a temporização e canal de salto de freqüência do *master*. O *slave* pode responder com qualquer tipo de pacote.

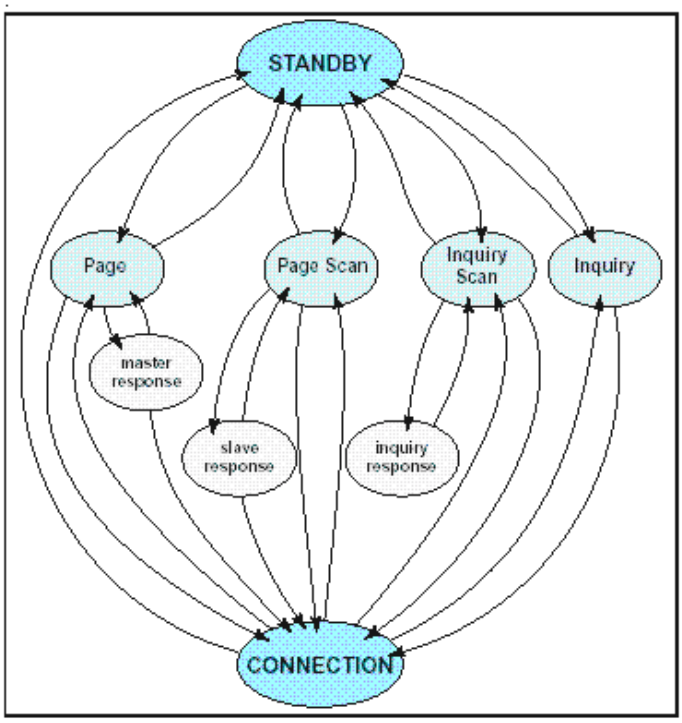

Figura 3.10 – Diagrama de estados do controlador de conexões Bluetooth.

3.7 Scatternets

Várias *piconets* podem estar na mesma área, consequentemente, unidades Bluetooth podem participar em duas ou mais *piconets* que estejam operando em áreas sobrepostas. Uma unidade Bluetooth pode ser slave em várias *piconets* e *master* em

somente uma *piconet.* Um grupo de *piconets* cujas conexões consistem de diferentes *piconets* é chamada de *scatternet*.

Algumas vezes um *master* e um *slave* podem trocar de funções (*master-slave switch*), o que poderá acontecer de duas formas:

1) Um chaveamento TDD entre o *master* e o *slave*, seguido por uma troca de *piconet* entre os participantes.
2) Outros *slaves* da antiga *piconet* podem ser transferidos para a nova *piconet*.

Quando uma unidade acusar o recebimento do pacote FHS, esta unidade usará os novos parâmetros definidos pelo novo *master,* completando o chaveamento de *piconets* [14].

Um *master* or *slave* pode se tornar um *slave* em outra *piconet* pelo processo de *paging* do *master* da outra *piconet.* Isto significa que qualquer unidade pode criar uma nova *piconet* através do *paging* de uma unidade que já é membro de uma *piconet.* Qualquer unidade numa *piconet* pode fazer o *page* no *master* ou *slave* em outra *piconet.* Isto pode levar a mudanças nos papéis de unidades *master* e *slave* numa nova conexão.

Comunicações entre *piconets* são estabelecidas sobre unidade compartilhadas. Multiplexação de tempo precisa ser usado para a unidade compartilhada comutar entre *piconets*. No caso de conexões ACL, a unidade pode solicitar a entrada em modo *hold* ou *park* na *piconet* atual, juntando-se na outra *piconet* durante este tempo alterando somente os parâmetros de canal. Unidades em modo *sniff* podem ter tempo suficiente para visitar outras *piconets* entre os intervalos *sniff*. Se uma conexão SCO estiver ativa, outras *piconets* podem ser visitadas somente nos intervalos (*slots*) não reservados entre eles [17].

Scatternets com boa qualidade é essencial, facilitando as conexões entre *piconets*. A quantificação da qualidade de uma *scatternet* não é fácil, mas os indicadores a seguir podem ser considerados [22]:

- Número de *piconets* – como medida da eficiência de um *scatternet.* Quanto mais *piconets*, maior a probabilidade de colisões ocorrem dentro dos 79 canais compartilhados.
- Nível máximo de dispositivos – o número máximo de *piconets* a que o dispositivo pertence. Se um *slave* pertencer a muitas *piconets*, e levando em conta que a comunicação entre *piconets* acontece via *slaves* compartilhados, um *slave* compartilhado pode ser um gargalo. A

multiplexação de tempo entre as *piconets* a que pertence precisa ser feita e requer uma fatia considerável de tempo.

- Diâmetro da rede – número máximo de saltos entre pares de dispositivos, como os tempos máximos de roteamento da *scatternet.*

O protocolo para formação da *scatternet* deve considerar dois fatores importantes de desempenho [22]:

- Complexidade de tempo – quantidade de tempo para formar uma *scatternet*, que precisa acontecer o mais rápido possível para minimizar o atraso aos usuários.
- Complexidade da mensagem – número de mensagens enviadas entre os dispositivos, conservando o consumo de energia dos dispositivos.

3.8 Transmissão de dados

3.8.1 Acesso ao meio

Num canal Bluetooth o acesso ao meio é obtido pela divisão dos intervalos de tempo em dois grupos: intervalos *master-to-slave* e *slave-to-master*. O *master* pode somente iniciar as transferências em intervalos numerados como pares e as respostas somente em intervalos numerados como ímpares. Os intervalos são numerados pelo *clock* do canal do Bluetooth. Um *slave* somente pode transmitir no intervalo *slave-to-master* se for endereçado previamente no intervalo *master-to-slave*. É responsabilidade do *master* em reservar capacidade suficiente para o canal *slave-to-master* (figura 3.11).

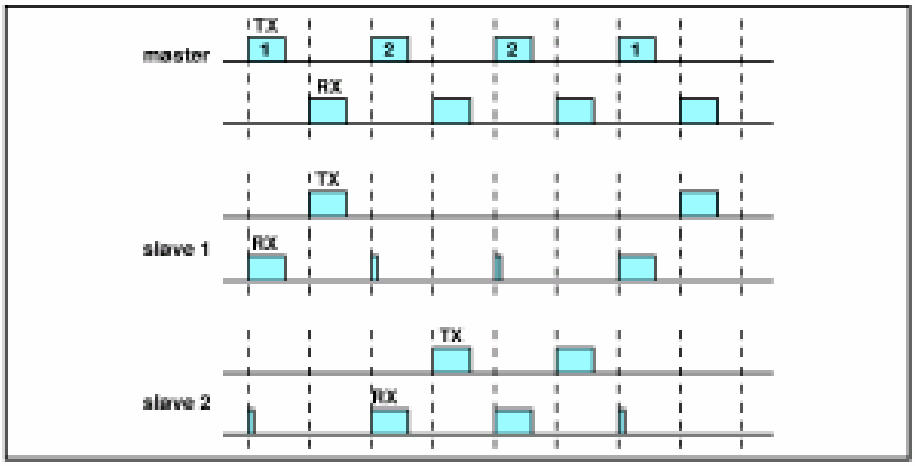

Figura 3.11 – Transmissão e recepção entre *master/slave*

3.8.2 Reconhecimento de pacotes e fluxo de controle

Quando ocorre a transmissão de pacotes dados ou voz/dados um esquema sem numeração ARQ (*Automatic Repeat Request*) é usado. Este pacote é retransmitido até ser recebida a confirmação de recepção, caso contrário ocorre um *timeout*. O reconhecimento, positivo ou negativo, é retirado do cabeçalho do pacote de retorno. Caso o pacote de retorno seja perdido é interpretado como negativo.

Na comunicação *master-to-slave* o pacote de retorno é enviado no intervalo de tempo subseqüente. Quando o *slave* transmite o pacote de retorno é recebido no próximo intervalo em que o *master* endereçar o *slave*.

Um bit de sequenciamento é usado para diferenciar os casos em que um pacote foi recebido com sucesso mas o reconhecimento foi perdido. Se um pacote for reenviado, o número de seqüência é mantido, e o receptor identificará pacotes duplicados.

Caso o *buffer* de recebimento ficar sobrecarregado, o dispositivo Bluetooth pode parar a transmissão enviando um campo de *flow* no pacote de retorno. Isto sinaliza como um sinal de parada mas não afeta pacotes de controles. Se um pacote de retorno não for recebido um sinal de prosseguimento é enviado.

O protocolo *Baseband* aplica filas FIFO em conexões ACL e SCO para transmissão e recepção. O *Link Manager* enche estas filas e o *Link Controller* esvazia-as automaticamente.

Quando uma fila FIFO de recepção estiver cheia, o controle de fluxo é usado para evitar a perda de pacotes e congestionamento. Se um dado não pode ser recebido, uma indicação de *stop* é inserida pelo *Link Controller* do receptor no cabeçalho do pacote de retorno. Quando um transmissor recebe uma indicação de *stop*, ele congela suas filas FIFO. Se o receptor estiver pronto, ele envia um pacote *go* que libera novamente o fluxo [14].

3.8.3 Modos de operação em estado de Connection

3.8.3.1 Active

A unidade Bluetooth participa ativamente do canal. O *master* organiza a transmissão em função da demanda de tráfego originada de ou destinada para diferentes *slaves*. Além disso, suporta transmissões regulares para manter *slaves* sincronizados com o canal. *Slaves* ativos ouvem os intervalos *master-to-slave* por pacotes. Se um *slave* ativo não é endereçado, ele pode aguardar até a próxima transmissão do *master*.

3.8.3.2 Sniff

É o modo em que as unidades sincronizadas numa *piconet* entram em estado de economia de energia, quando o nível de atividades é baixo. A unidade *slave* continua a ouvir a *piconet* numa taxa menor. O intervalo de *Sniff* é programável e depende da aplicação. Dos três modos de economia de energia é o menos eficiente.

3.8.3.3 Hold

Um dos modos de economia de energia para unidades sincronizadas numa *piconet.* O *master* pode colocar unidades *slave* em modo *hold*, assim como unidades *slaves* podem solicitar que sejam colocadas neste modo. Neste modo somente o relógio interno continua executando, e a transferência de dados reinicia automaticamente quando unidades saem deste modo.

3.8.3.4 Park

É o modo com maior economia de energia. Neste modo, uma unidade permanece sincronizada mas não gera tráfego. Seu endereço MAC (AM_ADDR) é descartado e ocasionalmente é ouvido tráfego do *master* para re-sincronização e verificação de mensagens de *broadcast*.

3.9 Finalizando uma conexão

A conexão entre dois dispositivos Bluetooth pode ser fechado em qualquer instante pelo *master* ou pelo *slave* enviando um comando "*detach*" ou "*reset*".

3.10 Conclusões do capítulo

O Bluetooth pode suportar canais de dados assíncronos e até 3 canais síncronos para voz simultaneamente. Até sete dispositivos podem estar ativos numa *piconet*, além do *master*. Este número pode ser maior se forem considerados dispositivos em estado *parked*. Utilizam dispositivos de rádio-freqüência de baixa potência e operam na faixa ISM do espectro magnético. Formam as redes *ad hocs*, com característica de grande mobilidade e durações de comunicações relativamente baixas.

Duas limitações podem ser apontadas no esquema do Bluetooth. A primeira está na necessidade de comunicação direta entre os dispositivos e, conseqüentemente limitados pela qualidade de serviço do canal de rádio entre eles. A segunda está na na

falta de suporte para o movimento de dispositivos ativos de uma interface de rede para outra. Essencialmente, um dispositivo Bluetooth numa *piconet* não pode ser comunicar-se com outro dispositivo Bluetooth numa *piconet* diferente sem que esta tenha sido previamente parte da primeira *piconet*.

4. QUALITY OF SERVICE – QoS

4.1 Introdução

Qualidade de serviços (QoS) em rede, refere-se à capacidade de prover melhores condições e recursos para tráfego pré-selecionado. Dentre os serviços empregados para prover qualidade de serviços são suporte para largura de banda, minimização de perdas de transmissão, gerenciamento e eliminação de congestionamento da rede, modelagem do tráfego da rede e definição de prioridades sobre o tráfego.

Para implementar uma arquitetura de qualidade de serviços são necessários três componentes. O primeiro destes componentes é a aplicação de mecanismos de enfileiramento, escalonamento e modelagem de tráfego sobre elementos básicos da rede. As técnicas de sinalização para coordenação de serviços entre os elementos básicos da rede referem-se ao segundo componente. O terceiro componente envolve as funções de policiamento e gerenciamento para controlar e administrar o tráfego sobre a rede.

4.2 Definições de QoS Networking

A funcionalidade de prover qualidade de serviços por elementos da rede é a capacidade de estabelecer de garantias de desempenho para o tráfego aplicado. Quando um serviço requer determinadas condições de tráfego para o elemento da rede, e este é provido de mecanismos para prover tais condições, então este elemento implementa a funcionalidade de qualidade de serviços.

Bernet [23] definiu qualidade de serviços como sendo “a capacidade de controlar mecanismos de tratamento de tráfego na rede de tal forma que a rede atenda os serviços necessários para certas aplicações e usuários sujeitos às políticas da rede”.

Nesta definição estão enfatizados dois conceitos. No primeiro, a capacidade de controlar mecanismos de tratamento de tráfego, o que implica na interação efetiva no gerenciamento dos recursos da rede para prover os serviços necessários.

No segundo conceito, “sujeito às políticas da rede”, que implica em assegurar que o processo de gerenciamento e alocação de recursos da rede está determinado por regras previamente acordadas.

Outro conceito a ser explicitado é alocação de banda variável. Esta pode ser considerada como uma forma de implementar uma aplicação de qualidade de serviços, pois a aplicação se adapta às condições da rede, dentro de certas condições. Em [24, 47] são apresentadas formas de adaptação de banda para redes sem fio *ad hoc*.

A qualidade de serviços em redes, independente do tamanho da rede, é formada pela concatenação da qualidade de cada um dos saltos (*hops*) numa rota de comunicação. As redes IP tradicionais, Internet, por exemplo, utilizam roteamento pelo melhor esforço (*best effort*), cujo princípio é baseado em filas FIFO - *First in First out*. O emprego de filas introduz os problemas da latência e da perda de pacotes em redes congestionadas, e, derivado da latência, o problema da variação dos atrasos entre os pacotes, conhecido como *Jitter*.

Partindo da afirmação anterior de que a qualidade de serviços em uma rede é dependente da qualidade com que cada *hop* trata os pacotes, identificamos um primeiro requisito, a qualidade de serviços por salto (*hop*). Sendo que o menor elemento é composto por aqueles que integram dois mais canais de comunicação, cada um deve ter sua própria característica para tratar a qualidade de serviços na comunicação entre os nós.

Outro aspecto que surge é a possibilidade de várias rotas. Notadamente em redes sem fio, e nestas, as redes *ad hoc*, vários caminhos paralelos podem existir entre dois pontos. Rotas podem existir e deixar de existir com alta frequência, dependendo da mobilidade dos nós integrantes das rotas. Várias rotas têm como vantagem redução da carga no caminho entre dois pontos, e consequentemente, a redução da possibilidade de perda de pacotes e do *jitter*.

Prover redes com qualidade de serviço depende basicamente de mecanismos que permitem o tratamento do tráfego da rede. Estes mecanismos implementam controles de acordo com políticas da rede. As políticas são aplicadas para usuários e aplicações.

O objetivo maior da qualidade de serviços em redes não tem enfoque individualizado para aplicações ou usuários. O objetivo é maximizar a utilização da rede para todas aplicações e usuários. Mecanismos de qualidade de serviços são aplicados para controlar a alocação de recursos entre aplicações e usuários [23,25,26].

No ambiente da Internet, o aumento da largura de banda para evitar congestionamento tem sido a solução mais óbvia. Porém, o problema é mais do que uma simples questão de capacidade, pois não somente o tráfego aumentou em volume, mas também mudou em sua natureza. Há novos tipos de tráfego, novas aplicações, enfim, variações significativas nos requisitos operacionais.

Largura de banda é o ponto principal e aquele que deve ser gerenciado de acordo com a demanda das aplicações. Os requisitos solicitados para a rede devem ser monitorados e gerenciados. Porém, quando há compartilhamento do canal com outros serviços e aplicações, não há mais garantias, ou então estas se tornam frágeis. Uma aplicação numa rede que não implementa qualidade de serviços, e que requer determinada banda pode ter seu requisito não mais atendido a partir da presença de outra aplicação usando parcial ou totalmente a banda disponível. Portanto, a garantia de qualidade de serviços precisa, através do gerenciamento da alocação de recursos, tratar os serviços isoladamente.

Tipo de taxa de dados	**Descrição**
Stream	Previsível. Taxa constante de bits (CBR – *Constant bit rate*).
Burst	Imprevisível. Taxa variável de bits (VBR – *Variable bit rate*).

Tabela 4.1 – Previsibilidade das taxas de dados.

As aplicações da rede podem ser caracterizadas em termos de previsibilidade da taxa de dados e da tolerância em relação aos atrasos (Tabela 4.2). Aplicações podem gerar uma taxa constante de bits ou então operar em modo de rajada, intercalando, aleatoriamente, instantes com alta taxa de transferência de bits e instantes sem transferência (tabela 4.1).

O tráfego gerado pelas aplicações pode também ser classificado quanto à sua tolerância a atrasos. A tabela 4.2 apresenta uma classificação dos tipos de tráfegos em função da sua tolerância aos atrasos.

Tolerância a atrasos	Tipo de tráfego	Descrição
Alta	Assíncrono	Sem restrições para liberação.
	Síncrono	Sensível ao tempo, mas flexível.
	Interativo	Atrasos podem ser percebidos por usuários e aplicações, mas não afeta utilização e funcionalidade..
	Isócrono	Sensível ao tempo a ponto de afetar sua utilização.
Baixa	Missão crítica	Atrasos afetam a funcionalidade.

Tabela 4.2 – Sensibilidade da aplicação aos atrasos de tráfego.

4.2.1 Recursos de redes

Aplicações e usuários demandam diferentes recursos de redes e geram diferentes tráfegos. Taxas diferentes de carga são submetidas às redes, com variações diferentes, com pacotes de tamanhos diferentes, em horários diferentes. Tudo isto aplicado sobre redes que tem recursos finitos. Para alocar recursos de rede às aplicações e usuários devem ser considerados os principais requisitos:

- *Bandwidth* – largura de banda, a taxa de tráfego necessária para uma aplicação.
- Latência – indicando o atraso que uma aplicação pode tolerar para transmissão de pacotes (retardo).
- *Jitter* – a variação da latência (do retardo).
- Perda de pacotes – o percentual de pacotes de dados perdidos.

Interfaces de equipamentos de redes que enviam e recebem tráfego em determinadas taxas de transmissão. Quando uma interface recebe mais tráfego do que consegue enviar adiante ocorre o congestionamento. Para tratar de congestionamento são aplicados mecanismos de enfileiramento, usando a memória dos dispositivos. Outro mecanismo é descartar pacotes. No primeiro é percebido o problema de variação de latência e no segundo a perda de pacotes.

A classificação do tráfego é outro mecanismo que pode ser empregado. Constitui-se de desenvolvimento sobre as filas geradas nos dispositivos, e são internos aos

dispositivos da rede, tendo a função de determinar qual tráfego é preferencial para uso dos recursos do dispositivo.

O ponto principal nos mecanismos de controle de tráfego é sua capacidade de classificar, enfileirar e escalonar, de forma diferenciada, todo tipo de tráfego, conforme necessário.

A classificação é obtida dos pacotes, que sendo diferenciados permitem a associação adequada de recursos. Os pacotes são então encaminhados para diferentes filas. Estas filas são servidas por algoritmos específicos que determinam a taxa com que o tráfego de cada fila é submetido à rede.

Portanto, para prover qualidade de serviços, os dispositivos que formam a rede devem ser configurados com:

- Informação de classificação para os dispositivos separarem o tráfego em diferentes filas;
- Filas e algoritmos de filas que tratem do tráfego separado nas filas.

Há essencialmente duas formas de qualidade de serviços [27]:

- **Reserva de recursos** (Serviços integrados): recursos são alocados para atender as solicitações de qualidade de serviços das aplicações, conforme critérios de políticas de gerenciamento de banda.
- **Priorização** (Serviços diferenciados) o tráfego de rede é classificado e os recursos alocados conforme critérios de políticas de gerenciamento de banda.

4.3 Taxonomia de mecanismos

Filas e algoritmos de serviços de filas são elementos críticos no tratamento de tráfego para prover qualidade de serviço. Estes mecanismos aumentam a eficiência dos recursos utilizados, aparentando serem processos que criam banda de transmissão.

Sempre que filas são formadas, algoritmos de serviços são os responsáveis por determinar a ordem e a taxa com que pacotes são retirados da fila e transmitidos. Algoritmos de serviços de fila são frequentemente complementados com esquemas de medição, policiamento e descarte (eliminação).

Mecanismos, ou esquemas, de filas devem prover qualidade de serviços tratando determinado tráfego preferencialmente em relação a outro tráfego.

Cada esquema de fila pode ter uma configuração ativa ou passiva. A maioria dos esquemas apresentada a seguir é ativa e foram categorizados conforme proposta apresentada em [23]. Mecanismos passivos são caracterizados pela sua neutralidade quanto ao fluxo de tráfego.

4.3.1 First In First Out - FIFO

Nesta técnica, a ordem de chegada dos pacotes determina totalmente a ordem de atendimento. Presume-se que o controle de congestionamento é implementado pelas fontes geradoras dos pacotes. O controle de congestionamento implementado nas fontes deve reduzir a taxa na qual são enviados pacotes quando perceberem congestionamentos.

A implementação de qualidade de serviços utilizando a técnica FIFO pode ser provida através da diferenciação de pacotes. Os pacotes podem ser tratados antes de entrarem na fila. Pacotes já liberados para a fila não podem mais ser alterados.

Este mecanismo de enfileiramento pode privilegiar uma aplicação em detrimento de outra, mas não pode garantir o controle do serviço sobre o fluxo de tráfego. Uma aplicação gerando muitos pacotes pode ser privilegiada na taxa de envio de pacotes enquanto que, uma aplicação com geração regular pode ficar privada da regularidade de envio dos seus pacotes.

FIFO é um esquema limitado para suportar qualidade de serviço pelo fato de não alterar a ordem dos pacotes enfileirados.

4.3.2 Técnica conservativa

Nesta técnica procura-se evitar que os pacotes sejam enfileirados. Na medida em que o tráfego chega é liberado imediatamente, não formando filas. Porém, se o fluxo de entrada for maior que o fluxo de saída ocorre a formação de filas. Estas filas podem ser tratadas por algoritmos específicos que irão selecionar os fluxos apropriados para liberar os pacotes a serem transmitidos. É um processo em que a transmissão é assegurada, estando sempre em uso.

Uma das formas de organizar os algoritmos de seleção de filas é com base em estabelecimento de prioridades das filas. Filas de baixa prioridade somente serão atendidas se não houverem pacotes a serem transmitidos em filas de prioridade superior.

Algoritmos de tratamento de filas geralmente são usados para compartilhar a capacidade limitada de recursos da rede com fluxos que obtém benefícios obtendo maior capacidade de recursos.

Exemplos de esquemas de enfileiramento com conservação de trabalho:

- *Strict Priority Queuing*
- *Fair Queuing Algorithms*
- *Nagle's Fair Queuing*
- *Bitwise Round Robin*
- *Fair Queuing*
- *Stochastic Fair Queing*
- *Deficit Round Robin*
- *DRR+ or Class DRR*
- *Weighted Fair-Queuing Schemes*

Esquema *Fair Queueing* - FQ

Neste esquema, ocorre a atribuição de um tempo de finalização para cada pacote, um valor teórico no qual o pacote seria transmitido completamente se todas as filas fossem servidas num esquema de *Bitwise Round Robin*[3] Os pacotes são inseridos numa lista de pacotes e organizados de acordo com o tempo de finalização. O escalonador retira pacotes desta lista quando houver capacidade disponível. Pacotes com tempos de finalizações menores são enviados antes dos que tiverem tempos maiores.

O objetivo deste esquema, desenvolvido por Demers, Keschav e Shenker, era solucionar o problema do fluxo do esquema *Nagles Fair Queuing* [28]. No esquema *Bitwise Round Robin*, cada fluxo pode enviar um bit por vez num modelo *round robin*. Porém, dada a impossibilidade de implementação desta solução, é sugerida uma aproximação que simula o esquema. A aproximação utilizada cacula o tempo que um pacote teria para deixar o sistema. Os pacotes são inseridos numa fila e organizados por ordem de saída. Mas a inserção de pacotes numa fila ordenada tem um custo elevado. Os melhores mais conhecidos para inserção e retirada de filas requerem uma complexida de tempo O(log(n)), sendo *n* o número de fluxos.

[3] Algoritmo que retira da fila um simples bit em cada turno, tornando irrelevante a distribuição dos tamanhos dos pacotes.

Um esquema simples para servir FQ iria requerer O(log(m)), onde *m* é o número de pacotes no sistema. Mas Keshav [29] mostrou que somente uma entrada por fluxo precisar ser inserida numa fila ordenada. Isto ainda resulta numa complexidade O(log(n)). Outras sugestões de implementação de Keshav tomaram no mínimo um tempo O(log(n)) no pior caso.

Esquema *Stochastic Fair Queuing* - SFQ

Proposto por McKenney [30], também tinha o objetivo de solucionar algumas ineficiências do esquema *Nagle's Fair Queuing* [31]. SFQ implementa um esquema de *hash* para mapear os pacotes entrantes para suas filas correspondentes.

Enquanto algumas variações poderiam sugerir o hash e o identificador do pacote para estabelecer um elo com uma fila correspondente, outra variação poderia sugerir uma fila para cada tipo possível de fluxo entrante. Porém, McKenney propôs um número de filas consideravelmente menor do que o número possível de fluxos. Todos fluxos que forem direcionados para a mesma fila têm tratamento equivalente. O processo de computar o *hash* é simplificado (garantido agora em O(1) no pior caso), e também permite a utilização de um número menor de filas. A desvantagem é o tratamento "injusto" para os fluxos que colidirem com outros. As garantias de "justiça" são probabilísticas. Se o tamanho do índice de *hash* for maior o suficiente do que o número de fluxos ativos através do sistema, a probabilidade de "injustiças" será pequena. A diferença com o esquema Nagle's é que o número de filas precisa ser somente um múltiplo do número de fluxos ativos, ao contrário do número de fluxos possíveis [32].

As filas são servidas num ciclo *round robin*, sem considerar os tamanhos dos pacotes. Quando não há *buffers* para armazenar os pacotes, o pacote ao final da fila maior é descartado. O processo de descarte é implementado não pelo descarte quando já estiver na fila, mas pela recusa de entrada do novo pacote. McKenney mostra como implementar o esquema de bufferização com complexidade de tempo O(1) usando técnicas de ordenamento.

Esquema *Deficit Round Robin* - DRR

Servir filas num esquema simples *round robin* pode ser feito constantemente a qualquer tempo. O problema maior é a “injustiça” causada pelos diferentes tamanhos de pacotes dos diferentes fluxos.

Nas figuras 4.1, 4.2 e 4.3 é representado um diagrama de funcionamento do esquema DRR. O ponto de referência é o *token*, que a cada ciclo reduz o *deficit*, liberando pacotes para transmissão.

DRR utiliza *Stochastic Fair Queuing* para atribuir filas aos fluxos. Para tratar as filas é utilizado um serviço *round robin* e um índice de serviço é atribuído a cada fluxo. A diferença com o *round robin* tradicional é que se uma fila não estava habilitada para enviar um pacote no ciclo anterior porque seu pacote era muito grande, o resto do índice anterior é adicionado ao índice do próximo ciclo. Assim, os *deficits* são mantidos e as filas que pouco mudaram num ciclo são compensadas no próximo ciclo [33].

É um esquema capaz de prover um controle mais eficiente em relação à capacidade de envio garantias de latência para determinados fluxos.

DRR tem complexidade O(1) um custo significavamente menor [23]. Em cada turno um número fixo de créditos é adicionado a cada fila. O pacote no início da fila é enviado somente se o tamanho do pacote for igual ou menor que o número de créditos acumulados pela fila. Se o pacote for enviado os créditos são reduzidos de forma equivalente, sendo desta forma que DRR faz a contabilização por tamanhos de pacotes. Pacotes adicionais são enviados do início da fila até que não tenha mais créditos suficientes para o próximo pacote ou não houver mais pacotes na fila.

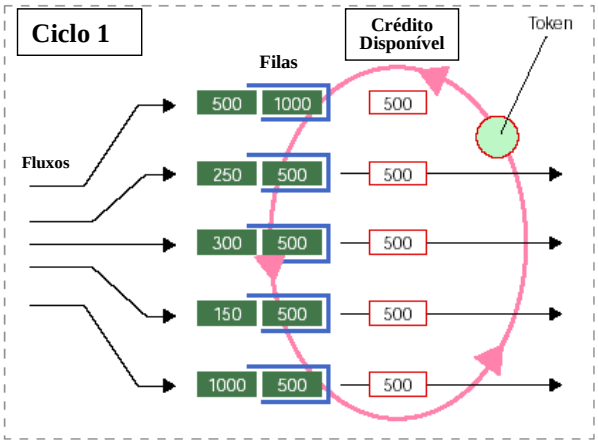

Figura 4.1 – Esquema *Deficit Round Robin* – (1/3)

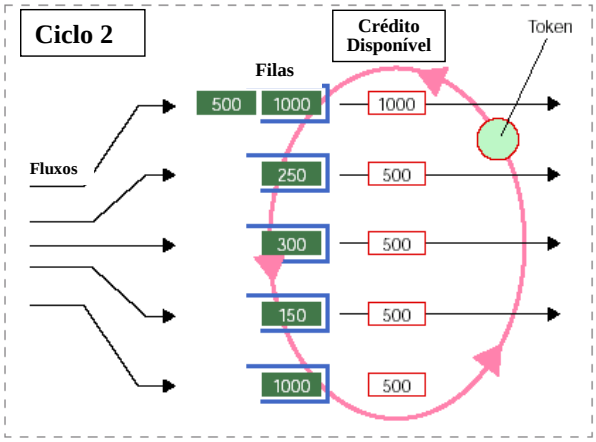

Figura 4.2 – Esquema *Deficit Round Robin* – (2/3)

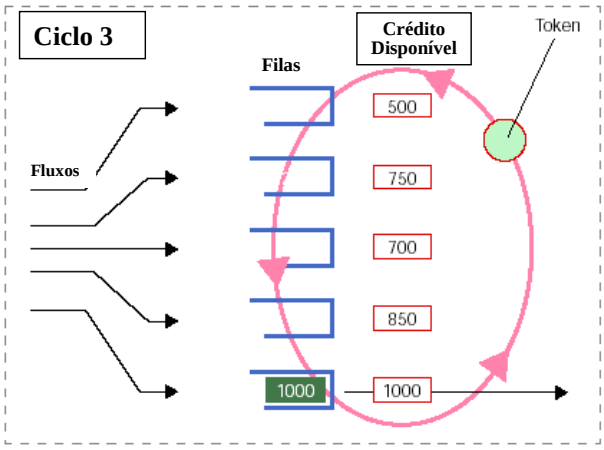

Figura 4.3 – Esquema *Deficit Round Robin* – (3/3)

4.3.3 Técnica não-conservativa

Este é um processo que pode ser visto como tendo um semáforo controlando o fluxo de tráfego. Algoritmos que implementam esta técnica geralmente são usados para limitar o impacto do fluxo de tráfego. Pode ser vista como uma aplicação de policiamento, pois limita a capacidade disponível para fluxos individuais com o objetivo de proteger um recurso compartilhado por múltiplos fluxos.

Exemplos de esquemas de enfileiramento sem conservação de trabalho:

- *Shaping Parameters and Buckets*
- *Leaky Bucket Model*
- *Token Bucket Model*
- Uma combinação entre *Token Bucket* e *Leaky Bucket*.

4.3.4 Esquema de descarte – Minimizar congestionamento

Nos sistemas com enfileiramentos e algoritmos para gerenciamento vistos até aqui, os pacotes são retirados da fila após terem sido transmitidos pelo emissor. Outras alternativas são necessárias nas situações em que o fluxo de entrada é superior ao fluxo de saída, acrescentando-se a limitação de recursos que sustentam a fila, como memória, por exemplo.

Uma das alternativas é eliminar pacotes antes de serem enfileirados, evitando o congestionamento dos recursos de gerenciamento e manutenção das filas. São também conhecidos como esquemas baseados em *buffers*.

Exemplos de esquemas de descarte:

- *Random Early Detection*
- *Weighted Random Early Detection*
- Adequação entre esquemas de descarte com vários tipos de tráfegos.

Esquema *Random Early Detection* - RED

É uma alternativa ao esquema FIFO, que descarta pacotes entrantes quando a fila estiver cheia. No esquema RED, pacotes são descartados mesmo que já estejam na fila. Implementações simples selecionam determinada porcentagem de pacotes a serem descartados randomicamente quando a fila atinge determinados níveis.

4.3.5 Garantias de latência e largura de banda

Esquemas de enfileiramento proporcionam qualidade de serviços (QoS) pelo controlae da capacidade de transmissão ou da banda disponível para certos fluxos de tráfegos e pelo controle da latência de pacotes destes fluxos. A capacidade da interface impõe um limite superior para a taxa total no qual o tráfego pode ser enviado.

Definida a capacidade de transmissão, uma forma de acompanhar a efetividade é através da velocidade com que determinado fluxo é transmitido. O acompanhamento pode também ser exercido pelo controle da latência de pacotes em determinados fluxos, ou seja, o tempo máximo que determinados pacotes precisam esperar antes de serem transmitidos. Geralmente a latência é impactada pela capacidade da interface, pela largura de banda provida para o fluxo correspondente e pela seqüência em que o fluxo é servido na interface.

4.4 Conclusões do capítulo

Qualidade de serviços em redes visa essencialmente controlar mecanismos que permitem o tratamento do tráfego da rede, atendendo aos requisitos que as aplicações requerem para que suas funcionalidades se tornem efetivas e factíveis. Este controle deve estar sujeito às políticas da rede.

Os requisitos básicos são recursos de banda, latência, *jitter* e perda de pacotes. O gerenciamento destes requisitos permite o estabelecimento adequado dos requisitos de qualidade de serviços das redes.

Prover qualidade de serviço pode ser feito de duas formas, priorizando serviços ou reservando recursos. Nas redes sem fio os fluxos de dados devem ser analisados sob estas formas de provisão de qualidade de serviços, pois compartilham a interface de rádio nos dispositivos local e remoto.

No próximo capítulo será analisado o problema específico do Bluetooth em prover qualidade de serviços.

5. QoS NO BLUETOOTH

5.1 Introdução

O Bluetooth implementa algumas funcionalidades para prover qualidade de serviços em canais ACL. A configuração de parâmetros no nível do protocolo L2CAP permite estabelecer algumas características como identificação do nível de serviço, carga média de tráfego, tamanhos máximos a serem transmitidos, atrasos e variações.

A interface do *Host Controller*, empregando um esquema FIFO, concentra todo o tráfego do dispositivo. Através de comandos específicos interage com o sistema para obter informações sobre o estado da interface.

Apesar desta interação, há uma série de deficiências em prover qualidade de serviços no Bluetooth. Uma delas, o controle de fluxo na interface, pode resultar em dois problemas potenciais, ou congestionando a *piconet*, ou congestionando o fluxo sobre múltiplas interfaces numa *scatternet*.

5.2 Canais ACL para aplicações sensíveis a atrasos

Canais ACL podem ser configurados para prover qualidade de serviços através de comandos HCI *QoS Setup*, especificando quais são os requisitos de tráfego e de qualidade (QoS) junto com a solicitação do serviço. As atuais implementações no *Baseband* não suportam o único serviço definido no Bluetooth, o serviço garantido, similar ao definido na arquitetura de serviços integrados (*Integrated Services*). Novos algoritmos de *polling* precisam ser desenvolvidos para suportar este tipo de serviço.

Os canais ACL podem prover confiabilidade em casos de interferência e em casos de erros de bits. Os atrasos causados por retransmissões são pequenos o suficiente para descartarem a necessidade de melhorias específicas, com confirmações sendo recebidas em 1.25 μsegs. Isto abre a possibilidade para executar retransmissões para aplicações sensíveis a atrasos, como aplicações interativas, em tempo real e streaming de áudio/vídeo. A retransmissão pode ser evitada setando *Flush Timeout* quando não forem mais necessárias.

Os canais ACL podem suportar banda variável e assimétrica, requeridas por algumas aplicações. Esta é a maior vantagem em utilizar canais ACL no lugar de canais SCO para aplicações de tempo real. Entretanto, o serviço de melhor esforço provido pelos canais ACL não garante os requisitos de atrasos e banda requeridos por este tipo de aplicações. O serviço garantido não provê requisitos de banda e atrasos, permanecendo um ponto a ser desenvolvido [34].

5.3 Controle de recursos

Há no máximo um canal ACL entre dois dispositivos Bluetooth, o que implica em compartilhamento do canal por aplicações executando no mesmo dispositivo. O tráfego gerado por cada aplicação compete por recursos no canal ACL. Este tráfego pode ter diferentes requisitos de qualidade de serviços (QoS) em termos de banda e atrasos. Além disso, dispositivos na mesma *piconet* precisam compartilhar entre si a banda disponível, resultando em contenção de recursos entre os dispositivos na mesma *piconet.* Neste caso não há garantias de que os requisitos de QoS de cada fluxo sejam atendidos. O controle sobre a alocação dos recursos é requerido para garantir que os requisitos de QoS de cada fluxo sejam satisfeitos [34].

5.4 Configuração Bluetooth

Qualidade de serviços é necessária quando há contenção de recursos entre fluxos de tráfegos. O Bluetooth provê alguns recursos de configuração que permitem o estabelecimento de requisitos qualitativos para fluxos de tráfegos. Trata-se mais de controle dos recursos disponíveis do que melhoria da capacidade do sistema. Exemplos de configurações são: definição do *Flush Timeout,* controle de *Multi-slot* e intervalo máximo de *polling*.

O nível L2CAP dispõe de algumas opções para negociação do tipo de serviço, tráfego e parâmetros de qualidade de serviços (QoS), conforme listado na tabela a seguir [34]:

Parâmetros de configuração	**Unidade**	**Descrição do parâmetro**
Flags	-	Reservado para uso futuro
Service Type	Sem tráfego Melhor esforço (default) Garantido	Identificar o nível de serviço.
Token Rate	Bytes / segundo	Carga de média de tráfego
Token Bucket Size	Bytes	Tamanho máximo a ser transmitido
Peak Bandwidth	Bytes / segundo	Taxa máxima de transmissão da origem
Latency	Microsegundos	Atraso máximo entre a geração do pacote e o início da transmissão do pacote.
Delay Variation	Microsegundos	Diferença entre atrasos máximos e mínimos. Pode ser usado para determinar o tamanho do *buffer* no ponto de recepção.

Tabela 4.3 – Parâmetros de configuração de QoS no nível L2CAP

Os parâmetros de qualidade de serviços, tráfego e tipo de serviço da tabela 4.3 são efetivados através do comando *QoS_Setup* do nível *Baseband*, exceto o parâmetro *Token Bucket Size*, não incluído neste comando.

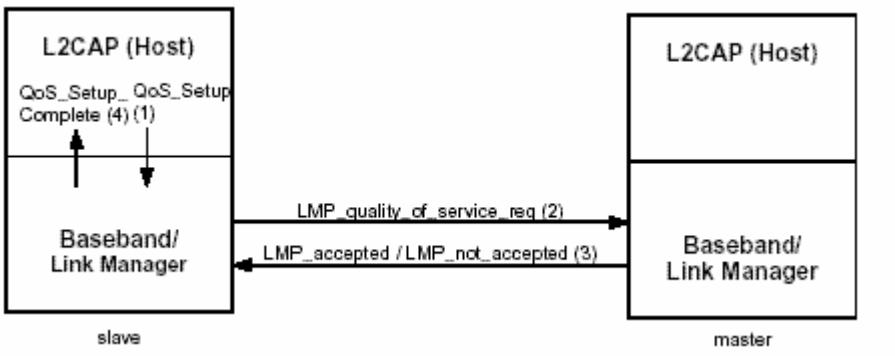

Figura 4.2 – Estabelecimento de QoS no Bluetooth

O comando *QoS_Setup* no nível da interface do *Host Controller* dispara para o nível LMP um *LMP_quality_of_service_req* se for *slave*, ou então, se for *master*, um *LMP_quality_of_service* para solicitar a qualidade de serviço (QoS) desejada. No caso

do *master* respondendo ao *slave* há duas mensagens possíveis que podem ser retornadas, aceitando a solicitação, via comando *LMP_accepted*, ou rejeitando a solicitação, via comando *LMP_not_accepted*.

O nível LMP gera um dos dois eventos para o L2CAP: *QoS_Setup_Complete* ou *QoS_Violation*, confirmando ou rejeitando para a aplicação o requisito de QoS emitido. A figura 4.2 mostra o diagrama da sequência de comandos LMP para estabelecimento de qualidade de serviços entre dispositivos Bluetooth [34].

5.5 Largura de banda – Bandwidth

Para que uma aplicação de comunicação execute satisfatoriamente, alguma quantidade de banda deve estar disponível sobre o canal Bluetooth. A quantidade de banda disponível influencia os atrasos decorrentes de enfileiramentos na transferência de dados. No Bluetooth a banda é determinada pelo algoritmo de *polling* executado pelo *master* da *piconet* e pelo tipo de pacote *Baseband* escolhido para transmissão pelo LMP. Os parâmetros incluídos na solicitação de QoS do LMP estão limitados ao tempo máximo entre *pollings* consecutivos do *master* ao *slave* e ao número de repetições de *broadcast*.

5.6 L2CAP e o Host Controller buffer

O L2CAP e o Baseband trocam pacotes via *Host Controller Interface*. Quando o L2CAP envia um pacote (dois bytes indicando o tamanho, dois bytes indicando o canal a ser usado e 0 a 65.535 bytes de informação), ele chega ao *Baseband* onde é armazenado para transmissão no *buffer* do *Host Controller Interface*. Este *buffer* implementa um esquema de fila do tipo FIFO.

O L2CAP têm um mecanismo de controle de fluxo com o *Baseband* para evitar sobrecarga no *buffer* do *Host Controller*. Há também um mecanismo opcional de controle no sentido inverso, do *Baseband* para o L2CAP para evitar o sobrecarga dos *buffers* do L2CAP.

O nível L2CAP envia um comando *Read_Buffer_Size* ao *Baseband* antes da transferência de pacotes. O comando retorna separadamente para cada tipo de canal, ACL e SCO, o tamanho máximo permitido de pacotes de dados e o número máximo de pacotes de dados que podem ser armazenados no *buffer* do *Host Controller*. Não podem ser enviados pacotes com tamanhos maiores que o máximo permitido e o tamanho

mínimo de pacote suportado pelo *Host Controller Interface* é de 255 bytes de informação.

O evento *Number_Of_Completed_Packets* é ativado pelo *Baseband* e informa ao L2CAP a quantidade de pacotes que foram completamente transmitidos ou descartados via *Flush*. O *Baseband* deve manter o L2CAP informado enquanto houver pacotes no *buffer* do *Host Controller*, mas este é um processo dependente de implementação.

O evento *Data_Buffer_Overflow* informa ao L2CAP a ocorrência de transbordo (*overflow*) no *buffer* do *Host Controller*. O comando *Set_Host_Controller_To_Host_Flow_Control* ativa ou desativa um mecanismo opcional para controle do fluxo do *Host Controller* para o L2CAP. O *Baseband*, através do comando *Host_Buffer_Size*, obtém a informação do tamanho máximo dos pacotes de dados e do tamanho do *buffer* de recepção do L2CAP. O L2CAP informa ao *Baseband*, via evento *Host_Number_Of_Compeleted_Packets*, o nível de ocupação do *buffer* de recepção.

Quando um pacote de dados do *Host Controller Interface* é maior que o formato de pacote *Baseband*, o pacote é segmentado em múltiplos pacotes *Baseband*. O *Link Manager Protoco*l é responsável pela segmentação e reagrupamento dos pacotes *Baseband* em pacotes para o *Host Controller Interface*.

5.7 As deficiências de QoS no Bluetooth

O Bluetooth na versão 1.0 apresenta uma série de deficiências quanto ao suporte de Qualidade de Serviços [34]. A lista das deficiências inclui:

- Regra de transmissão seqüencial L2CAP – A transmissão de um pacote L2CAP deve completar antes de iniciar a transmissão de outro pacote. Se forem considerados um ambiente com múltiplas aplicações e diferentes requisitos de QoS executando sobre o mesmo dispositivo Bluetooth, esta regra pode trazer problemas para as aplicações que requisitarem garantias de atrasos.
- Serviço de suporte QoS em canais ACL – O serviço padrão em canais ACL é melhor esforço sem garantias de atrasos ou de banda.
- Filtragem de retransmissões – o mecanismo empregado não é a prova de falhas, fazendo com que o receptor possa interpretar um novo pacote como uma duplicata ou descartar um pacote sob certas condições.

- Seleção de tipo de pacote – Dependendo das condições dinâmicas da interface de rádio pode se tornar difícil implementar um algoritmo de seleção otimizado.
- Garantias de atrasos em canais ACL – Há vários problemas potenciais em prover garantias de banda e atrasos para serviços ACL.
- Controle de fluxo – Em casos de congestionamento de canal ACL o mecanismo de controle de fluxo pode parar todo o tráfego passante por não estar provido de mecanismos que parem somente o fluxo de tráfego ou a aplicação que está causando o congestionamento.
- Atrasos de setup – Devido à ausência de de um canal de controle comum, os procedimentos de *Inquiry* e *Paging* são relativamente longos no Bluetooth.
- Controle de tráfego – Quando prioridades são introduzidas em *scatternets*, o tráfego priorizado sobre um canal congestionado pode subjugar o tráfego melhor esforço (*best effort*) deste canal.

5.7.1 O Problema do Controle de Fluxo

O controle do fluxo apresenta duas situações problemáticas: o congestionamento do *buffer* do *Host Controller* e o controle do fluxo sobre múltiplas interfaces, este último considerando o escopo de *scatternets.*

5.7.1.1 Congestionamento do *buffer* do *Host Controller*

Um dispositivo Bluetooth que hospede várias aplicações apresenta na sua arquitetura um ponto de convergência de tráfego, o *buffer* do *Host Controller Interface*. Todos pacotes gerados por aplicações, independentemente dos requisitos de qualidade de serviços requeridos fluem para este *buffer*. Este *buffer* implementa o mecanismo de fila FIFO (ver seção 4.3.1).

O problema ocorre quando os pacotes com requisitos de QoS não conseguem espaço na fila do *buffer*, sobrecarregado com tráfego melhor esforço gerado por aplicações sem requisitos de qualidade de serviços.

A figura 5.1 [34] representa o *buffer* do *Host Controller* totalmente tomado por tráfego melhor esforço num canal ACL. O tráfego melhor esforço (ACL BE) está representado no lado direito da figura. Este é o tráfego sem requisitos de qualidade de

serviço. O tráfego à esquerda da figura, ACL QoS, representa o tráfego com requisitos de qualidade de serviço.

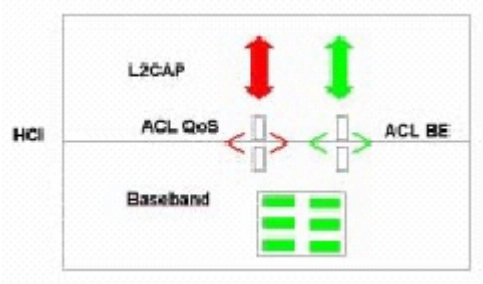

Figura 5.1 – Bloqueio de *buffer* HCI

5.7.1.2 Controle do fluxo sobre múltiplas interfaces

Nesta seção será apresentado o problema quando aplicado numa *scatternet*, onde o fluxo precisa ser controlado sobre múltiplas interfaces sem fio [34].

Numa *piconet* a comunicação é direta entre o *master* e o *slave* (ver seção 3.6 e seguintes). Transmissões entre *slaves* devem obrigatoriamente passar pelo *master*, fazendo aumentar em pelo menos dois saltos a comunicação numa *piconet*, o que envolve todas as interfaces dos dispositivos envolvidos.

Nas *scatternets* não há limites quanto ao número de saltos. Porém, o problema surge quando houver várias aplicações numa *piconet* (um ou mais *slaves*) gerando pacotes destinados a dispositivos de outra piconet.

Neste caso, o *buffer* do dispositivo master da *piconet* onde os pacotes estão sendo gerados pode ficar sobrecarregado no envio dos pacotes para o *slave* que atua como *gateway* entre as *piconets*.

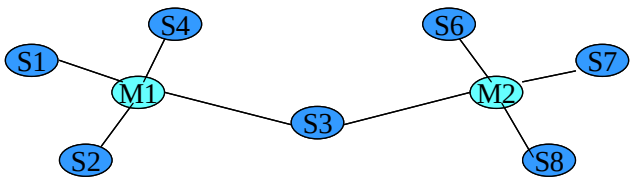

Figura 5.2 – Congestionamento em *scatternet*

A figura 5.2 [34], ilustra o cenário para o problema exposto. Tomando-a como referência para melhor entendimento da situação, consideremos todos canais de comunicação do tipo ACL e todos com 100 kbps de largura de banda.

Os *slaves* S1 e S2 hospedam diferentes aplicações que geram tráfegos na taxa 100 kbps e 150 kbps, respectivamente. O destino do tráfego de S1 é S7 e o destino do tráfego de S2 é S6.

O *buffer* do nó M1 pode ficar congestionado considerando somente este tráfego. Porém, o *buffer* do nó S3 pode ficar congestionado com o tráfego enviado pelo nó M1 e por uma aplicação que esteja ativa e gerando tráfego. Este nó também tem seu desempenho afetado pelo processo de chaveamento entre as duas *piconets*. Já o nó M2 pode ficar com o *buffer* congestionado pelo tráfego gerado pelos *slaves* de sua *piconet*, ou por uma aplicação ativa neste nó, e também por tráfego enviado pelo nó S3.

5.7.1.3 Por que isto é um problema

O Bluetooth tem um mecanismo de controle de fluxo *Stop-and-Wait* no canal ACL. Este mecanismo interrompe toda a transferência de dados ACL. Há também o mecanismo obrigatório de controle de fluxo na direção *host* (L2CAP) para *Host Controller* e o mecanismo opcional na direção *Host Controller* para *host* (L2CAP), conforme figura 5.3 [34].

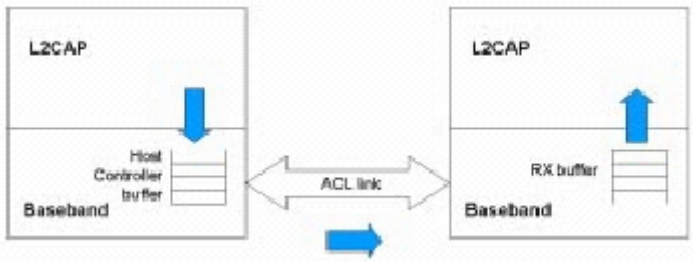

Figura 5.3 – Controle de fluxo do Bluetooth

Estes três mecanismos exercem pressão no sentido contrário ao tráfego quando o mesmo não pode ser enviado adiante no canal congestionado. O nível L2CAP verifica a situação do *buffer* e encontrando-o indisponível não pode enviar seus pacotes, independente se tiverem ou não requisitos de qualidade de serviços. A pressão por recursos é computada para a aplicação e seu tratamento passível de implementação.

Conforme descritos na seção 4.2.1, os mecanismos que integram a rede devem ser configurados com informações que permitam a classificação do tráfego em diferentes filas, além de algoritmos que tratem o tráfego separadaradamente nas filas. O mecanismo de fila implantado no *buffer* do *Host Controller Interface* é do tipo FIFO, que é limitado para a implantação de qualidade de serviços - QoS (ver seção 4.3.1).

Qualidade de serviço e largura de banda estão diretamente relacionados. Mecanismos de qualidade de serviços não criam banda, não ampliam a capacidade de banda. Mecanismos de qualidade de serviço aumentam a eficiência dos recursos utilizados.

5.8 Conclusões do capítulo

O *buffer* da interface do *Host Controller* é o ponto de convergência do tráfego a ser transmitido entre módulos Bluetooth. Diversas aplicações podem gerar diferentes tipos de tráfego sobre canais ACL. Este tráfego pode ser melhor esforço ou com requisitos de qualidade de serviços, mas se o primeiro for gerado em taxas superiores, o *buffer* da interface pode ficar totalmente carregado com pacotes melhor esforço.

Empregando esquema FIFO, a interface não tem como gerenciar o conteúdo da fila. A sequência deste trabalho apresenta alternativas para esta situação.

6. PLATAFORMA DE TESTES

6.1 Introdução

A ferramenta usada neste trabalho foi o *software* Network Simulator – versão 2.1b7 (ns-2), de outubro de 2000, que é parte do projeto VINT (*Virtual Internetwork Testbed*), desenvolvido pelo ICS (*Information Sciences Institute*) da *University of Southern Califórnia*. Tem sido uma ferramenta usada por grande número de pesquisadores para simularem ambientes de redes com e/ou sem fios.

O objetivo deste capítulo é verificar e validar o processo de análise usado sobre os arquivos de *trace* gerados pelo simulador ns-2. Na simulação de uma ambiente com maior número de elementos e tempo maior de exercício, os arquivos de *trace* são gerados na ordem de milhares de linhas.

Devido ao tamanho destes arquivos será definido um cenário simplificado, com uma fila entre dois pontos, geração determinística de tráfego e com intervalo de tempo reduzido. Este modelo será exercitado com o simulador ns-2 e também com o simulador Arena. Os resultados do processo de análise sobre os arquivos de *trace* do ns-2 serão comparados com os resultados apresentados nos relatórios gerados pelo Arena.

Adotar uma fila determinística facilita a interpretação e julgamento dos valores em qualquer ponto do processo. Desta forma, ao focar qualquer ponto do arquivo *trace* será possível perceber o comportamento do sistema naquele instante. Manter o processo num tempo reduzido facilitará a visualização do processo durante toda a execução.

6.2 A ferramenta ns-2

Este simulador permite adaptações em C++, Java além de permitir scripts escritos numa linguagem própria chamada Otcl (*Object tool command language*), que é a interface de configuração e de comandos para o simulador.

O simulador ns-2 é orientado a objetos, conduzido por eventos discretos, e simula uma variedade de redes IP. São implementados protocolos como TCP e UDP, tráfegos como FTP, Telnet, *Constant Bit Rate* (CBR) e *Variable Bit Rate* (VBR), mecanismos de gerenciamento de filas de roteadores como *Drop Tail* (FIFO), *Random Early Detection* (RED) e *Class Based Queueing* (CBQ), algoritmos de roteamento como Dijkstra, além de outras implementações [48].

O simulador ns-2 é acompanhado de uma ferramenta chamada de *Network Animator* (NAM). Com esta ferramenta é possível visualizar o diagrama da rede durante as animações, acrescentar comentários, avançar ou retroceder na visualização do comportamento do modelo.

6.3 Validação e entendimento do simulador ns-2

O primeiro passo foi escrever um *script* em linguagem tcl, o qual seria interpretado pelo simulador ns-2. Neste trabalho foi utilizada uma versão compilada para operar em plataforma Intel, com Microsoft Windows 2000.

O programa da figura 6.1 é um *script*, que gera um arquivo *trace* (figura 6.2). As características deste modelo são:

- Tráfego de pacotes com tamanho de 2 bytes cada;
- Intervalo de 1 segundo entre cada pacote, iniciando em 0 segundo e terminando em 20 segundos do período da simulação – caracterizando um processo de geração de carga determinístico;
- Canal de comunicação de 8 bits/segundo (1 Byte/segundo);
- Atraso de 1 (um) segundo, para facilitar os cálculos com a variável “Tempo”;
- Fila do tipo FIFO, com limite de 50 pacotes enfileirados;
- Tempo total de 50 segundos de simulação, garantindo no modelo a aplicação da teoria do Fluxo de Equilíbrio, onde o estado da fila ao final do processo é igual ao estado da fila no início do processo.

O arquivo *trace* gerado pela execução do programa foi importado para uma planilha Excel, selecionando somente algumas colunas. A planilha utilizada para os cálculos está nas figuras 6.3 e 6.4.

Das 12 colunas geradas no arquivo *trace*, somente a primeira, a segunda, a terceria, a quarta, a sexta e a décima-segunda coluna foram importadas. A primeira é a coluna das ações ocorridas na simulação. O sinal "+" significa entrada em fila ou sistema, "-" significa a retirada da fila, "d" representa descarte de pacote e "r" significa o recebimento do pacote. Neste protótipo não houve registro de "d". Esta coluna recebeu o nome de Op quando importada para a planilha.

A segunda coluna é do instante de tempo em que a ação ocorreu. Na planilha foi inserida uma coluna adicional, chamada T Aux. O objetivo foi eliminar a ocorrência de tempo zero, necessário para os cálculos de tempo de resposta, cobrindo intervalo de tempo, não somente os valores absolutos.

```
# INICIO
        set ns [new Simulator]
        set file_trace [open trace_file.tr w]
        $ns trace-all $file_trace
        set nam_file [open nam_file.nam w]
        $ns namtrace-all $nam_file
        proc finish {} {
                global ns file_trace
                $ns flush-trace
                close $file_trace
                exec nam nam_file.nam &
                exit 0     }
#
        for {set i 0} {$i <= 1} {incr i} {
                set n($i) [$ns node]          }
#
        $ns simplex-link $n(0) $n(1) 8b 1s DropTail
#
        set null0 [new Agent/Null]
        $ns attach-agent $n(1) $null0
#
        set cbr1 [new Agent/CBR]
        $ns attach-agent $n(0) $cbr1
        $cbr1 set packetSize_ 2
        $cbr1 set interval_ 1s
#
        $ns connect $cbr1 $null0
        $ns queue-limit $n(0) $n(1) 50
#
        $ns at 0.0 "$cbr1 start"
        $ns at 20.0 "$cbr1 stop"
        $ns at 50.0 "finish"
        $ns run
# FIM#
```

Figura 6.1 – *Script* em linguagem tcl

A terceira e quarta colunas são equivalentes às colunas Org e Dst, significando o nó de origem e destino, respectivamente. Não tem qualquer importância neste protótipo. A quinta coluna é do tamanho do pacote, 2 bytes neste protótipo, e a última coluna traz a identificação do pacote no processo de simulação.

```
+ 0 0 1 cbr 2 ------- 0 0.0 1.0 0 0
- 0 0 1 cbr 2 ------- 0 0.0 1.0 0 0
+ 1 0 1 cbr 2 ------- 0 0.0 1.0 1 1
- 2 0 1 cbr 2 ------- 0 0.0 1.0 1 1
+ 2 0 1 cbr 2 ------- 0 0.0 1.0 2 2
r 3 0 1 cbr 2 ------- 0 0.0 1.0 0 0
+ 3 0 1 cbr 2 ------- 0 0.0 1.0 3 3
- 4 0 1 cbr 2 ------- 0 0.0 1.0 2 2
+ 4 0 1 cbr 2 ------- 0 0.0 1.0 4 4
r 5 0 1 cbr 2 ------- 0 0.0 1.0 1 1
+ 5 0 1 cbr 2 ------- 0 0.0 1.0 5 5
- 6 0 1 cbr 2 ------- 0 0.0 1.0 3 3
+ 6 0 1 cbr 2 ------- 0 0.0 1.0 6 6
r 7 0 1 cbr 2 ------- 0 0.0 1.0 2 2
+ 7 0 1 cbr 2 ------- 0 0.0 1.0 7 7
- 8 0 1 cbr 2 ------- 0 0.0 1.0 4 4
+ 8 0 1 cbr 2 ------- 0 0.0 1.0 8 8
r 9 0 1 cbr 2 ------- 0 0.0 1.0 3 3
+ 9 0 1 cbr 2 ------- 0 0.0 1.0 9 9
- 10 0 1 cbr 2 ------- 0 0.0 1.0 5 5
+ 10 0 1 cbr 2 ------- 0 0.0 1.0 10 10
r 11 0 1 cbr 2 ------- 0 0.0 1.0 4 4
+ 11 0 1 cbr 2 ------- 0 0.0 1.0 11 11
- 12 0 1 cbr 2 ------- 0 0.0 1.0 6 6
+ 12 0 1 cbr 2 ------- 0 0.0 1.0 12 12
r 13 0 1 cbr 2 ------- 0 0.0 1.0 5 5
+ 13 0 1 cbr 2 ------- 0 0.0 1.0 13 13
- 14 0 1 cbr 2 ------- 0 0.0 1.0 7 7
+ 14 0 1 cbr 2 ------- 0 0.0 1.0 14 14
r 15 0 1 cbr 2 ------- 0 0.0 1.0 6 6
+ 15 0 1 cbr 2 ------- 0 0.0 1.0 15 15
- 16 0 1 cbr 2 ------- 0 0.0 1.0 8 8
+ 16 0 1 cbr 2 ------- 0 0.0 1.0 16 16
r 17 0 1 cbr 2 ------- 0 0.0 1.0 7 7
+ 17 0 1 cbr 2 ------- 0 0.0 1.0 17 17
- 18 0 1 cbr 2 ------- 0 0.0 1.0 9 9
+ 18 0 1 cbr 2 ------- 0 0.0 1.0 18 18
r 19 0 1 cbr 2 ------- 0 0.0 1.0 8 8
+ 19 0 1 cbr 2 ------- 0 0.0 1.0 19 19
- 20 0 1 cbr 2 ------- 0 0.0 1.0 10 10
r 21 0 1 cbr 2 ------- 0 0.0 1.0 9 9
- 22 0 1 cbr 2 ------- 0 0.0 1.0 11 11
r 23 0 1 cbr 2 ------- 0 0.0 1.0 10 10
- 24 0 1 cbr 2 ------- 0 0.0 1.0 12 12
r 25 0 1 cbr 2 ------- 0 0.0 1.0 11 11
- 26 0 1 cbr 2 ------- 0 0.0 1.0 13 13
r 27 0 1 cbr 2 ------- 0 0.0 1.0 12 12
- 28 0 1 cbr 2 ------- 0 0.0 1.0 14 14
r 29 0 1 cbr 2 ------- 0 0.0 1.0 13 13
- 30 0 1 cbr 2 ------- 0 0.0 1.0 15 15
r 31 0 1 cbr 2 ------- 0 0.0 1.0 14 14
- 32 0 1 cbr 2 ------- 0 0.0 1.0 16 16
r 33 0 1 cbr 2 ------- 0 0.0 1.0 15 15
- 34 0 1 cbr 2 ------- 0 0.0 1.0 17 17
r 35 0 1 cbr 2 ------- 0 0.0 1.0 16 16
- 36 0 1 cbr 2 ------- 0 0.0 1.0 18 18
r 37 0 1 cbr 2 ------- 0 0.0 1.0 17 17
- 38 0 1 cbr 2 ------- 0 0.0 1.0 19 19
r 39 0 1 cbr 2 ------- 0 0.0 1.0 18 18
r 41 0 1 cbr 2 ------- 0 0.0 1.0 19 19
```

Figura 6.2 – Arquivo *trace*

#	Op.	T Aux	T Reg	Org	Dst	Pkt Sz	Pkt Id	Tempo entre chegadas	Tempo de resposta	Tempo de espera	S erviço por pkt	# Pkts no sistema	# Pkts na fila	# Pkts em serviço
	+	1	0	0	1	2	0							
	-	1	0	0	1	2	0			0				
T01												1	0	1
	+	2	1	0	1	2	1	1						
T02												2	1	1
	-	3	2	0	1	2	1			1				
	+	3	2	0	1	2	2	1						
T03												3	1	2
	r	4	3	0	1	2	0		2		2			
	+	4	3	0	1	2	3	1						
T04												3	2	1
	-	5	4	0	1	2	2			2				
	+	5	4	0	1	2	4	1						
T05												4	2	2
	r	6	5	0	1	2	1		3		2			
	+	6	5	0	1	2	5	1						
T06												4	3	1
	-	7	6	0	1	2	3			3				
	+	7	6	0	1	2	6	1						
T07												5	3	2
	r	8	7	0	1	2	2		4		2			
	+	8	7	0	1	2	7	1						
T08												5	4	1
	-	9	8	0	1	2	4			4				
	+	9	8	0	1	2	8	1						
T09												6	4	2
	r	10	9	0	1	2	3		5		2			
	+	10	9	0	1	2	9	1						
T10												6	5	1
	-	11	10	0	1	2	5			5				
	+	11	10	0	1	2	10	1						
T11												7	5	2
	r	12	11	0	1	2	4		6		2			
	+	12	11	0	1	2	11	1						
T12												7	6	1
	-	13	12	0	1	2	6			6				
	+	13	12	0	1	2	12	1						
T13												8	6	2
	r	14	13	0	1	2	5		7		2			
	+	14	13	0	1	2	13	1						
T14												8	7	1
	-	15	14	0	1	2	7			7				
	+	15	14	0	1	2	14	1						
T15												9	7	2
	r	16	15	0	1	2	6		8		2			
	+	16	15	0	1	2	15	1						
T16												9	8	1
	-	17	16	0	1	2	8			8				
	+	17	16	0	1	2	16	1						
T17												10	8	2

Figura 6.3 – Planilha do *trace* (ns-2) – Parte 1

#	Op.	T Aux	T Reg	Org	Dst	Pkt Sz	Pkt Id	Tempo entre chegadas	Tempo de resposta	Tempo de espera	S erviço por pkt	# Pkts no sistema	# Pkts na fila	# Pkts em serviço
	r	18	17	0	1	2	7		9		2			
	+	18	17	0	1	2	17	1						
T18												10	9	1
	-	19	18	0	1	2	9			9				
	+	19	18	0	1	2	18	1						
T19												11	9	2
	r	20	19	0	1	2	8		10		2			
	+	20	19	0	1	2	19	1						
T20												11	10	1
	-	21	20	0	1	2	10			10				
T21												11	9	2
	r	22	21	0	1	2	9		11		2			
T22												10	9	1
	-	23	22	0	1	2	11			11				
T23												10	8	2
	r	24	23	0	1	2	10		12		2			
T24												9	8	1
	-	25	24	0	1	2	12			12				
T25												9	7	2
	r	26	25	0	1	2	11		13		2			
T26												8	7	1
	-	27	26	0	1	2	13			13				
T27												8	6	2
	r	28	27	0	1	2	12		14		2			
T28												7	6	1
	-	29	28	0	1	2	14			14				
T29												7	5	2
	r	30	29	0	1	2	13		15		2			
T30												6	5	1
	-	31	30	0	1	2	15			15				
T31												6	4	2
	r	32	31	0	1	2	14		16		2			
T32												5	4	1
	-	33	32	0	1	2	16			16				
T33												5	3	2
	r	34	33	0	1	2	15		17		2			
T34												4	3	1
	-	35	34	0	1	2	17			17				
T35												4	2	2
	r	36	35	0	1	2	16		18		2			
T36												3	2	1
	-	37	36	0	1	2	18			18				
T37												3	1	2
	r	38	37	0	1	2	17		19		2			
T38												2	1	1
	-	39	38	0	1	2	19			19				
T39												2	0	2
	r	40	39	0	1	2	18		20		2			
T40												1	0	1
	r	42	41	0	1	2	19		21		2			
T41												0	0	0

Figura 6.4 – Planilha do *trace* (ns-2) – Parte 2

Os cálculos foram realizados da seguinte forma:

- Tempo entre chegadas – na ordem cronológica, a diferença entre os tempos das ações “+”;
- Tempo de resposta – tomando sempre identificadores iguais dos pacotes, foi subtraído o tempo da linha com ação “+” do tempo da linha “r”;
- Tempo de espera – para os mesmos identificadores de pacotes, foi subtraído o tempo da linha com ação “+” do tempo da linha “-”;
- Serviço por pacote – para os mesmos identificadores de pacotes, foi subtraído o tempo da linha com ação “-” do tempo da linha “r”;
- # de pacotes no sistema – a soma dos campos seguintes;
- # de pacotes na fila – na ordem cronológica, somada uma unidade para cada “+” encontrado, e reduzida uma unidade para cada “-” encontrado;
- # de pacotes em serviço – na ordem cronológica, somada uma unidade para cada “-” encontrado, e reduzida uma unidade para cada “r” encontrado.

6.4 Resultados da prototipagem com simulador ns-2

As quantidades medidas e apresentadas na planilha das figuras 6.3 e 6.4 são resultantes do processo de importação do arquivo de trace da simulação para o gerenciador de planilhas eletrônicas Microsoft Excel. O estudo da análise operacional está baseado no relacionamento destas quantidades durante o intervalo de medição [0,T], sendo T = 41 segundos.

Não faz parte deste trabalho explicar a teoria das filas, sendo então, apresentados diretamente os resultados das análises deste modelo. Informações sobre análise operacional e teoria das filas são encontradas em inúmeras bibliografias.

Um dos teoremas mais comuns é a Lei de Little [49], que permite relacionar o número médio de pacotes do sistema com o tempo médio dispendido no sistema, como segue:

Tempo médio no sistema = taxa de chegada X tempo médio de resposta

Esta lei é válida enquanto há equilíbrio de fluxo, onde o número de pacotes entrando no sistema é igual ao número de pacotes completando sua existência no sistema, de tal forma que não há novos pacotes sendo criados nem pacotes perdidos infinitamente no sistema. O sistema é válido mesmo quando há perda de pacotes por

limitação de espaço nas filas, onde a lei pode ser aplicada na parte que consiste da espera, pois, encontrando uma posição na fila, não é perdido.

A simulação foi realizada com base num modelo de fila simples, com canal único, taxa de chegadas e de serviços constantes e determinísticas. Para os primeiros 20 segundos, a taxa de chegadas é de um pacote por segundo (λ = 1 pacote/segundo). Para os 20 segundos finais (T=[21, 40]), a taxa de chegadas é zero (λ = 0 pacote/segundo). O sistema pode atender somente um pacote por vez. Não houve qualquer atividade para a simulação no intervalo [41, 50].

Todos pacotes precisam de dois segundos para ser transmitidos. Portanto, a taxa de serviço é 0,5 segundo por pacote (μ = 0,5 segundo/pacote). A partir das planilhas nas figuras 6.3 e 6.4, outros indicadores podem ser calculados e deduzidos:

a) Tempo médio de resposta : 11,5 segundos;
b) Tempo máximo de resposta: 21 segundos;
c) Tempo mínimo de resposta: 2 segundos;
d) Somatório do tempo de resposta: 230 segundos;
e) Tempo médio de espera: 9,5 segundos;
f) Tempo máximo de espera: 19 segundos;
g) Tempo mínimo de espera: 0 (zero) segundos;
h) Somatório do tempo de espera: 190 segundos;
i) Número médio de pacotes na fila: 3,8 pacotes (*);
j) Número máximo de pacotes na fila: 10 pacotes
k) Número mínimo de pacotes na fila: 0 (zero) pacote;
l) Tempo de utilização do canal: 40 segundos (20 pacotes * 2 segundos).

(*) O número médio de pacotes é obtido pelo somatório dos resultados da multiplicação dos tempos, em segundos, pela quantidade de pacotes enfileirados. Neste caso observou-se que houve um pacote na fila por 10 segundos, 04 pacotes em fila por um segundo, e assim, sucessivamente, quatro pacotes em fila por dois até nove segundos (10*1 + 4 * 1 + ...+ 4 * 9). Esta equação resulta em 190, que dividido pelo tempo da simulação (50 segundos) resulta em 3,8 pacotes em média na fila.

6.5 Resultados da prototipagem com simulador Arena

Os mesmos parâmetros definidos para o ns-2 foram utilizados para simular uma fila FIFO com o simulador Arena, versão 4.0, disponível em versão acadêmica [50].

Na figura 6.5 está a descrição do processo básico chamado *Create*, responsável pela definição da carga a ser aplicada ao modelo, tipo constante, com intervalo de um segundo entre as entradas, gerando no máximo 20 entradas, iniciando no instante 0 da simulação.

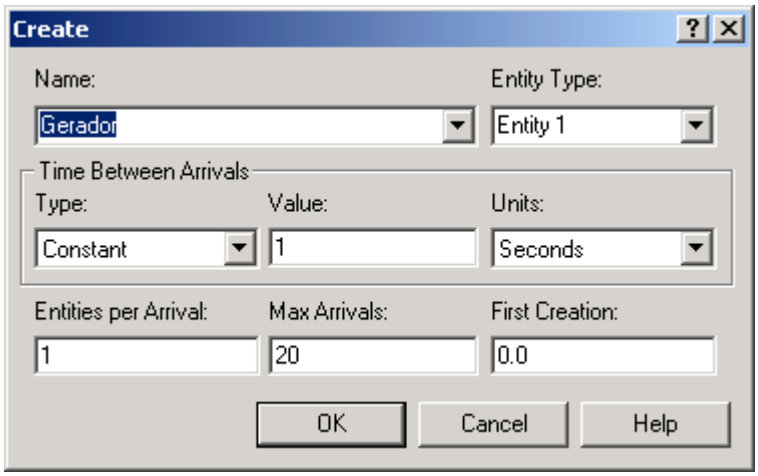

Figura 6.5 – Arena – Processo básico *Create*

O processo básico chamado *Process* desempenha a função do canal de comunicação, configurando uma taxa de serviços de um pacote em dois segundos.

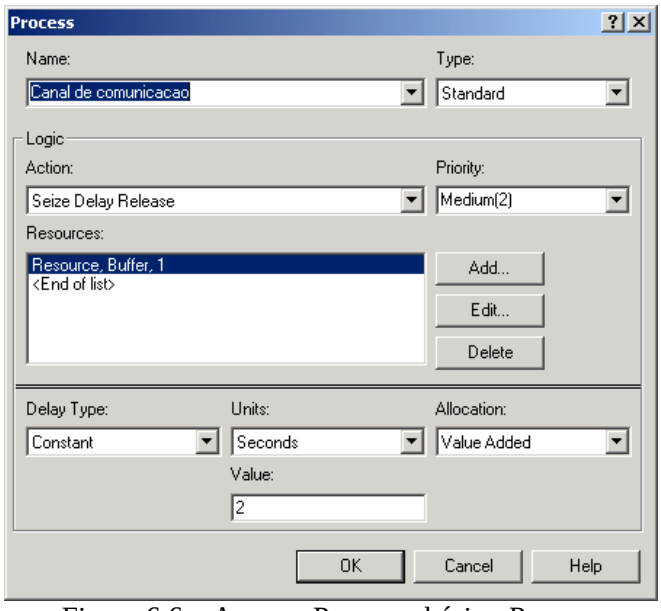

Figura 6.6 – Arena – Processo básico *Process*

Na figura 6.7 estão os parâmetros da execução. Uma replicação com duração de 50 segundos, tempo suficiente para garantir a entrega de todos os pacotes entrantes no sistema.

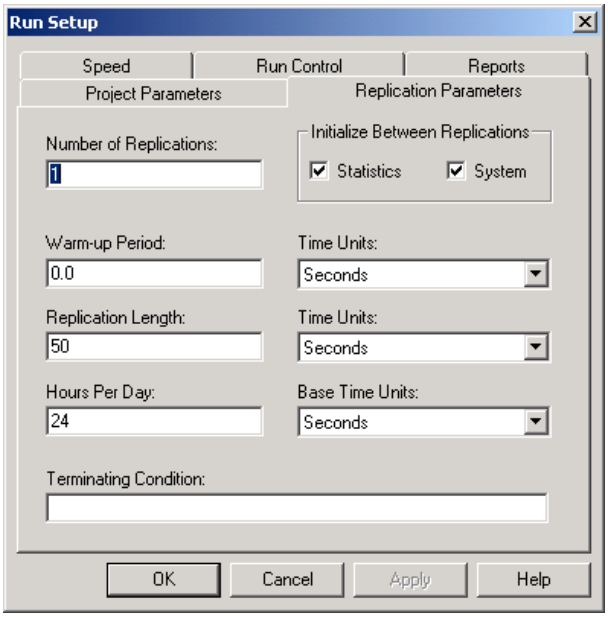

Figura 6.7 – Arena – Parâmetros de execução

As figuras 6.8 e 6.9 apresentam dois instantes da simulação. O primeiro caracterisa a situação da fila quando decorridos 20 segundos de simulação, e o segundo quando a execução da simulação já está finalizada.

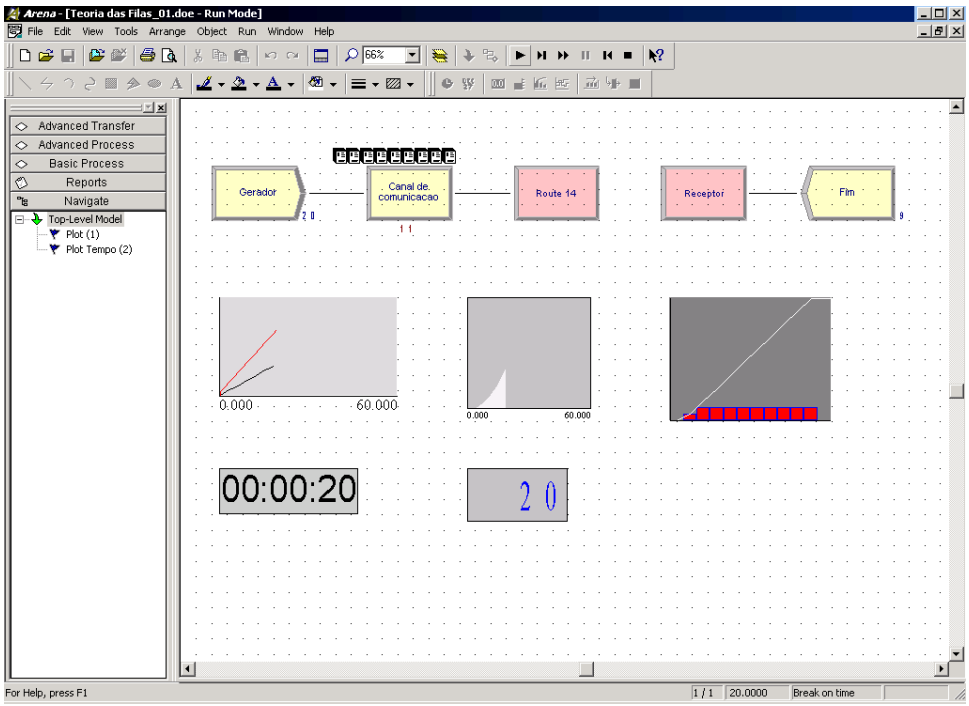

Figura 6.8 – Arena – Simulação – Ponto máximo de enfileiramento

Aos 20 segundos de simulação ocorre o enfileiramento máximo de 10 pacotes. Neste instante há 11 pacotes presentes no sistema, 10 enfileirados e um sendo transmitido. É equivalente ao instante T20 da figura 6.4. Há na figura 6.8 um temporizador marcando 20 segundos; um contador de pacotes marcando 20; um gráfico de linhas, onde a linha vermelha representa os pacotes entrantes e a linha preta os pacotes atendidos. O gráfico de área representa o tempo de utilização do canal de comunicação e o gráfico da direita mostra, com barras, o número de pacotes entrantes por instante de tempo e, em formato de linhas, o mesmo ítem com os valores acumulados.

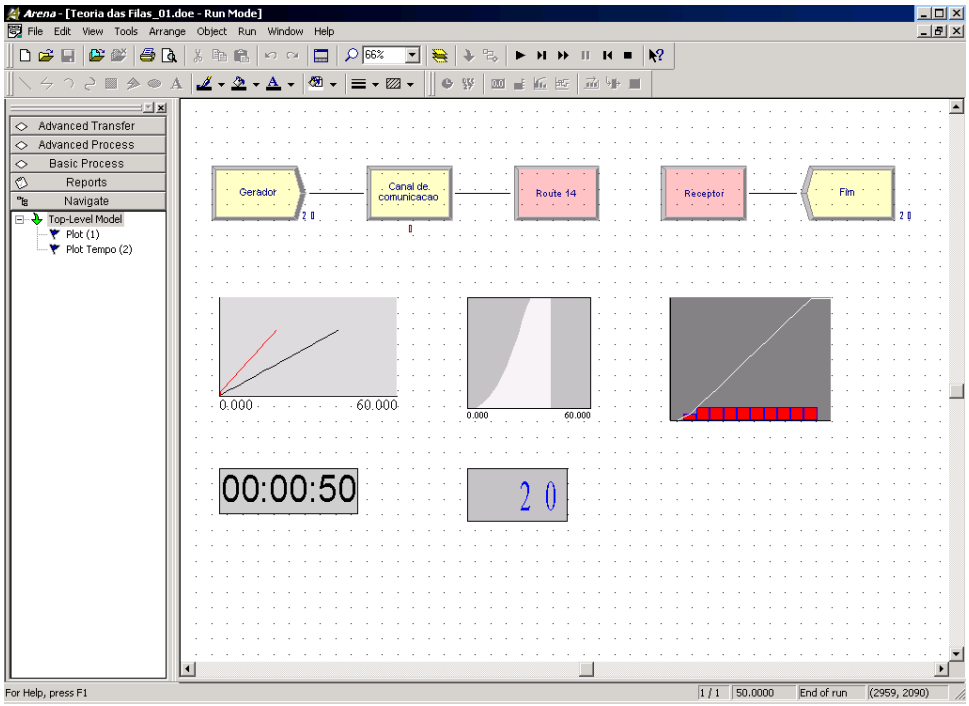

Figura 6.9 – Arena – Simulação – Fim da simulação

Aos 50 segundos da simulação pode-se perceber a mudança no gráfico de linhas dos pacotes de entrada e saída, a área representando o tempo de utilização do canal de comunicação e o comportamento das entradas.

Os gráficos foram incluídos para fins de ilustração e das capacidades que podem ser obtidas com o simulador Arena. Os resultados desta simulação estão demonstrados na sequência através dos relatórios gerados.

Simulação Teoria Replications: 1

Replication 1

Start Time: 0,00 Stop Time: 50,00 Time Units: Seconds

Entity Detail Summary

Time

	01	02	04	05	06	07
	VA Time	NVA Time	Transfer Time	Wait Time	Other Time	Total Time
ENTITY 1	2.00	0.00	0.00	9.50	0.00	11.50
Total	2.00	0.00	0.00	9.50	0.00	11.50

Other

	Number In	Number Out
ENTITY 1	20	20
Total	20	20

ENTITY 1

Time	Average	Half Width	Minimum	Maximum
NVA Time	0.00	(Insufficient)	0.00	0.00
Other Time	0.00	(Insufficient)	0.00	0.00
Total Time	11.5000	(Insufficient)	2.0000	21.0000
Transfer Time	0.00	(Insufficient)	0.00	0.00
VA Time	2.0000	(Insufficient)	2.0000	2.0000
Wait Time	9.5000	(Insufficient)	0.00	19.0000
Other	Value			
Number In	20			
Number Out	20			
WIP	4.6000	(Insufficient)	0.00	11.0000

Figura 6.10 – Arena – Relatório *Entities*

Neste relatório, figura 6.10, pode-se observar a média (*Average*) total, os valores máximos e mínimos obtidos na simulação. Os resultados são os mesmos dos obtidos com os cálculos do arquivo *trace* do simulador ns-2. O tempo médio de resposta é 11,5 segundos, o tempo médio de enfileiramento de 9,5 segundos, com 20 pacotes entrantes e 20 saíntes. Além disso há o número máximo de enfileiramento, 11 pacotes.

Simulação Teoria Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 50,00 Time Units: Seconds

Process Detail Summary

Time

	01	05	07
	VA Time	Wait Time	Total Time
Canal de comunicacao	2.00	9.50	11.50
Total	2.00	9.50	11.50

Accumulated Time

	Accum VA Time	Accum Wait Time
Canal de comunicacao	40.00	190.00
Total	40.00	190.00

Other

	Number In	Number Out
Canal de comunicacao	20.00	20.00
Total	20.00	20.00

Canal de comunicacao

Time	Average	Half Width	Minimum	Maximum
Total Time	11.5000	(Insufficient)	2.0000	21.0000
VA Time	2.0000	(Insufficient)	2.0000	2.0000
Wait Time	9.5000	(Insufficient)	0	19.0000

Accumulated Time	Value
Accum VA Time	40.0000
Accum Wait Time	190.00

Other Value

Figura 6.11 – Arena – Relatório *Processes*

No relatório sobre o processo, figura 6.11, que representa o canal de comunicação obtem-se os mesmos valores já identificados no relatório da figura 6.10. A informação adicional neste relatório é o tempo acumulado de espera de cada pacote, 190 segundos e o tempo no canal de todos os pacotes, 40 segundos. O tempo acumulado pode ser obtido com o ns-2 a partir da soma dos tempos de espera da planilha de apoio, e o tempo no canal é facilmente deduzido pela multiplicação da quantidade de pacotes (20) pelo tempo de uso de canal por cada pacote, 2 segundos.

Simulação Teoria Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 50,00 Time Units: Seconds

Queue Detail Summary

Time

	Waiting Time
Canal de comunicacao.Queue	9.50
Total	9.50

Other

	Number Waiting
Canal de comunicacao.Queue	3.80
Slave_0.Queue	0.00
Total	3.80

Canal de comunicacao.Queue

Time	Average	Half Width	Minimum	Maximum
Waiting Time	9.5000	(Insufficient)	0	19.0000
Other	**Average**	**Half Width**	**Minimum**	**Maximum**
Number Waiting	3.8000	(Insufficient)	0	10.0000

Slave_0.Queue

Other	Average	Half Width	Minimum	Maximum
Number Waiting	0	(Insufficient)	0	0

Figura 6.12 – Arena – Relatório *Queue*

Ao analisar a fila formada, novamente pode-se confirmar os valores obtidos com o simulador ns-2. O número máximo de enfileiramento alcançado foi 10 pacotes, sendo que 19 pacotes passaram pela fila (figura 6.12). Pelos cálculos da simulação com o ns-2 obtivemos o tempo médio de enfileiramento registrado em 9,5 segundos, o que pode ser confirmado agora com o resultado obtido com o simulador Arena.

Simulação Teoria Replications: 1

Replication 1 Start Time: 0,00 Stop Time: 50,00 Time Units: Seconds

Resource Detail Summary

Usage

	Number Busy	Number Scheduled	Utilization
BUFFER	0.80	1.00	0.80

Other

	Number Times Used	Scheduled Utilization
BUFFER	20.00	0.80

BUFFER

Usage	Average	Half Width	Minimum	Maximum
Number Busy	0.8000	(Insufficient)	0	1.0000
Number Scheduled	1.0000	(Insufficient)	1.0000	1.0000
Utilization	0.8000	(Insufficient)	0	1.0000

Other	Value
Number Times Used	20.0000
Scheduled Utilization	0.8000

Figura 6.13 – Arena – Relatório *Buffer*

No relatório da figura 6.13 pode-se verificar que há somente um *buffer*, que foi utilizado 20 vezes e somente um pacote por vez. Isto significou um nível de utilização de 80 %.

6.6 Conclusões do capítulo

Com este protótipo foi possível estabelecer um procedimento para análise dos arquivos trace gerados pelo simulador ns-2. A validação dos valores importados na planilha, a confirmação dos cálculos baseados nas hipóteses e formulações da lei de

Little, e a identificação dos parâmetros que devem ser configurados no simulador foram resultados expressivos deste protótipo.

Com base nestes resultados, ficam garantidos os procedimentos para cálculo de latência e *jitter*, o primeiro pela diferença dos atrasos considerando, na ordem cronológica, as diferenças das ações registradas como “-”. Decorrente deste cáculo poderá ser obtido o *jitter*, pela diferenças das diferenças cálculadas no passo anterior. A perda de pacotes é obtida pela observação das ações registradas como “d”.

As simulações feitas com o simulador Arena reforçaram a validade do método. Com o protótipo do processo empregado no simulador ns-2 e confirmado posteriormente através dos relatórios das simulações usando o Arena, podemos afirmar que o processo está validado.

7. EXPERIMENTOS E RESULTADOS

7.1 Introdução

Foram realizadas várias simulações utilizando o simulador *Network Simulator* 2, também conhecido como ns-2. Cada esquema de fila foi simulado isoladamente, gerando arquivos de *trace* e arquivos para simulações gráficas.

Os arquivos de *trace* foram importados para uma planilha eletrônica. Optou-se por utilizar uma planilha eletrônica pela facilidade em trabalhar com os dados, tratando-os com as mais diferentes combinações e agrupamentos, e pela agilidade para geração de gráficos. Foram geradas planilhas eletrônicas independentes para cada esquema de fila.

A arquitetura do modelo simulado é composta de duas *piconets* interligadas, formando uma *scatternet.* As filas são formadas em dois pontos, no dispositivo *slave*, gerador de tráfego, e no dispositivo *master*, que recebe os dados. O *master* , não sendo o destino final dos pacotes, deve enviá-los pelo caminho que une as duas *piconets*. Neste ponto está o principal ponto de concentração do tráfego da *scatternet* e que é o foco principal das análises deste estudo.

O objetivo deste capítulo é apresentar os resultados das simulações e análises realizadas. Os comentários e descrição das particularidades, dos pontos importantes, das sutilezas percebidas estão agrupados pelos principais tópicos que envolvem qualidade de serviços, citado nos capítulos anteriores, sendo eles, perda de pacotes, latência e *jitter*. Cada tópico está suportado pela apresentação de gráficos e tabelas.

7.2 Arquitetura do modelo simulado

A definição da arquitetura do modelo considera a simplificação do processo de análise, a facilitação da compreensão e a visualização do problema. Com estas considerações, o resultado será o entendimento rápido e objetivo do problema e das alternativas de solução.

Foram então configuradas duas *piconets*, A e B. Cada *piconet* formada por seis *slaves* para um *master*. O sétimo *slave* é comum as duas *piconets*, atuando como *gateway* entre as duas e formando uma *scatternet*.

Entre os seis dispositivos *slaves* de cada *piconet* foi estabelecida uma relação unívoca de gerador de pacotes, na *piconet* A, e de receptor de pacotes, *piconet* B. Todos os *slaves* da *piconet* A devem enviar os pacotes para o seu dispositivo *master*, e este redirecionar os pacotes para o dispositivo *slave* comum as duas *piconets*.

Quando os pacotes estão presentes no dispositivo *gateway*, este deve alternar do *master* da *piconet* A para o *master* da *piconet* B e enviar os dados para este dispositivo. O *master* de B redireciona os pacotes para os dispositivos *slaves* destinatários. Não foi considerado o fator de chaveamento deste dispositivo entre os *masters* das *piconets*.

No modelo simulado não foi considerado o tráfego retorno. A tecnologia Bluetooth permite o tráfego em sentidos opostos simultaneamente. Seu dispositivo de rádio está projetado para enviar e receber dados ao mesmo tempo.

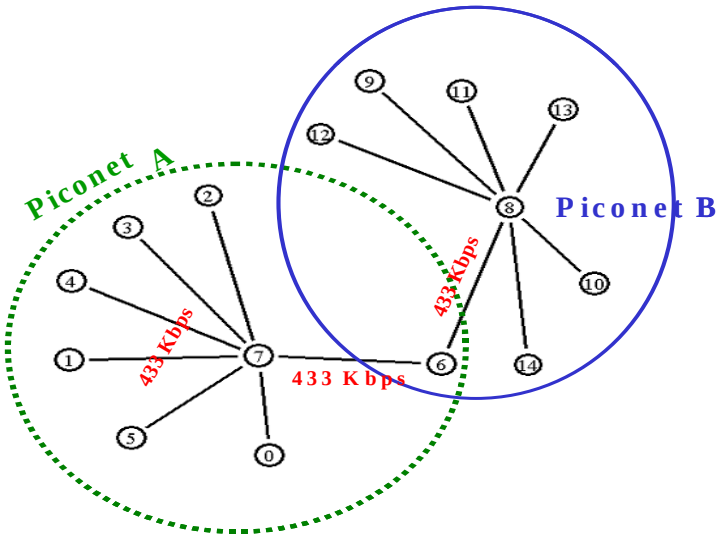

Figura 7.1 – Arquitetura da scatternet

Entretando, uma das características de um modelo *full-duplex* foi mantida, a largura de banda do canal de comunicação, 433 Kbps, atributo de um canal simétrico (o canal assimétrico tem 723 Kbps num sentido e 56,4 Kbps no sentido oposto). O canal configurado entre os *slave*s e o *master* é do tipo assimétrico - ACL. A figura 7.1 descreve graficamente o modelo simulado.

Outra definição do modelo é relativa ao tamanho dos pacotes. O critério adotado para definir os tamanhos dos pacotes é o tempo necessário para transmissão sobre o canal. Desta forma, o menor pacote foi definido para consumir um segundo de tempo de transmissão, e o maior pacote deverá consumir 32 segundos, conforme tabela 7.1.

Este canal teve os seguintes atributos quando configurado no simulador: *simplex*, *bandwidth* de 432 Kbps e retardo (*delay*) de um segundo. Um atraso de um segundo pode não representar a realidade num processo de transmissão de curta distância, porém, facilita sobremaneira o processo de análise, melhorando a visualização do comportamento do modelo. Porém, isto não impede o processo de análise para uma situação com tempos reais. Quando este for o caso, basta substituir o atraso pelo valor real e proceder à simulação.

Nós		Tamanho dos pacotes		Tempo para
Origem	Destino	KBytes	Kbits	transmissão
0	9	54	432	1 seg.
1	10	108	864	2 segs
2	11	216	1.728	4 segs
3	12	432	3.456	8 segs
4	13	864	6.912	16 segs
5	14	1.728	13.824	32 segs

Tabela 7.1 – Tamanhos e tempos de pacotes.

Para definir o tempo considerado como suficiente para a simulação, foi estabelecido o critério de 50 % dos pacotes gerados terem, no mínimo, entrado em processo de transmissão. Considera-se em processo de transmissão os pacotes na *piconet* ou na *scatternet*. Diante disso, o tempo estabelecido e padrão para todos os esquemas simulados ficaram em 500 segundos.

Num intervalo de 30 segundos, cada *slave* gerador de tráfego produziu e enviou 29 pacotes. Neste processo de geração de pacotes, totalmente determinístico, o sistema foi exposto à uma carga gerada de 174 pacotes. Alguns foram perdidos, resultados das dinâmicas dos esquemas de filas, outros permaneceram enfileirados ao término do processo de simulação.

Os mesmos critérios de análise, elaboração de tabelas e apresentação de gráficos foram adotados para todos os esquemas simulados. Pela pouca expressividade das diferenças entre alguns esquemas, e procurando destacar os comportamentos, optou-se por apresentar os resultados isoladamente, com gráficos para cada modelo de fila ou tamanho de pacote.

Seguindo a mesma diretriz para apresentar os resultados, nos gráficos que demonstram o comportamento dos esquemas com diferentes tamanhos de pacotes, a linha dos pacotes perdidos foi multiplicada pelo fator 10. Adotando esta estratégia, o comportamento dos esquemas diante de perdas de pacotes ficou evidenciado. Como sempre havia somente a eliminação de um pacote por vez, a linha nunca assumiu um valor diferente de zero ou 10.

Conforme já citado anteriormente, o processo adotado neste trabalho é totalmente determinístico. Algumas variáveis do processo de geração de pacotes foram arbitradas, como tempo de transferência e taxa de geração de pacotes, instantes iniciais e finais da geração dos fluxos. Desta forma torna-se mais fácil mapear o processo ao longo do tempo. O mapeamento permite prever qual é a situação teórica do sistema em qualquer instante de tempo. Na tabela 7.2 temos a situação teórica quando tiverem decorrido 500 segundos de simulação – tempo definido como final da simulação deste trabalho.

Pacotes (KB)	Tempo da geração do primeiro pacote (segundos)	Tempo da geração do último pacote (segundos)	Duração da transferência de cada pacote (segundos)	Pacotes enviados ao nó master aos 500 segundos.
54	1	29	1	29
108	2	58	2	29
216	4	116	4	29
432	8	232	8	29
865	16	464	16	29
1730	32	928	32	15

Tabela 7.2 – Situação teórica do modelo aos 500 segundos de simulação.

Outro mapeamento que pode ser construído a partir do determinismo do processo é a situação em intervalos de tempos. Pode-se prever quais os fluxos com maiores taxas e presença em diferentes intervalos. A tabela 7.3 apresenta os principais intervalos para esta análise. Teoricamente, a cada intervalo haverá um fluxo dominante. No intervalo seguinte, este fluxo não está mais presente e novo fluxo passa a ter domínio. Domínio, neste caso, significa número maior de pacotes em relação aos demais.

Intervalo de tempo (segundos)	Pacotes em KB enviados para o nó master (6)						Total de pacotes
	54	108	216	432	865	1730	
01 – 30	29	15	7	3	1	0	55
31 – 60	-	14	8	4	2	1	29
61 – 120	-	-	14	7	4	2	27
121 – 240	-	-	-	15	8	4	27
241 - 500	-	-	-	-	14	8	22

Tabela 7.3 – Comportamento teórico do sistema por intervalo de tempo

Analisando o comportamento teórico com base em intervalos de tempos, no primeiro intervalo, de 01 a 30 segundos, o nó *master* deve receber 55 pacotes. Destes, 29 foram pacotes de 54 KB, mais de 50% dos pacotes a serem recebidos. Ao mesmo tempo, estes 29 pacotes representam todos os pacotes de 54 KB gerados no intervalo.

O primeiro intervalo é também o período em que todos os fluxos geram pacotes. Em função dos tempos necessários para transferência, vários pacotes não entraram em processo de transferência para o nó *master*.

O segundo intervalo, teoricamente sem pacotes de 54 KB passa a ter pacotes de 108 KB representando pouco menos de 50% das transferências. Representa também o término da transferência de todos pacotes gerados com este tamanho. E assim sucessivamente para os demais intervalos, variáveis em suas durações, mas representando sempre o término da transferência de um dos fluxos.

Foram escolhidos três esquemas de filas para esta simulação. O pressuposto para seleção dos modelos era pertencerem ao conjunto das técnicas conservativas. Isto é justificado pelo fato do dispositivo *master* do Bluetooth empregar a técnica de *polling* para acessar os dispositivos *slaves*. Sendo uma técnica de *polling*, não poderá ocorrer

perda de “oportunidades” para realizar uma comunicação. Seria improdutivo se um dispositivo não realizasse uma transmissão no momento em que tivesse a oportunidade para fazê-lo, ficando então ocioso e aguardando uma nova oportunidade de transmissão.

7.3 Resultados da simulação

Repetindo o que já foi escrito anteriormente, o Bluetooth implementa o esquema de filas FIFO na interface do *Host Controller*. Os pacotes são retirados da fila para transmissão seguindo a sequência de chegada. O esquema FIFO não implementa qualquer funcionalidade que permita agregar ou assegurar qualidade de serviços. Não são utilizados critérios para seleção ou escalonamento de pacotes.

Utilizando o esquema FIFO foram realizadas duas simulações. A diferença entre ambas está na definição do tamanho máximo da fila. Uma simulação considerou um tamanho máximo de 30 pacotes, e a outra, 50 pacotes. Desta forma pode-se verificar quais são os impactos quando o tamanho da fila é alterado.

A próxima análise simulou o esquema SFQ. Para este esquema adotou-se como limite de enfileiramento o valor padrão, já previamente definido para a versão compilada do simulador. Neste esquema a fila atingia seu limite quando retinha 30 pacotes.

Os esquemas *Fair Queuing* foram propostos para prover condições de isolar fluxos e prover garantias de qualidade de serviço em resposta ao mecanismo FIFO. A idéia básica é servir os fluxos em proporções, independentemente dos volumes gerados pelos fluxos. Algoritmos de FQ procuram liberar o uso da banda disponível e, ao mesmo tempo, assegurar garantias de qualidade de serviços na presença de congestionamentos.

Operando no esquema de FIFO para cada fluxo, e no esquema de *bit-by-bit round-robin*[4], FQ apresenta excelentes resultados. Entretanto, requer um alto custo computacional para tratamento bits transmitidos, tornando-se inviável em comunicações de alta velocidade [30].

Nas simulações realizadas não foi possível identificar comportamentos que caracterizassem a funcionalidade do esquema. Os resultados das simulações foram incluídos no trabalho, embora apresentem praticamente os mesmos valores observados

[4] Bit-by-bit round-robin servem filas de tal maneira que alocam a banda igualmente para cada fila formada pelos fluxos de tráfego.

com FIFO. Há pequenas diferenças na ordem em que alguns pacotes são transferidos, principalmente nas inversões entre pacotes grandes e pequenos em determinados momentos da simulação. Porém, os resultados não foram sensíveis o suficiente para demonstrar comportamentos diferentes em relação à FIFO.

O esquema DRR, configurado nesta versão do simulador para um limite de máximo de 25.000 bytes enfileirados, já possibilitou análises similares às desenvolvidas para o esquema SFQ, devido à evidente diferença na dinâmica dos esquemas.

No anexo F estão as figuras demonstrando graficamente a distribuição das filas obtidas em cada simulação. Os dados de cada simulação estão distribuídos estatisticamente dentro dos critérios de uma distribuição normal. Isto significa que os dados são válidos para a realização da análise, além de, se mantidos os mesmos desempenhos e aumentando o tempo de simulação teremos os mesmos resultados.

7.3.1 Perda de pacotes

As figuras do anexo A representam, em forma gráfica, o comportamento dos esquemas de filas, mostrando linhas de pacotes enfileirados (“enfileiramento”), pacotes em serviço, ou seja, aqueles que estão sendo transmitidos naquele instante (“Em serviço”), e pacotes perdidos (“perdidos*10”).

Quanto aos pacotes “em serviço”, observa-se uma variação na linha cujos valores variam entre um e dois. Esta é a forma que o simulador registra a informação no arquivo de *trace*, marcando o início de transferência de um pacote antes da marcação do final de transferência do pacote que estava em processo de transferência.

7.3.1.1 *First In First Out* (FIFO)

As figuras A1 e A2 representam, respectivamente, os comportamentos das simulações do esquema FIFO com limites máximos de 30 e 50 pacotes. A figura A3 representa o comportamento do esquema SFQ, e as figuras A4 e A5 representam os esquemas FQ e DRR, respectivamente.

Dois comportamentos distintos podem ser observados : a simples rejeição do pacote que está chegando à fila, característica dos esquemas FIFO e FQ; e a seleção e eliminação de pacotes que já estavam presentes na fila, utilizadas pelos esquemas SFQ e DRR.

O comportamento da rejeição de pacotes descarta os pacotes que chegam ao final da fila, independente de tamanho ou fluxo de tráfego. Cada pacote que chegasse à fila

era colocado ao final da mesma. Se a fila excedesse o limite estabelecido, o pacote era descartado.

	First In, First Out (FIFO)							
	Scatternet				***Piconet***			
Tamanho	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**
54	29	7 (24%)	0	22 (76%)	29	0	0	29
108	29	10 (34%)	0	19 (66%)	29	0	0	29
216	29	12 (41%)	0	17 (59%)	29	0	0	29
432	29	11 (38%)	2	16 (55%)	29	0	0	29
865	29	7 (24%)	16	6 (21%)	29	0	0	29
1.730	15	2 (13%)	9	4 (27%)	29	0	14	15
	160	49 (31%)		84 (53%)				

Tabela 7.4 – Resultados da simulação – FIFO – limite: 30 pacotes

Na simulação FIFO com limite de 30 pacotes, há quatro intervalos a serem comentados. O primeiro deles, entre 23 e 34 segundos, há predominância de perda dos pacotes menores (54 bytes). São os pacotes com maior taxa de chegada para este momento.

Entre 50 e 80 segundos concentram-se as perdas dos pacotes de 108 bytes, agora com maior taxa de chegada. O terceiro intervalo, 105-130 segundos, concentra as perdas dos pacotes de 216 bytes, então com a maior taxa de chegada. No quarto intervalo, 216-241 segundos, as perdas concentram-se nos pacotes superiores a 432 bytes, já com uma taxa de chegada mais uniforme. A tabela 7.4 sumariza o comportamento desta simulação. Na simulação não houve perdas de pacotes na *piconet*.

Na análise do arquivo de *trace* da simulação FIFO com limite em 50 pacotes, é marcante a redução na taxa de perda de pacotes. No gráfico relativo a este esquema há dois intervalos a serem comentados. O primeiro, entre 50 e 80 segundos, quando as perdas concentram-se nos pacotes de 108 e 216 bytes. Não há perdas dos pacotes menores (54 bytes) porque todos já foram acomodados na fila. As perdas são equivalentes às taxas de entrada quando a fila está próxima ao seu limite, quanto maior a taxa de entrada de um fluxo, maior taxa de perda de pacotes deste fluxo.

	First In, First Out (FIFO)							
	Scatternet				***Piconet***			
Tamanho	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**
54	29	0 (0%)	0	29 (100%)	29	0	0	29
108	29	4 (14%)	0	25 (86%)	29	0	0	29
216	29	6 (21%)	0	23 (79%)	29	0	0	29
432	29	4 (14%)	12	13 (45%)	29	0	0	29
865	29	4 (14%)	20	5 (17%)	29	0	0	29
1.730	15	0 (0%)	11	4 (27%)	29	0	14	15
	160	18 (11%)		99 (62%)				

Tabela 7.5 – Resultados da simulação – FIFO – limite: 50 pacotes

No intervalo de 160 a 197 segundos, registram-se novas perdas de pacotes, desta vez causadas pela transferência constante dos pacotes maiores, ocupando por mais tempo o canal de comunicação, forçando o enfileiramento e a perda de pacotes. Os números desta simulação podem ser vistos na tabela 7.5. Esta simulação também não teve perdas na piconet.

Ainda na análise do esquema FIFO, com auxílio das figuras A1 e A2, observa-se que há momentos onde o nível de enfileiramento diminui. Esta diminuição é explicada através de dois eventos, o atendimento de vários pacotes de menores tamanhos, reduzindo rapidamente o nível do enfileiramento, e o término da chegada de pacotes menores. Ou seja, o primeiro momento tem uma taxa de atendimentos superior a taxa de chegadas, e o segundo, pela diminuição da taxa de entrada de pacotes.

7.3.1.2 *Stochastic Fair Queuing* (SFQ)

Na análise do esquema SFQ pode ser constatado dois comportamentos distintos, ambos relacionados com a *piconet* e *scatternet.* Na *piconet,* onde os fluxos são uniformes em cada *slave,* as perdas de pacotes está diretamente associada com a quantidade de pacotes presentes na fila. A explicação para este comportamento é a política implementada pelo esquema na atribuição de limites.

Embora esta constatação possa parecer redundante quando observados os comportamentos na *scatternet,* deve-se atentar que o tráfego nos *slaves* é homogêneo, somente pacotes de igual tamanho, e com uma taxa uniforme de chegadas de pacotes. Na *scatternet,* o tráfego é heterogêneo, os pacotes são de tamanhos diversos e a taxa de chegadas é imprevisível.

O esquema SFQ libera para um mesmo fluxo somente 50 % do tamanho máximo especificado para a fila total. Assim, as perdas de pacotes registradas na *piconet* estão diretamente relacionadas com o atingimento deste limite na fila do *slave*. Não há um descarte de pacotes, mas sim a rejeição de entrada.

Explicando com números, no modelo utilizado foi definido um limite máximo de 30 pacotes na fila. As filas formadas na *piconet*, com tráfego homogêneo, estavam então limitadas em 15 pacotes, a metade do tamanho máximo. Esta limitação também se tornou evidente na *scatternet*, quando um fluxo homogêneo atingia o limite médio, obtido pela divisão do valor máximo, 30, pela quantidade de fluxos conhecidos por instante de tempo da simulação.

O esquema SFQ pode ser classificado como um mecanismo ativo de gerenciamento de filas. Atua diretamente sobre o conteúdo da fila, alternando a transferência dos diferentes fluxos e assegurando que cada fluxo tenha sua vez na transferência de pacotes.

Para implementar esta forma de funcionamento, o esquema SFQ forma filas separadas para cada tipo de fluxo de entrada e no esquema de *round-robin* transfere um pacote de cada fluxo por vez. Os pacotes de cada fluxo são retirados considerando um esquema FIFO.

Durante a simulação, este esquema foi adaptando automaticamente os limites de enfileiramento para cada fluxo, em função da taxa de entrada e saída do sistema. A tabela 7.6 apresenta os resultados da simulação com o esquema SFQ.

Os valores tabulados para a *scatternet* podem ser explicados com auxílio do gráfico A3. Esta explicação, realizada a seguir, será mais detalhada por apresentar características específicas, além dos tradicionais mecanismos FIFO e LIFO.

	Stochastic Fair Queuing (SFQ)							
	Scatternet				***Piconet***			
Tamanho	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**
54	29	15 (52%)	2	12 (41%)	29	0	0	29
108	28	17 (61%)	0	11 (39%)	29	1	0	28
216	21	14 (67%)	0	7 (33%)	29	8	0	21
432	17	10 (59%)	0	7 (41%)	29	12	0	17
865	15	5 (33%)	2	8 (53%)	29	14	0	15
1.730	14	0 (0%)	7	7 (50%)	29	15	0	14
	124	61 (49%)		52 (42%)				

Tabela 7.6 – Resultados da simulação – SFQ

No gráfico A3 foram selecionados quatro instantes para serem analisados. O primeiro destes instantes é relativo ao início da perda de pacotes. Quando ocorre o primeiro descarte, a fila tem 17 pacotes em espera, sendo 10 pacotes de 54 bytes, 5 de 108 bytes e 2 de 216 bytes. Neste instante, somente estes três fluxos são conhecidos, fazendo com que a capacidade da fila seja distribuída igualmente entre os três. Cada fluxo pode enfileirar até 10 pacotes, resultado da divisão do limite máximo, 30 pacotes, pelos três fluxos oficialmente presentes no sistema até o momento. O primeiro pacote a ser enfileirado e acima deste limite é imediatamente descartado. Mesmo havendo disponibilidade da fila total, o mecanismo não permite sua entrada.

Na definição automática de novos limites para as filas dos fluxos, é perfeitamente possível acontecer que aos fluxos antes abaixo do seu limite, poderem passar a condição oposta, acima do limite. Isto aconteceu em vários momentos da simulação.

Outra observação importante, decorrente do que foi exposto acima, é a possibilidade de ocorrerem rejeições de pacotes de fluxos cujas filas estão abaixo do seu limite momentâneo, mas, na soma total dos pacotes enfileirados já foi atingido o limite máximo definido pelo esquema.

Faz-se mister esclarecer um aspecto sobre a eliminação de pacotes no esquema SFQ em razão do que foi comentado anteriormente, sobre as diferenças entre os mecanismos quanto à rejeição ou eliminação de pacotes. SFQ rejeita a entrada de pacotes do fluxo cuja fila atinja o limite a ele definido para aquele momento. Se pacotes de outros fluxos chegarem a filas que não atingiram ainda o limite estabelecido para elas naquele instante, estes pacotes são aceitos. Sendo assim, SFQ faz uma seleção dos pacotes de cada fluxo antes da decisão de rejeitar ou aceitar sua entrada na fila.

Continuando a análise, o segundo instante compreende o intervalo de 35 a 42 segundos da simulação, quando a fila apresenta um número maior de pacotes enfileirados. Neste intervalo, os seis fluxos da simulação já enviaram pelo menos um pacote a *scatternet*. O algoritmo do esquema redefine o limite máximo de 30 pacotes dividindo-o para os seis fluxos conhecidos neste momento, resultando em no valor máximo de cinco pacotes para cada fluxo. São rejeitados pacotes pertencentes a todo fluxo cuja quantidade de pacotes estiver acima deste novo limite é rejeitado.

Aos 32 segundos da simulação, a fila esta composta de 10 pacotes de 54 bytes, oito pacotes de 108 bytes, quatro de 216 bytes, dois pacotes de 432 bytes, dois pacotes de 865 bytes e um pacote de 1730 bytes. Nos três segundos anteriores, são negados: um

pacote de 54 bytes e dois pacotes de 108 bytes. Somente um pacote de 432 bytes é aceito na fila.

Um pacote de 432 bytes entrou na fila em meio às várias rejeições registradas. Isto somente foi possível porque o limite total e o limite definido para este fluxo neste instante não tinham sido atingidos. O fluxo de 432 bytes tinha somente um pacote enfileirado quando podia chegar a cinco e o fluxo total contava com 26 pacotes de um total de 30 possíveis.

O terceiro instante selecionado para avaliação consiste do intervalo de 58 a 67 segundos de simulação, com menor índice de rejeição de pacotes. O tráfego mais agressivo é formado pelo fluxo de pacotes de 216 bytes.

São atendidos dois pacotes, 216 e 438 bytes, ou seja, dois pacotes saíram da fila. Mas foram aceitos um pacote de cada tamanho, pois os limites dos fluxos haviam se reduzido. Há também a entrada de dois pacotes dos fluxos de 865 e 1730 bytes, aceitos por haver espaço na fila total e também nos valores atribuídos pelo mecanismo aos limites dos fluxos.

O quarto intervalo selecionado demonstra a redução do nível de enfileiramento e a continuidade das rejeições. Embora aparente ser uma ambigüidade, as rejeições de pacotes acontecem com o fluxo de 432 e 865 bytes, sendo o primeiro deles o mais agressivo agora. Os pacotes são rejeitados porque o limite dos fluxos estava no seu limite máximo.

No quarto e último instante da análise, havia ainda enfileirados: nove pacotes de 54 bytes, sete de 108 bytes – ambos acima do limite dos fluxos, três de 216 bytes, quatro de 432 bytes, três de 865 bytes e também três de 1730 bytes.

7.3.1.3 *Fair Queuing* (FQ)

Observando o gráfico A4, relativo ao esquema FQ, constata-se a similaridade do comportamento com o esquema FIFO. O ponto único de diferença observado aconteceu entre o intervalo de 44 e 50 segundos, onde houve uma rejeição de pacote no esquema FIFO e que não aconteceu no esquema FQ.

Para explicar esta diferença é necessária uma análise das configurações e parametrizações dos objetos e classes definidas no simulador ns-2. O simulador estava pré-configurado para limites iguais ou superiores à 50 pacotes para o esquema FQ. Sem esta análise não é possível explicar o motivo de tal comportamento.

	Fair Queuing (FQ)							
	Scatternet				***Piconet***			
Tamanho	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**
54	29	0 (0%)	0	29 (100%)	29	0	0	29
108	29	3 (10%)	0	26 (90%)	29	0	0	29
216	29	6 (21%)	0	23 (79%)	29	0	0	29
432	29	4 (14%)	12	13 (45%)	29	0	0	29
865	29	4 (14%)	20	5 (17%)	29	0	0	29
1.730	15	0 (0%)	11	4 (27%)	29	0	14	15
	160	17 (11%)		100 (63%)				

Tabela 7.7 – Resultados da simulação – FQ

Praticamente, a eliminação dos pacotes acontece quando a fila atinge o limite definido de 50 pacotes. Qualquer pacote que chegar ao dispositivo quando este já contiver 50 pacotes enfileirados é automaticamente descartado. O descarte de pacotes acontece independente de outros critérios, como tamanho ou fluxo de origem. A tabela 7.7 mostra os resultados obtidos com a simulação.

7.3.1.4 *Deficit Round Robin* (DRR)

A observação do comportamento da fila sob o esquema DRR, gráfico A5, apresenta similaridades em relação aos esquemas anteriores. Não significa que o dinamismo seja o mesmo, mas os resultados das simulações apresentam situações idênticas. O limite total de enfileiramento acontece praticamente nos mesmos intervalos. A diferença está relacionada aos tamanhos dos pacotes atendidos.

Analisando os dados do arquivo *trace*, e de certa forma representada na linha “Em serviço”, não se constata a presença de pacotes maiores em transmissão nos segundos iniciais. Esta constatação pode ser feita por uma linha reta sensivelmente superior ao resto da linha “Em serviço”. Por exemplo, aos 32 segundos ocorre a transferência do primeiro pacote de 432 bytes, aos 84 segundos o primeiro pacote de 865 bytes e, aos 174 segundos, o primeiro pacote de 1730 bytes.

Na tabulação dos resultados finais, tabela 7.8, percebe-se de forma mais clara o comportamento do esquema DRR. Os pacotes maiores são os que apresentam os maiores números de descartes e maiores números de pacotes enfileirados ao final do processo de simulação. Os pacotes menores, além de não terem nenhuma perda, também não permaneceram na fila ao término da simulação.

Um ponto importante desta simulação é identificar como o simulador define os critérios para eliminação de pacotes. Toda vez em que o limite de 25.000 bytes era superado processava-se uma retirada de pacotes da fila.

A grande diferença deste esquema e os anteriores esta relacionada diretamente com o processo de atendimento e descarte de pacotes. Enquanto que os mecanismos anteriores rejeitam a entrada de novos pacotes quando a fila está no seu limite, o esquema DRR elimina pacotes já presentes na fila. O processo de descarte considera o tempo de presença na fila e o tamanho do pacote. Quanto mais tempo estiver na fila e maior o tamanho do pacote, maior a probabilidade de ser eliminado.

A quantidade de pacotes eliminados procura sempre atingir uma folga de aproximadamente 5 % do tamanho total da fila, em bytes. Isto pode envolver um ou mais pacotes, o que explica as concentrações de eliminações em alguns momentos da simulação.

	Deficit Round Robin (DRR)							
	Scatternet				***Piconet***			
Tamanho	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**	**Recebidos**	**Perdidos**	**Na fila**	**Atendidos**
54	29	0 (0%)	0	29 (100%)	29	0	0	29
108	29	0 (0%)	0	29 (100%)	29	0	0	29
216	29	0 (0%)	1	28 (97%)	29	0	0	29
432	29	7 (24%)	9	13 (45%)	29	0	0	29
865	29	12 (41%)	11	6 (21%)	29	0	0	29
1.730	15	6 (40%)	6	3 (20%)	29	14	0	15
	160	25 (16%)		105 (68%)				

Tabela 7.8 – Resultados da simulação – DRR

7.3.2 Latência

Em todos os esquemas simulados ficou evidente o impacto dos pacotes maiores nos tempos de latência. Quando observados os gráficos do anexo B, torna-se necessário separar a análise entre os esquemas FIFO e FQ dos esquemas SFQ e DRR.

Em ambos os gráficos do esquema FIFO, figuras B1 e B2, há o mesmo comportamento, com crescimento constante do tempo de latência. Não é o tamanho da capacidade de enfileiramento da fila que modifica o comportamento, mas sim o perfil do conteúdo da fila, o que será esclarecido nos parágrafos seguintes. O esquema FQ apresenta, com pequenas e sutis variações, um crescimento constante muito próximo do que o observado no esquema FIFO (Ver figura B4).

É na latência que pode ser observada a diferença do esquema SFQ. Enquanto que pacotes são rejeitados, a latência dos que são liberados pelo sistema tende sempre a ser baixa.

Analisando a figura B3, a partir dos 51 segundos ocorre o primeiro aumento, causado pelo frequência maior de transferência dos pacotes grandes em relação aos transferidos inicialmente. O maior volume de transferência estava, inicialmente, nos pacotes de 54, 108 e alguns de 216 bytes. A partir de 51 segundos já entram no sistema pacotes de 432, 865 e 1730 bytes.

A partir de 90 segundos todos os fluxos passam a ter pacotes transferidos. Dada a variação nos tamanhos dos pacotes, a latência passa ser crescente, agora pela presença dos pacotes maiores

Analisando a latência no esquema DRR (figura B5), observa-se um comportamento similar ao do esquema FIFO no intervalo de 0 a 128 segundos de simulação. É o período em que a latência permanece abaixo dos 100 segundos.

A diferença entre os dois esquemas evidencia-se quando observado o tempo necessário para atingir 150 segundos de latência, onde DRR apresentou o melhor indicador, levando 210 segundos de simulação. FIFO precisou de 174 segundos para atingir este tempo, uma diferença de 20% no tempo.

Esta diferença de comportamento é o resultado do processo de gerenciamento da fila efetuado pelo DRR, privilegiando os pacotes menores e, consequentemente, aumentando a taxa de saída da fila. (figura B5).

Quanto mais aumenta a taxa de chegadas de pacotes grandes, maior passa a ser o a latência. Observando os gráficos dos esquemas SFQ e DRR, os aumentos nos tempos de latência ao final da simulação passam a ser muito mais acentuados do que em relação ao início.

Os aumentos dos tempos são resultados do nível de utilização da fila. Considerando que o tempo de latência de um pacote é a soma dos tempos de transferência de cada pacote que o antecede na fila, quanto mais próximo do limite da fila, mais tempo irá acumular.

Além do nível de utilização da fila, também o perfil dos pacotes presentes influencia no tempo. Se a fila estiver predominantemente preenchida com pacotes menores, menores serão os tempos computados na latência dos pacotes enfileirados

quando comparados com um perfil oposto. Se a predominância for de pacotes maiores, maiores os tempos a serem acumulados pelos pacotes enfileirados.

SFQ tem a tendência de equilibrar os tempos de latência ao procurar atender cada fluxo no processo de *round-robin*. Mantendo um padrão de atendimento em tráfegos uniformes, este esquema tende a definir um tempo padrão de latência. No processo simulado, este padrão é identificado nos momentos em que os menores pacotes predominam no sistema. Na medida em que novos fluxos chegam ao sistema, há mudanças nos tempos, mas sempre com a tendência de equilíbrio.

7.3.3 Jitter

Na análise do *jitter*, valores positivos significam que houve um aumento na latência de dois pacotes consecutivos. Isto significa que o segundo pacote permaneceu mais tempo na fila em relação ao anterior. Valores negativos, ao contrário, significam que o segundo pacote permaneceu menos tempo na fila. Valores nulos significam que ambos pacotes permaneceram o mesmo tempo na fila, caracterizado pela transferência seguida de pacotes com o mesmo tamanho.

Analisando os gráficos do anexo C, as variações positivas do *jitter* estavam associadas à transferência de pacotes maiores, forçando uma situação de enfileiramento, pois a taxa de saída passava a ser menor que a taxa de entrada de pacotes. Este fator torna-se cada vez mais acentuado na medida em que pacotes maiores tornam-se mais frequentes na fila.

Quando as variações eram nulas, principalmente nos esquemas FIFO, FQ e DRR (figuras C1, C3, C4 e C5), a característica comum entre eles estava na ocorrência maior de pacotes iguais consecutivos na fila. Nos esquemas FIFO e FQ, a característica deve-se à taxa de entrada dos pacotes menores ser maior ao início da simulação. No esquema DRR, a característica deve-se ao mecanismo de *round-robin* privilegiar os pacotes menores em cada ciclo de transferência.

Nos esquemas FIFO e FQ, o processo de descarte de pacotes não influencia diretamente no *jitter*, pois os pacotes não chegam a entrar na fila. Entretanto, no esquema SFQ (figura C3), o processo de descarte influencia ao evitar que o *jitter* tenha tendência de aumento para quem já está na fila. No esquema DRR, o processo de descarte influencia diretamente o *jitter* ao eliminar os pacotes que estão mais tempo presentes na fila, além de considerar o tamanho dos mesmos, eliminando primordialmente os pacotes maiores.

A variabilidade mais acentuada é observada no esquema SFQ. Da mesma forma que observado anteriormente, isto é efeito do processo de atendimento das filas de cada fluxo em sequência, um por vez, intercalando todos os fluxos, de pequenos a grandes pacotes.

Ainda no esquema SFQ, a partir dos 118 segundos da simulação, o *jitter* tem variações muito grandes. Este comportamento torna-se padrão a partir deste ponto. Este efeito reforça o que foi afirmado para a latência deste esquema, que tende sempre a uma normalidade, um padrão de comportamento.

O *jitter* também segue o mesmo comportamento dos resultados verificados com a simulação FIFO, limite de 50 pacotes. Enquanto que há predominância de pacotes menores na fila, menorers são as variações nas latências. Quando os pacotes maiores passam a integrar a fila, as flutuações aumentam. Estas flutuações tornam-se cada vez maiores quanto mais pacotes maiores estiverem enfileirados.

No esquema DRR, o *jitter* é regular em sua variação enquanto os pacotes menores estão sendo transmitidos em maior número. As grandes flutuações acontecem a partir do momento em que há uma intercalação entre os pacotes maiores.

*Jitte*r é uma condição totalmente dependente da situação da fila em cada momento. Não somente em termos de pacotes enfileirados, mas também dos perfis destes pacotes, se maiores ou menores.

8. CONCLUSÕES

A necessidade de recursos é geralmente percebida quando sua disponibilidade começa a se tornar escassa. Mesmo sendo válida para qualquer tipo de recurso, é no contexto tecnológico que se pode destacar a importância do tema.

A complexidade envolvida na interligação de sistemas tecnológicos requer a aplicação de inteligência nos modelos para que os sistemas operem da melhor forma possível, com a maximização e otimização dos seus processos. Ao mesmo tempo, os sistemas devem ser funcionais, de fácil aplicação e uso.

O objetivo deste trabalho foi analisar e sugerir uma solução que melhor operacionalizasse o tratamento de pacotes diante de possíveis situações de esgotamento de recursos. Especificamente na formação de filas diante de um ponto de convergência num sistema. É quando equipamentos e sistemas devem estar preparados para prover garantias e assegurar a confiabilidade das funções e serviços prestados.

Tratando inicialmente sobre o esquema FIFO, foi observado que o mecanismo é adequado para tratamento da demanda de pacotes. Porém, quando os limites impostos à fila são atingidos, o que caracteriza uma situação de esgotamento de recursos, o mecanismo recusa novas entradas sem adotar qualquer critério de seleção. Na prática, os pacotes são rejeitados enquanto não houver espaço para novas entradas.

No modelo gerado para as simulações, o limite estabelecido considerou somente a quantidade de pacotes presentes na fila. O sistema não requer qualquer processo de tomada de decisão que não seja o limite de pacotes presentes na fila, ou seja, não são consideradas as características específicas dos pacotes.

Uma variação possível poderia considerar o tamanho da fila em número de bytes. Esta variação, porém, altera as características do esquema FIFO na medida em que implica na necessidade de um sistema de classificação prévia dos pacotes. O sistema precisaria comparar o tamanho do pacote com o espaço disponível e então decidir pela liberação ou descarte.

Embora um sistema classificador de pacotes possa alterar os indicadores de desempenho associados com mecanismos de filas, esta possibilidade complementa o sistema de forma geral, mas não modifica o esquema da fila em si.

Os indicadores de latência, *jitter* e perda de pacotes podem ser melhorados com a utilização de um classificador, mas o tempo e desempenho do processo de classificação também devem ser inseridos e contabilizados nas análises. Além da contabilização de fatores adicionais, a implementação em dispositivos deverá considerar recursos para a execução deste processo.

Mas latência e *jitter* somente ocorrem quando há um processo efetivo de formação de filas. Estes valores são nulos quando não um tempo de espera pela liberação de uso de recursos.

Embora esta afirmação possa implicar em que os valores de latência e *jitter* sejam mais estáveis quanto menor o nível de enfileiramento, a implicação não é suficiente por si somente. É preciso considerar o perfil dos pacotes presentes na fila.

O esquema FIFO, conforme já mencionado, não atua diretamente sobre os elementos da fila. Sendo assim, nenhum fator externo pode influenciar nos tempos de latência e *jitter* que não seja a chegada e nível de utilização da fila. Latência e *jitter* são, então, resultados do comportamento da fila. O comportamento da fila é determinado pelas caraterísticas dos pacotes presentes. A latência e o *jitter* são tão variáveis quanto forem diferentes os tamanhos dos pacotes na fila.

As simulações realizadas consideraram o impacto causado pelo aumento do tamanho da capacidade de pacotes presentes na fila. Esta consideração não contribuiu para um desempenho melhor na questão do enfileiramento do *buffer* da interface do *Host Controller* do Bluetooth. Não há uma contribuição efetiva porque não há mudanças na dinâmica da fila – o esquema de fila é o elemento determinante, neste caso, FIFO.

Como o esquema FIFO não implementa qualquer política para tratamento do tráfego, qualquer fluxo mais agressivo pode tomar todos os recursos e preencher a fila

exclusivamente com seus pacotes, em detrimento aos demais fluxos, independente de apresentarem ou não requisitos de qualidade.

O esquema SFQ implementa um esquema de *round-robin* para todos os fluxos de entrada. Passa a ser uma abordagem diferente de FIFO, que atende por ordem de chegada. SFQ atribui recursos de forma equilibrada para todos os fluxos de entrada.

Para cada fluxo de entrada são atribuídos limites. Estes limites são dinâmicos, conforme a quantidade de fluxos que se fizerem presentes no sistema. Os limites são calculados pela divisão do limite máximo pela quantidade de fluxos presentes. A cada novo fluxo ocorre uma nova divisão, sempre atribuindo novos limites e atribuídos igualmente para todos os fluxos.

O mecanismo de *round-robin* faz com que cada fluxo envie um pacote por vez. Os fluxos mais agressivos terão o mesmo tratamento que os fluxos menos agressivos. Os primeiros terão maior probabilidade de perda de pacotes, pois estarão atingindo o limite atribuído pelo mecanismo SFQ com maior frequência.

Outra diferença entre os dois esquemas de filas está no processo de retirada de pacotes da fila. FIFO toma o pacote mais antigo do sistema, sem considerar os fluxos. SFQ retira um pacote de cada fluxo por vez, sendo retirado de cada fluxo o pacote mais antigo.

A rejeição de pacotes é outra diferença, pois também é tratada por fluxo. O fluxo mais agressivo pode ter a maior perda de pacotes por dois motivos: ou (1) por atingir seus limites com maior rapidez; ou (2) por ter reduzido seu limite com a entrada de um novo fluxo e então passar à condição de excesso de pacotes presentes.

Aplicando o esquema SFQ para substituir FIFO na interface do *Host Controller*, obtém-se a garantia de atendimento de fluxos de tráfego diferentes entre si. No problema exposto, dois tráfegos, um melhor esforço e outro com requisitos de qualidade de serviços, não necessariamente haverá algum privilégio para um dos tráfegos, mas sim a garantia de recursos para ambos. Ou seja, o *buffer* não ficará congestionado com somente um tipo de tráfego.

Como o próprio Bluetooth implementa um mecanismo de qualidade de serviços, o esquema SFQ assegura o funcionamento do processo, eliminando o ponto falho do esquema FIFO – congestionamento dorecurso pelo tráfego com maior taxa de chegadas.

Porém, mesmo com o esquema SFQ resolvendo o problema de congestionamento da fila, o ponto crítico do esquema está nos tempos de latência e *jitter*. Isto pode ser

observado comparando-se os gráficos dos anexos D e E. Nestes gráficos estão demonstrados os impactos sobre os diferentes pacotes comparados com os esquemas de filas simulados.

Analisando o esquema FQ nada pode ser afirmado de forma conclusiva quando comparada com FIFO, tendo como base os resultados das simulações. FQ também atende pela ordem de chegada, e rejeita pacotes sem considerar os fluxos.

A análise deste esquema reforça também a afirmação feita em [23], sobre a sua pouca eficiência, razão pela qual é substituído por outros esquemas, como *Start time Fair Queuing, Stochastic Fair Queuing* (SFQ), *Self-clocked Fair Queuing, Eligible Start time Fair Queuing, Smallest Eligible Fair Finishing Time First*, baseados em parâmetros com baixo compartilhamento de reservas [32]

O esquema DRR apresenta a melhor utilização de recursos entre os fluxos de entrada. Aplicando um esquema de créditos acumulativos e saques de créditos, proporciona uma distribuição mais "justa" de recursos para os fluxos de entrada.

Obtendo uma forma de distribuição mais equilibrada, pode-se afirmar que DRR mostrou-se como a melhor alternativa para solucionar o problema apresentado. A perda de pacotes está relacionada com os limites da fila. Latência e *jitter* tendem a apresentar menores variações, e estas tendem a ser mais uniformes. A figura A5 ilustra o que foi comentado.

Se a fila da interface do *Host Controller* pode ficar congestionada com um tipo de tráfego quando utilizado o esquema FIFO, o mesmo não acontece com o esquema DRR. O esquema DRR, através de créditos e saques, equilibra o atendimento aos fluxos. Se um fluxo não tem créditos suficientes, não é atendido, acumulando-os então para a rodada seguinte. São também acumulados créditos quando estes forem superiores ao que é necessário para uso na rodada em curso – neste caso soma-se a diferença entre crédito e uso com o novo crédito atribuídos à fila.

8.1 Trabalhos futuros

A continuidade deste trabalho pode ser direcionada para avaliação de outros esquemas. Incluem-se nesta lista esquemas como o *Random Early Detection* (RED), *Weighted Random Early Detection* (WRED), e as variações do esquema FQ, *Start time*

Fair Queuing, Self-clocked Fair Queuing, Eligible Start time Fair Queuing, Smallest Eligible Fair Finishing Time First.

Alguns destes esquemas são utilizados em roteadores, e, se bem observarmos, o dispositivo Bluetooth que atua como *gateway* numa *scatternet* pode ser considerado como um roteador. A literatura específica sobre RED é facilmente encontrada em vários sites de pesquisa.

Outra oportunidade de trabalho futuro envolve a simulação de tráfego IP sobre Bluetooth, com confirmação de recebimento (*round trip*). O simulador ns-2 está configurado para gerar pacotes TCP Reno, NewReno e Vegas.

Da oportunidade anterior deriva outra oportunidade de trabalho futuro, e uma das áreas de pesquisa que ainda permanece aberta em relação ao Bluetooth e redes *ad hoc*, relacionadas com mecanismos de reserva de banda através do protocolo *Resource Reservation Protocol* (RSVP).

Seguindo na mesma linha, uma abordagem interessante que surgiu durante a realização deste trabalho, é a elaboração de uma proposta de mecanismos de QoS similares ao esquema *Differentiated Services* (*DiffServ*). Considerando que alguns nós estejam fixos em determinada área de cobertura, estes nós poderiam implementar características de serviços que habilitem a formação de *scatternets* com qualidade de serviço.

Para minimizar as caracterísitcas de imprevisibilidade de movimentos, típicos de redes móveis, as propostas anteriores podem ser complementadas com algoritmos de previsibilidade de tráfego. Trabalhos similares estão propostos considerando redes de comunicação celular.

Especificamente nos aspectos da tecnologia Bluetooth, este trabalho poderia ser complementado com mecanismos que simulem as características de *polling* do master aos *slaves*. Com modificações no algoritmo de polling, características de banda variável podem ser implementadas se considerar *pollings* mais frequentes para os mesmos dispositivos. Entretanto, as pesquisas encontradas relativas a este aspecto concentraram-se na fundamentação matemática, sem que houvesse alguma simulação para visualizar e analisar o comportamento de tal esquema.

9. BIBLIOGRAFIA

[01] "*Introduction to Wireless LANs*" – http://www.wlana.org, em 31/01/2002.

[02] IEEE Computer Society LAN MAN Standards Committee. "*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*". New York, New York, 1997. IEEE Std. 802-11, 1997.

[03] PERKINS, Charles. "*Ad hoc Networking*". Addison-Wesley, New Jersey, 2001.

[04] CORSON, S., MACKER, J. "*Mobile Ad hoc Network (MANET): Routing Protocol performance Issues and Evaluation Considerations*". IETF RFC 2501, January 1999.

[05] HÄNNIKÄINEN, M., HÄMÄLÄINEN, T.D., NIEMI, M., SAARINEN, J. "*Trends in personal wireless data communications*". Elsevier Computer Communications 25 (2002) 84-99.

[06] "*The RF Spectrum – An Overview*" – http:/ /www.SSS-mag .com/work1.html, em 16/02/2002.

[07] SCHILLING, Donald L., PICKHOLTZ, Raymond L., MILSTEIN, Laurence B. "*Spread Spectrum Goes Commercial*", IEEE Spectrum, Agosto, 1990

[08] ROBERTS, Randy. "*Introduction to Spread Spectrum*" – http:/ /www.SSS-mag. com, em 16/02/2002.

[09] KAMERMAN, A. "*Spread Spectrum Techniques Drive WLAN Performance*", Microwaves & RF, September, 1996. pags. 109-114.

[10] HILLER, Kimberly. "*Wireless LAN: An Overview*", Gartner Group Research, http://www.techrepublic.com . Publicado 04/01/2001.

[11] MILLER, Micheal. "*Descobrindo Bluetooth*"; tradução de Altair Dias Caldas de Moraes e Claudio Belleza Dias. – Rio de Janeiro : Campus, 2001.

[12] YAIZ, Rachid Ait, HEIJENK, Geert. "*Polling in Bluetooth a Simplified Best effort Case*", Proceedings of the 7th annual CTIT Workshop, Enschede, Fevereiro, 2001.

[13] GUO, Y. "*Philiphs FM/IF systems for GMSK/GFSK receivers*", Philips Semiconductors 1997, http://www.semiconductors.com/acrobat/applicationnotes/ AN1997.pdf. em 28/02/02.

[14] *Home page* do Palowireless Wireless Resource Center, http://www.palowireless.com/ infotooth/tutorial/radio.asp.- em 04/02/2002.

[15] LINDHOLM, Tancred. "*Setting up a Bluetooth packet Transport Link*". – Department of Computer Science, Helsinki University of Technology.

[16] *Home page* do ABC Klubben - Suécia, http://www.abc.se/~m10183/bluet02.html. - em 04/02/2002.

[17] *Home page* da Ericsoon, http://learning.ericsson.net/ Bluetoothdemo/ summary/ blxs5.html. – em 28/01/2002.

[18] *Home page* da Ericsson, http://learning.ericsson.net/Bluetoothdemo/summary/ blxs12.html – em 28/01/2002.

[19] *Home page* da Ericsson, http://learning.ericsson.net/Bluetoothdemo/summary/ blxs13.html – em 28/01/2002.

[20] "*Specification of the Bluetooth System*, v1.1", February 22, 2001, http: //www.Bluetooth.com/ - em 11/01/2002.

[21] SCHWINGENSCHLÖGL, Christian; HEIGL,Anton. "*Development of a Service Discovery Architecture for the Bluetooth Radio System*". Technische Universität München, Institute of Communication Networks.

[22] LAW, Ching, MEHTA, Amar K., SIU, Kai-Yeung. "*Performance of a new Bluetooth scatternet formation protocol*". Proceedings of the ACM Symposium on Mobile Ad hoc Networking and Computing (MobiHoc) 2001, Long Beach, California, USA, October 2001.

[23] BERNET, Yoram. "*Networking Quality of Service and Windows Operating Systems*". New Riders Publishing and Microsoft Corporation, USA, 2001.

[24] CHOI, Sunghyun. "*Qos Guarantees in Wireless/Mobile Networks*". Dissertação para o grau de Doctor of Philosophy. University of Michigan, 1999.

[25] NICHOLS, K. Jacobson, V. ZHANG, L. "*A Two-bit Differentiated Services Architecture for the Internet*". IETF Internet Draft <draft-nichols-diff-svc-arch-00.txt>, November 1997.

[26] CLARK, D., WROCLAWSKI, J. "*An approach to Service Allocation in the Internet*". IETF Internet Draft <draft-clark-diff-svc-alloc-00.txt>, July 1997.

[27] ARMITAGE, Grenville. "*Quality of service in IP networks: foundations for a multi-service Internet*". Technology Series, MacMillan Technical Publishing, USA, 2000.

[28] DEMERS, Alan, KESHAV, Srinivasan, SHENKER, Scott. "*Analysis and simulation of a fair queueing algorithm*". Proceedings of the Sigcomm '89 Symposium on Communications Architectures and Protocols, 19(4):1-12, September 1989.

[29] KESHAV, Srinivasan. "*On the efficient implementation of fair queueing*". In Internetworking: Research and Experience Vol. 2, 157-173, September 1991.

[30] MCKENNEY, Paul E. "*Stochastic fairness queueing*". In Internetworking: Research and Experience Vol. 2, 113-131, January, 1991.

[31] NAGLE, John B. "*On Packet Switches with Infinite Storage*". IEEE Transactions on Communications, Vol. COM-35, No. 4, 435-438, April 1987.

[32] ABUAMSHA, Oula, PEKERGIN, Nihal. "*Comparison of Fair Queuing Algorithms with a Stochastic Approach*". In Proceedings of Mascots'98, pages 139-144, 1998.

[33] SHREEDHAR, M, VARGHESE, G. "*Efficient fair queuing using deficit round robin*". Proceedings of ACM SIGCOMM '95, Aug. 1995.

[34] ZEE, Martin van der, HEIJENK, Geert. "*Quality of Service in Bluetooth Networking – Part I*". Doc. no. 10/0362-FCP NB 102 88 Uen, 03/01/2001.

[35] MANIATIS, Petros, ROUSSOPOULOS, Mema, SWIERK, Ed, LAI, Kevin, APPENZELLER, Guido. ZHAO, Xinhua. BAKER, Mary. "*The Mobile People Architecture*". Mobile Computing and Communications Review, July 1999.

[36] THAI, B., SENEVIRATNE, Aruna. "*IPMoA: Integrated Personal Mobile Architecture*". Proceedings for International Symposium on Computers and Communications (ISCC), Hammamet, Tunisia, 2001.

[37] CHAN, Jonathan, ZHOU, S., SENEVIRATNE, Aruna. "*A QoS Adaptive Mobility Prediction Scheme for Wireless Networks*". Proceedings of IEEE GLOBECOM'98, November, 1998.

[38] SU, William, LEE, Sung-Ju, GERLA, Mario. "*Mobility prediction and routing in ad hoc wireless networks*". International Journal of Network Management, 2001; 11:3-30.

[39] ARDON, S., DIOT, C., LANDFELD, B., SENEVIRATNE, Aruna. "*Resources Reservation in a Reactive QoS Scheme*". University of NSW, Kensington 2001, Austrália.

[40] BROCH, David A., MALTZ, David B. Johnson, HU, Yih-Chun, JETCHEVA, Jorjeta. "*A Performance Comparison of Multi-Hop Wireless Ad hoc Network Routing Protocols*". ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom98), *page*s 85-97, 1998.

[41] BHAGWAT, Pravin, SEGALL, Adrian.. "*A routing vector method (RVM) for routing in Bluetooth scatternets*". IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99).

[42] KALIA, M. GARG, S.,SHOREY, R. "*Scatternet Sctruture and Inter-Piconet Communication in the Bluetooth System*". IEEE National Conference on Communications, New Delhi, 2000.

[43] MIKLOS, Gy. Racz, A. Turanyi, Z. Valko, A. Johansson, P. "*Performance aspects of Bluetooth scatternet formation*". Proceedings of The First Annual Workshop on Mobile Ad hoc Networking and Computing, 2000.

[44] LAW, Ching, SIU, Kai-Yeung. "*A Bluetooth scatternet formation algorithm*". Proceedings of the IEEE Symposium on Ad hoc Wireless Networks 2001. San Antonio, Texas, USA, November 2001.

[45] SALONIDIS, Theodoros, BHGWAT, Pravin, TASSIULAS, Leandros, LAMAIRE, Richard. "*Distributed Topology Construction of Bluetooth Personal Area Network*". Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societes, 2001.

[46] AGGARWAL, Alok, KAPOOR, Manika, RAMACHANDRAN, Lakshmi, SARKAR, Abhinanda. "*Clustering algorithms for wireless ad hoc networks*". Proceedings of the 4th International Workshop on Discrete Algorithms and

Methods for Mobile Computing and Communications, Boston, MA, August 2000, pp. 54-63.

[47] PATI, H. K., MALL, R., SENGUPTA, I. "*An efficient Bandwidth reservation and call admission control scheme for wireless mobile networks*". Elsevier, Computer Communications 25 (2002) 74-83

[48] FALL, K., VARADHAN, K.,editors (2001) "*The ns Manual (formely ns Notes and Documentation)*". The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox Parc, work in progress. (22/03/2002 disponível em http: //www.isi.edu/nsman/ns/ns-documentation.html).

[49] JAIN, Raj. "*The art of computer systems performance analysis*". John Willey & Sons, NY, 1991.

[50] FREITAS Filho, Paulo José. "*Introdução à modelagem e simulação de sistemas – com aplicações em Arena"*. Visual Books, Novembro, 2001.

ANEXO A – Gráficos – Enfileiramento e Perdas de pacotes

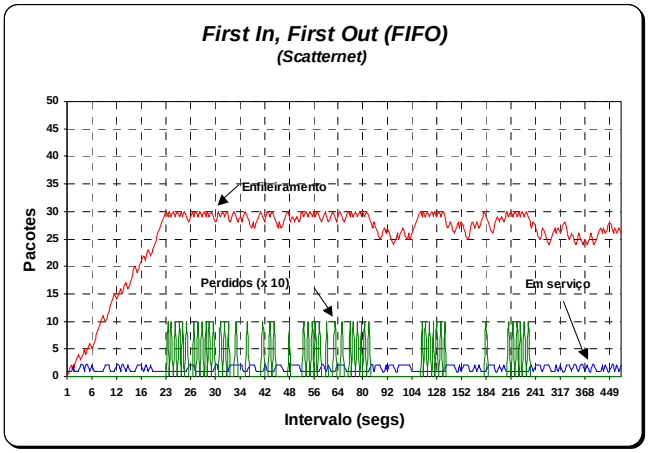

Figura A1 – Comportamento esquema FIFO (30)

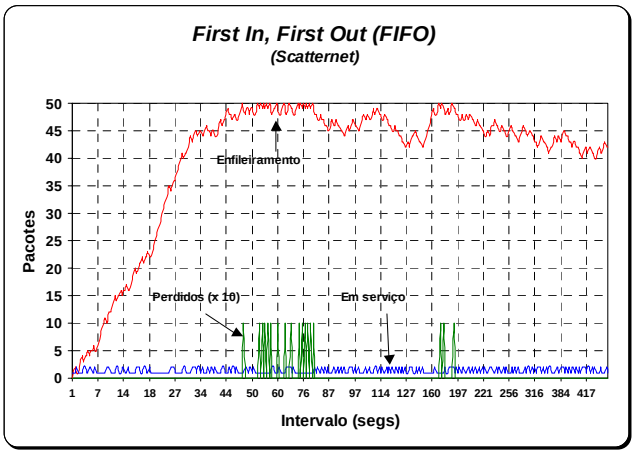

Figura A2 – Comportamento esquema FIFO (50)

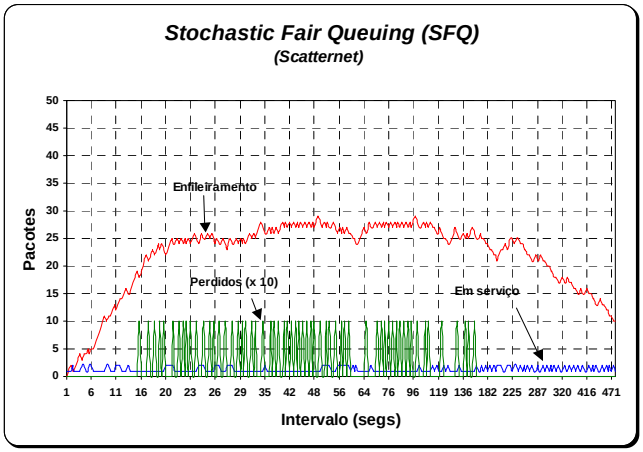

Figura A3 – Comportamento esquema SFQ

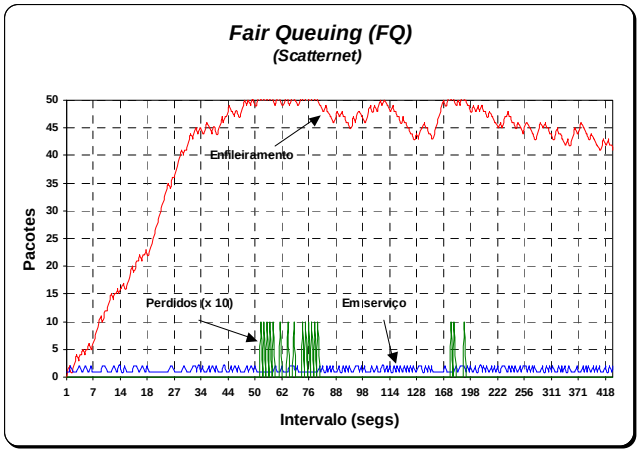

Figura A4 – Comportamento esquema FQ

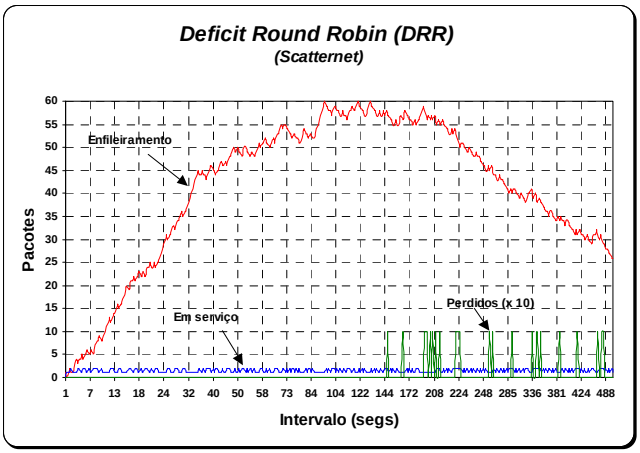

Figura A5 – Comportamento esquema DRR

ANEXO B – Gráficos – Latência

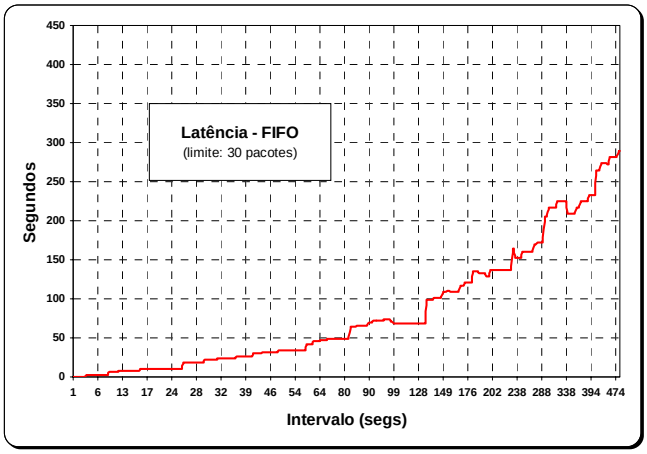

Figura B1 – Latência – esquema FIFO (30)

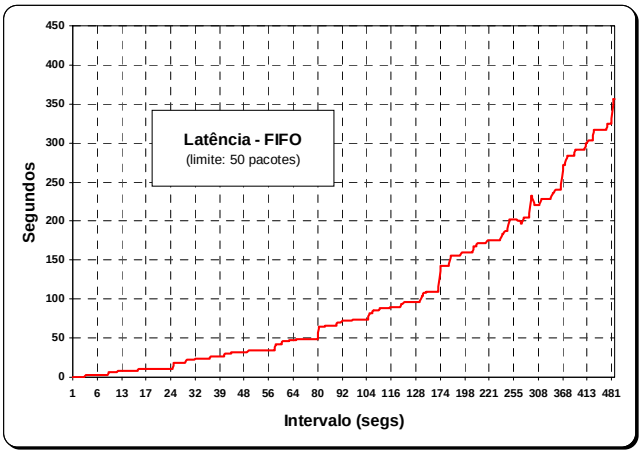

Figura B2 – Latência – esquema FIFO (50)

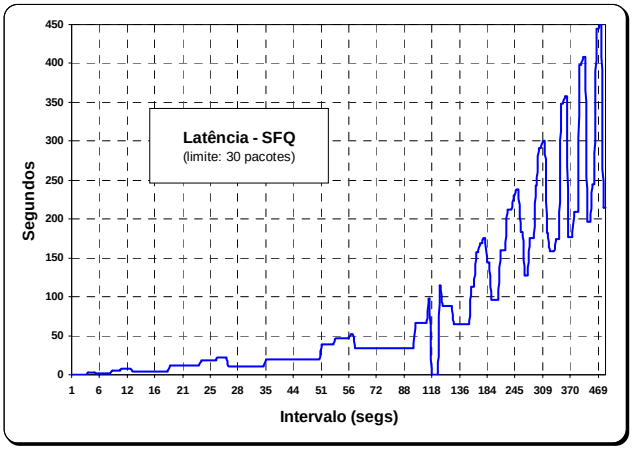

Figura B3 – Latência – esquema SFQ

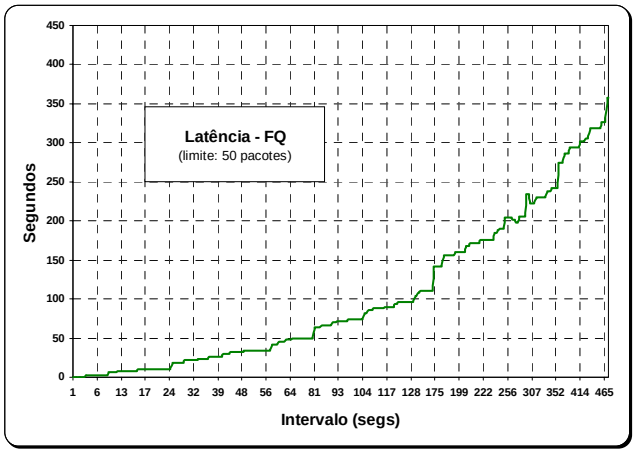

Figura B4 – Latência – esquema FQ

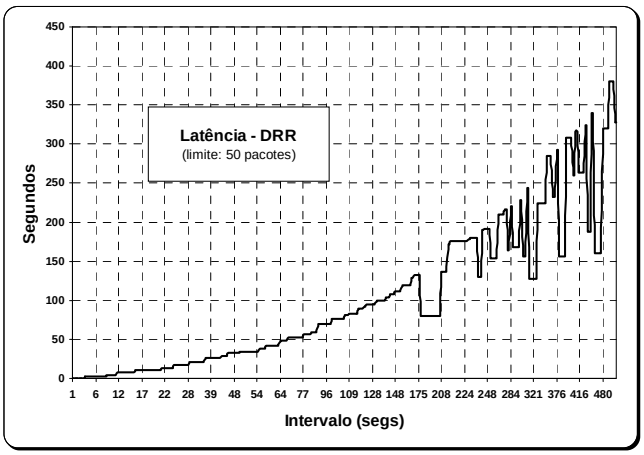

Figura B5 – Latência – esquema DRR

ANEXO C – Gráficos – *Jitter*

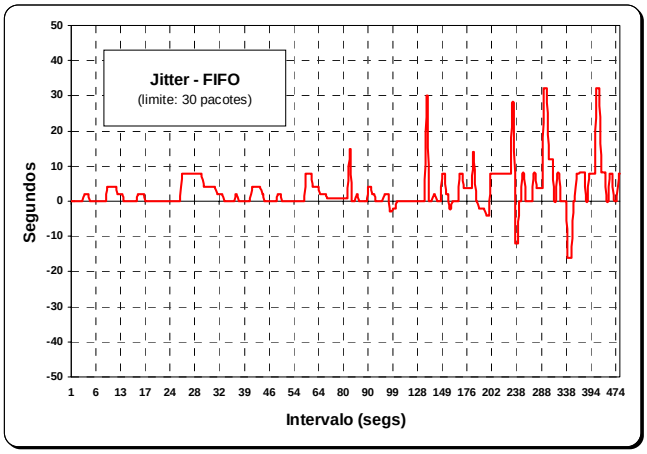

Figura C1 – *Jitter* – esquema FIFO (30)

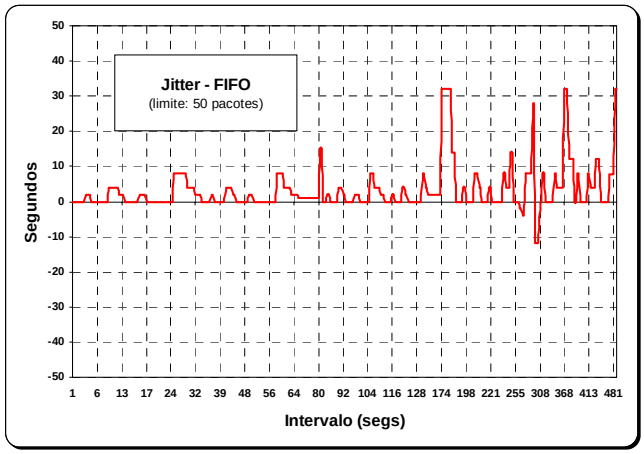

Figura C2 – *Jitter* – esquema FIFO (50)

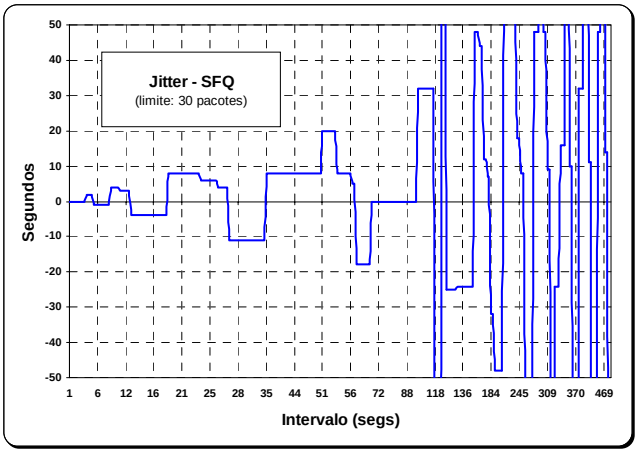

Figura C3 – *Jitter* – esquema SFQ

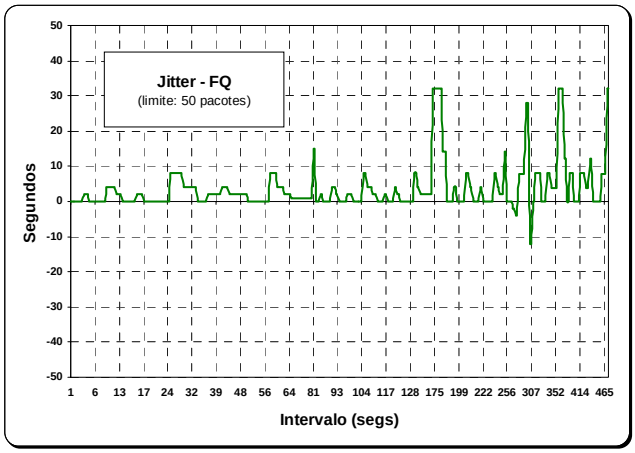

Figura C4 – *Jitter* – esquema FQ

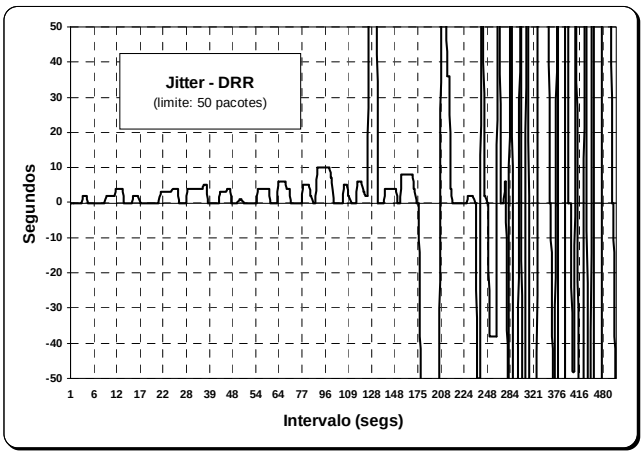

Figura C5 – *Jitter* – esquema DRR

ANEXO D – Gráficos – Latência de Pacotes vs Esquemas

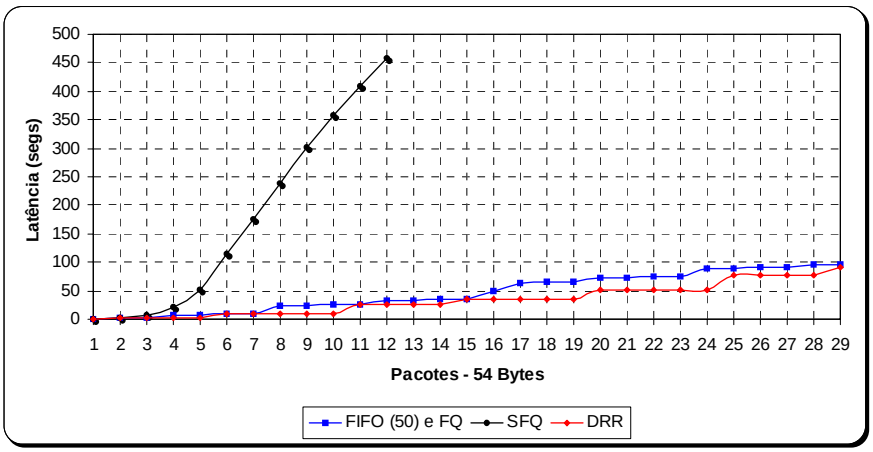

Figura D1 – Latência pacotes 54 bytes / Esquemas de filas

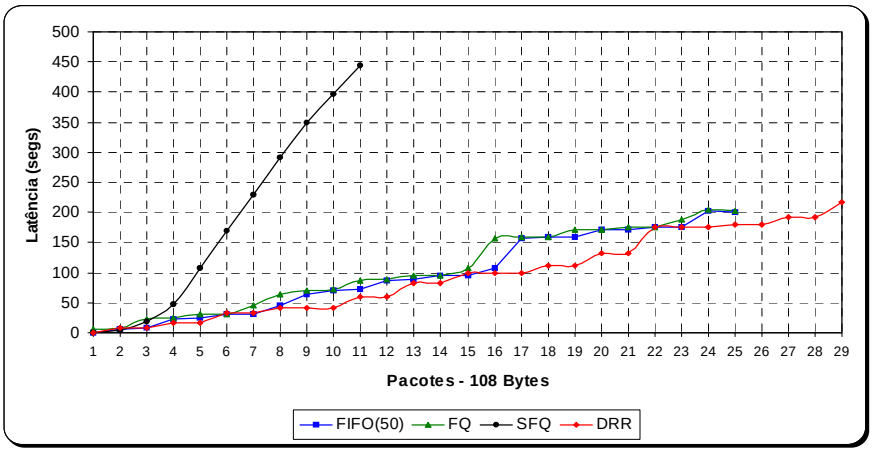

Figura D2 – Latência pacotes 108 bytes / Esquemas de filas

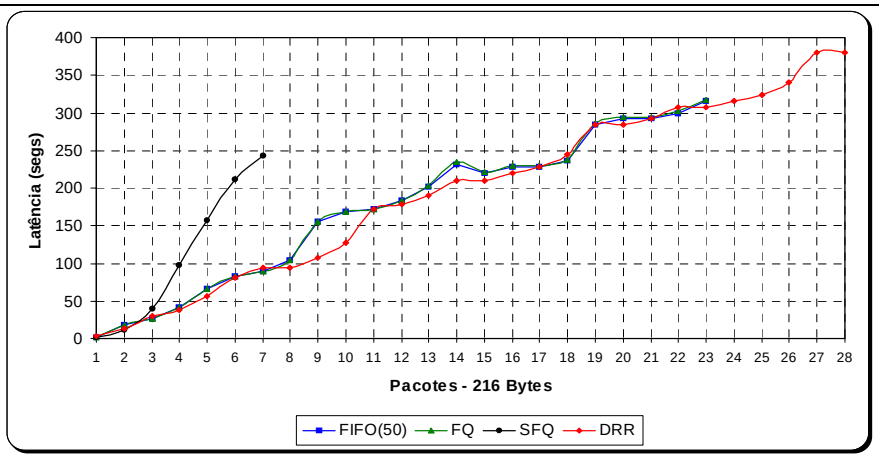

Figura D3 – Latência pacotes 216 bytes / Esquemas de filas

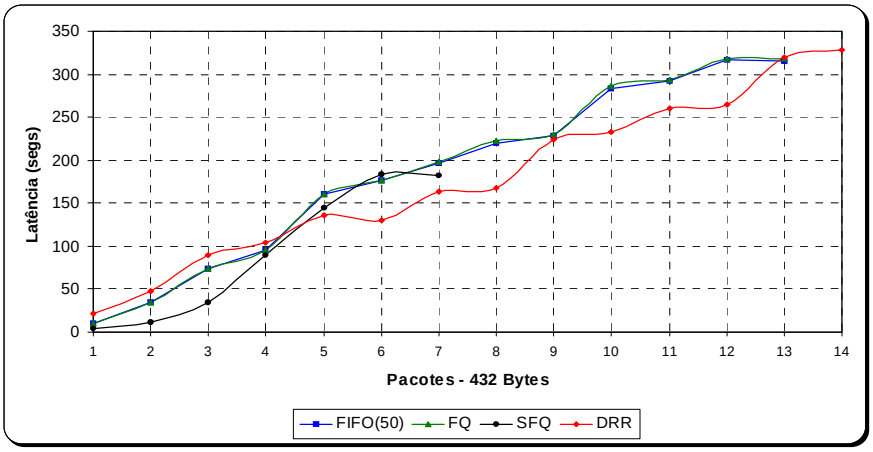

Figura D4 – Latência pacotes 432 bytes / Esquemas de filas

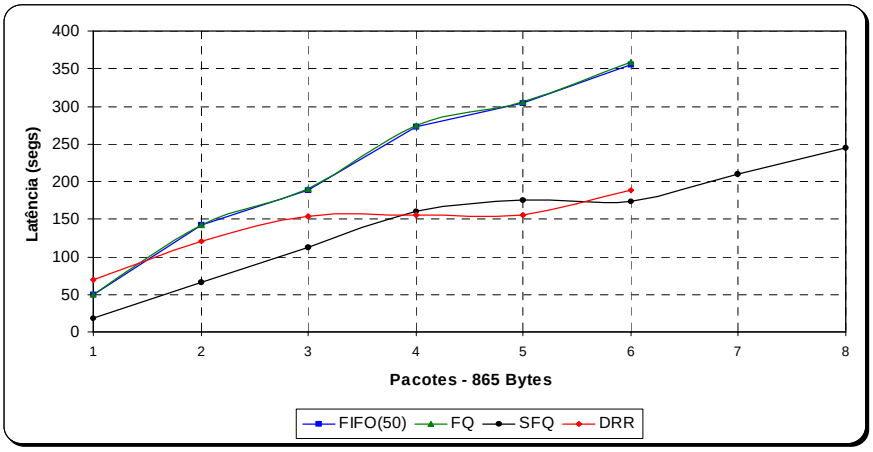

Figura D5 – Latência pacotes 865 bytes / Esquemas de filas

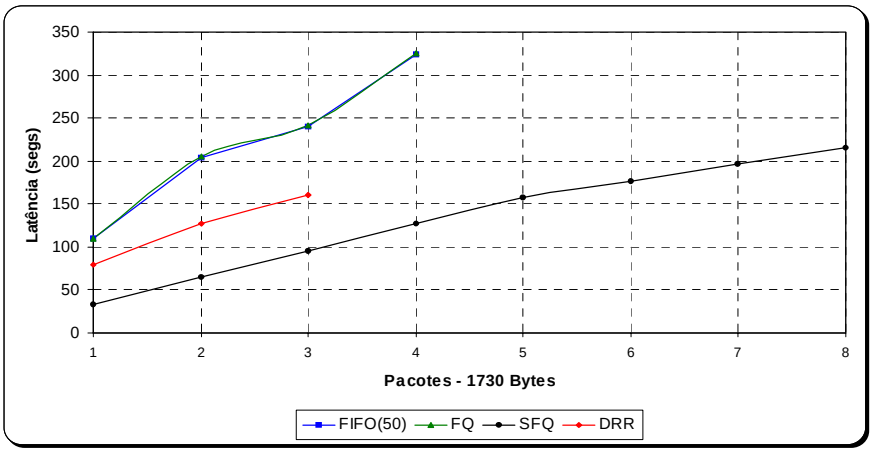

Figura D6 – Latência pacotes 1730 bytes / Esquemas de filas

ANEXO E – Gráficos – *Jitter* de Pacotes vs Esquemas

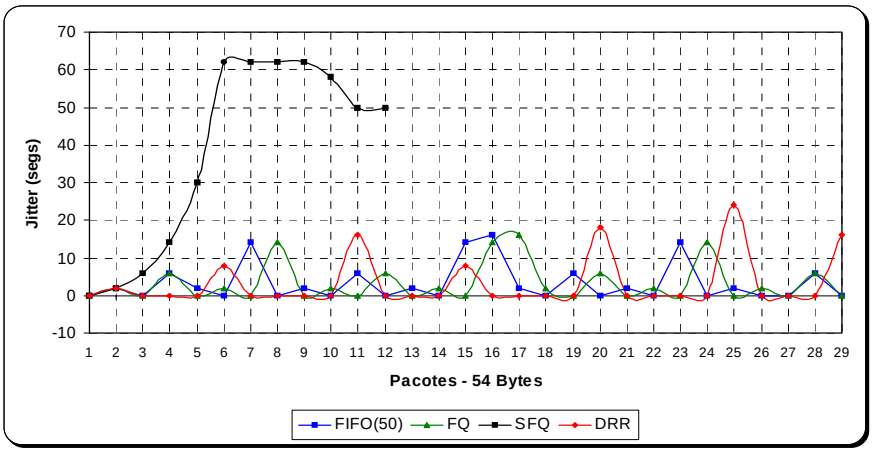

Figura E1 – *Jitter* pacotes 54 bytes / Esquemas de filas

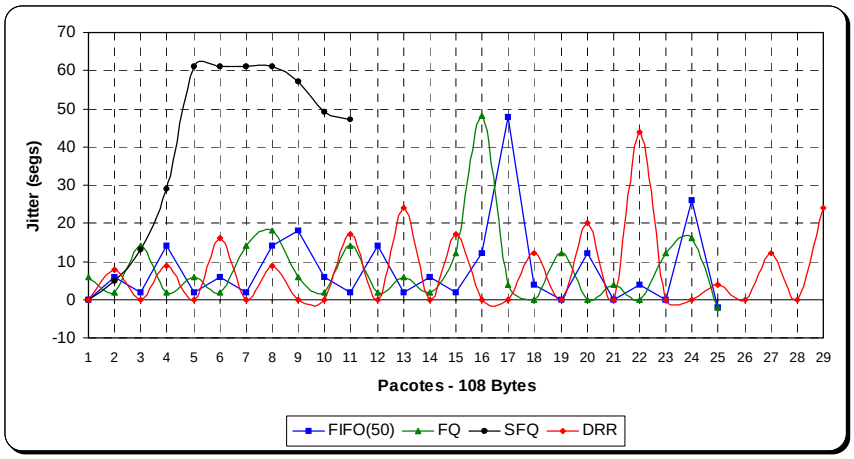

Figura E2 – *Jitter* pacotes 104 bytes / Esquemas de filas

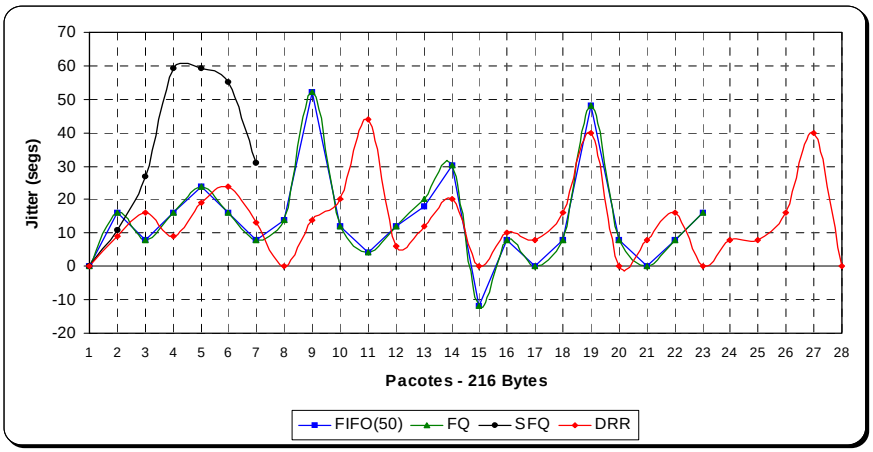

Figura E3 – *Jitter* pacotes 216 bytes / Esquemas de filas

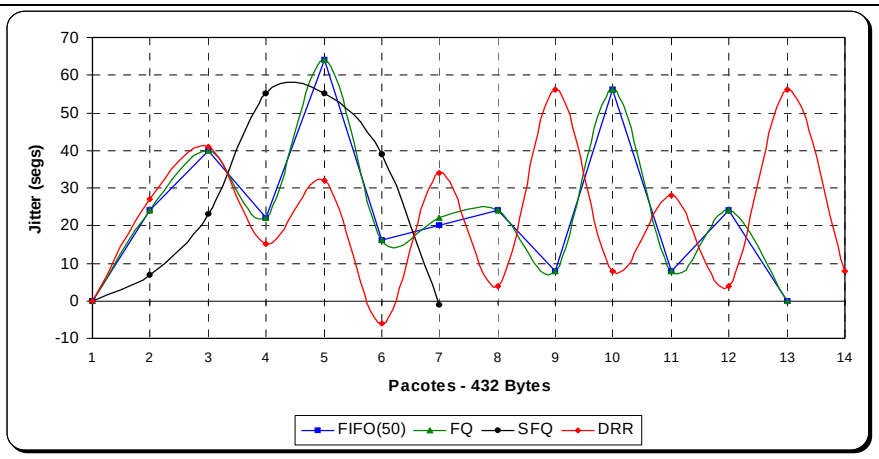

Figura E4 – *Jitter* pacotes 432 bytes / Esquemas de filas

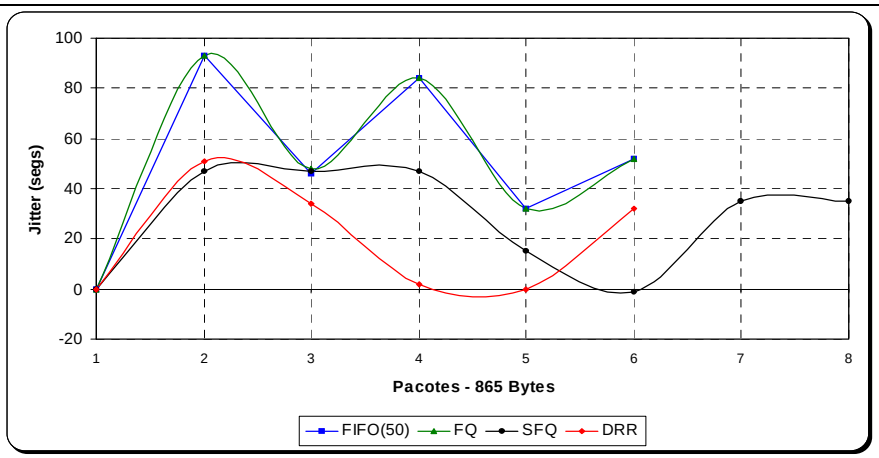

Figura E5 – *Jitter* pacotes 865 bytes / Esquemas de filas

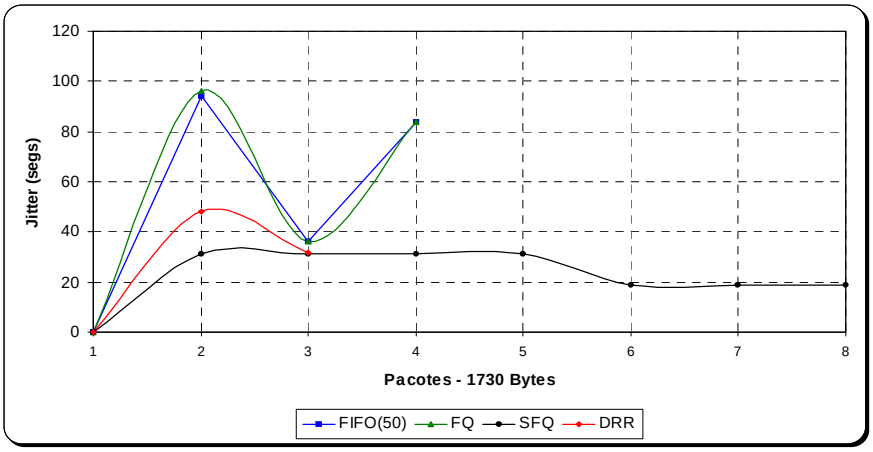

Figura E6 – *Jitter* pacotes 1730 bytes / Esquemas de filas

ANEXO F – Gráficos – Distribuição Estatística

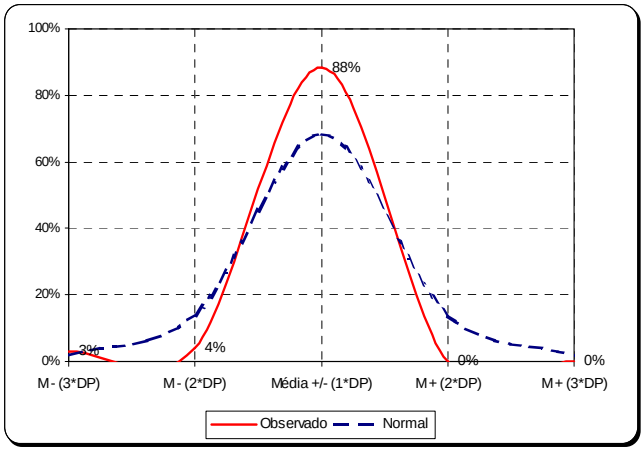

Figura F1 – Distrib. Estatística – FIFO (30)

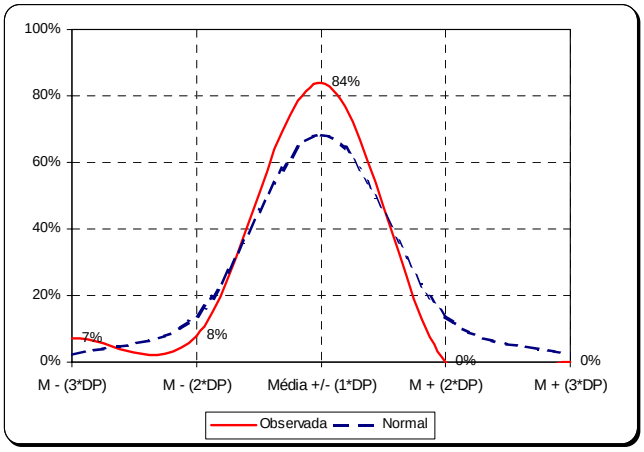

Figura F2 – Distrib. Estatística – FIFO (50)

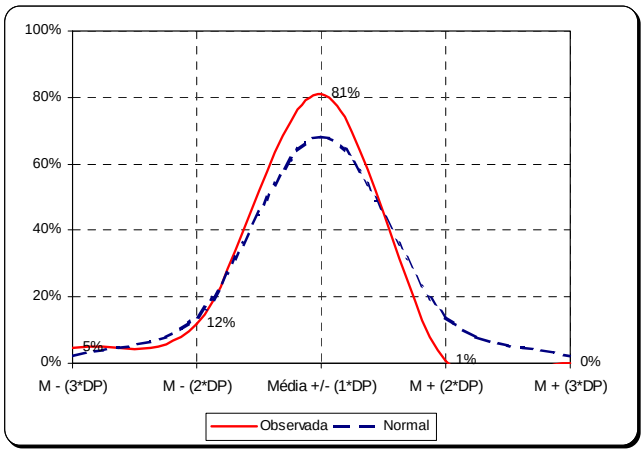

Figura F3 – Distrib. Estatística - SFQ

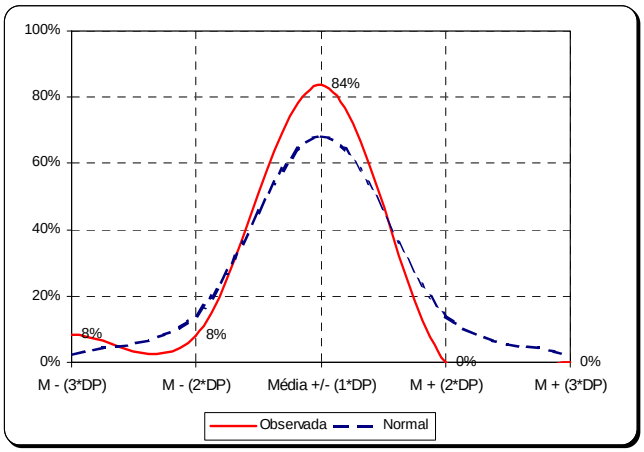

Figura F4 – Distrib. Estatística - FQ

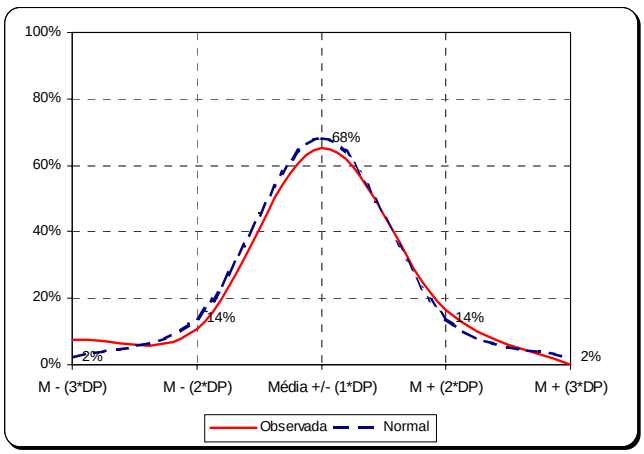

Figura F5 – Distr. Estatística - DRR