Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Dissertação de Mestrado

Um Protocolo para Gerência de *Handoff* em Redes Pessoais Sem Fio para Aplicações de Tempo-Real

Loreno Feitosa de Oliveira

Campina Grande, Paraíba, Brasil

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Um Protocolo para Gerência de *Handoff* em Redes Pessoais Sem Fio para Aplicações de Tempo-Real

Loreno Feitosa de Oliveira

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Angelo Perkusich, DSc.
(Orientador)
Leandro Dias da Silva, DSc.
(Orientador)

Campina Grande, Paraíba, Brasil

O48p

2007 Oliveira, Loreno Feitosa de.

Um Protocolo para Gerência de *Handoff* em Redes Pessoais Sem Fio para Aplicações de Tempo-Real / Loreno Feitosa de Oliveira. - Campina Grande, 2007.

115f: il. color.

Dissertação (Mestrado em Ciência da Computação), Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Referências.

Orientadores: Dr. Angelo Perkusich, Dr. Leandro Dias da Silva.

1. Embutido (*embedded*). 2. Tempo-Real. 3. Camada Física. 4. *Handoff*. 5. Computação Pervasiva. 6. Computação Ubíqua. 7. Computação Móvel. 8. Multimídia. I. Título.

CDU – 004.031.43/.6(043)

"UM PROTOCOLO PARA GERÊNCIA DE *HANDOFF* EM REDES PESSOAIS SEM FIO PARA APLICAÇÕES DE TEMPO-REAL"

LORENO FEITOSA DE OLIVEIRA

DISSERTAÇÃO APROVADA EM 31.08.2007

PROF. ANGELO PERKUSICH, D.Sc
Orientador

PROF. LEANDRO DIAS DA SILVA, D.Sc
Orientador

PROF. MARCO AURÉLIO SPOHN , Ph.D
Examinador

PROF. ANTONIO MARCUS NOGUEIRA LIMA, Dr.
Examinador

CAMPINA GRANDE – PB

Resumo

Redes pessoais sem fio, WPANs (*Wireless Personal Area Networks*), são redes de curto alcance, em torno de 10 m, cujo centro é o usuário. O cenário de uso geral é aquele onde dispositivos dentro da área de cobertura da WPAN comunicam-se diretamente entre si ou com recursos do mundo exterior (recursos fora da WPAN) através de pontos de acesso que ofereçam esse tipo de encaminhamento de dados.

WPANs vêm ganhando atenção nos últimos anos principalmente devido ao surgimento de novas tecnologias de transmissão sem fio que viabilizam este tipo de rede, particularmente Bluetooth. De fato, o desenvolvimento das WPANs confunde-se com o desenvolvimento do Bluetooth, que tem sido usado como ponto de partida em diversos estudos e protótipos nesta área. Sendo a mobilidade de usuários a principal característica das WPANs, um número de questões surge quando se pensa no desenvolvimento de aplicações direcionadas para esse novo paradigma. Uma das principais se refere à gerência de *handoff*. *Handoff* é o processo pelo qual conexões, rotas de dados, e estados associados à provisão de algum serviço são transferidos entre pontos de acesso à medida que o usuário se move entre suas áreas de cobertura.

Apesar de seu alinhamento com o modelo de rede das WPANs, Bluetooth não possui facilidades para o gerenciamento de *handoffs* além de suas operações padrão para localização e conexão com dispositivos próximos; *inquiry* e *paging* respectivamente. Adicionalmente, o tráfego de dados dessas operações pela interface Bluetooth possui prioridade sobre o tráfego de dados das aplicações do usuário. Essa característica possui especial impacto sobre um tipo particular de aplicações: aquelas que demandam transferências de dados em tempo-real, como aplicações de *streaming*. Ao tornar o canal sem fio indisponível para o tráfego de dados, seja pela temporária perda total de conexão com pontos de acesso durante *handoffs* ou por preempção da interface para as operações citadas, aplicações de tempo-real têm seus desempenhos comprometidos devido à quebra de requisitos temporais associados às suas trocas de dados.

Nesse contexto, neste trabalho é proposto um protocolo para gerência de *handoffs* em WPANs Bluetooth. O protocolo apresentado é voltado para o uso com aplicações que demandam transferências de dados em tempo-real, sendo demonstrada nesse trabalho sua adequação para esse tipo de aplicação. O protocolo apresentado foi projetado levando-se em consideração as limitações dos potenciais dispositivos clientes (pequenos dispositivos móveis com pouco poder de processamento, pouca memória, largura de banda restrita, etc). Assim, são transferidas para os pontos de acesso todas as atividades relativas às transições entre pontos de acesso dos dispositivos móveis. O protocolo apresentado descarta ainda a necessidade de sinalizações ou quaisquer outras trocas de mensagens entre dispositivos móveis e pontos de acesso durante os *handoffs*. Por utilizar apenas operações padronizadas do Bluetooth, viabiliza-se seu uso junto com qualquer dispositivo programável equipado com interface Bluetooth de acordo com a especificação, sendo portanto dispensada a necessidade de, por exemplo, modificar a pilha Bluetooth dos dispositivos.

Abstract

Wireless personal area networks (WPANs), are a mobile short range wireless network, with typical range of 10 meters, where the user is the center. The general usage scenario is where devices within the WPANs range communicate directly to each other or with resources from the external world (outside the WPAN) through access points which offer routing service.

WPANs have been gaining attention over the last few years mainly due the emergence and popularization of novel wireless communication technologies that enable this kind of network, notably Bluetooth. In fact, the development of WPANs is closely related to the development of Bluetooth, which has been used as starting point to several studies and prototypes in this field. As the user mobility is the main feature of WPANs, a number of questions arise when developing applications targeted to this new paradigm. One of the most important refers to the handoff management. Handoff is the process through which network connections, routes, and states associated to services in course are seamlessly transferred between access points as the user moves through their coverage areas.

Despite its alignment with the WPAN's network model, Bluetooth has no facilities for aiding the management of handoffs besides its standard operations for querying nearby devices and connect to them, inquiry and paging respectively. Moreover, the data traffic of these operations has priority over user applications' data traffic. This property has special impact for a particular kind of application: those that require real-time data transfer, such as streaming applications. When the wireless channel is unavailable for data transfers, with temporary connection loss with access points during handoffs, or interface preemption for the inquiry and paging operations, real-time applications have their performance compromised in consequence of violation of temporal requirements.

In this work, a protocol for managing handoffs in Bluetooth-based WPANs is presented. The protocol is focused on the use of applications that demand real-time data transfers. The adequacy of the protocol to this kind of application is analyzed through a case study for an audio streaming application. The protocol is designed focusing on the limitations of potential client devices (small portable devices with limited computational power, memory, bandwidth, battery life, etc). Therefore, all the handoff management operations are transferred to access points. There is no need of signaling or any other kind of coordination or message exchange between access points and mobile devices during handoffs. Due to the use of standardized Bluetooth operations, any programmable device with a standard complaint Bluetooth interface can be used without changes on any underlying software layer, such as the Bluetooth stack. It is also presented a formal modelling and validation of the protocol to ensure it behaves according to its specification. The formal model is important to understand the protocol, unanbiguous documentation, and to easy the validation of changes and extension by automatic simulation and proof of properties.

Agradecimentos

À Deus, por me dar a força necessária para continuar em frente mesmo nos momentos de maior dificuldade.

Aos meus pais, Evando e Zélia, por todos esforços para me proporcionar uma educação de qualidade. Agradeço a eles por todas as oportunidades dadas, pela compreensão por minhas ausências durante o mestrado, pelos ensinamentos, cobranças, incentivo, e confiança.

Aos meus orientadores, Angelo e Leandro, pelo incentivo, apoio, pelas discussões enriquecedoras, pela paciência diante de minhas dificuldades, pela sabedoria da orientação, e pelos recursos disponibilizados.

À minha namorada, Ana Emília, por todo seu apoio, suas palavras de incentivo, pela ajuda com assuntos administrativos após minha mudança de cidade, por sua compreensão nos momentos difíceis e por dividir comigo as conquistas e momentos felizes. Te agradeço muito por compreender minhas ausências durante este longo caminho.

Aos meus colegas de trabalho, e superiores em diferentes momentos, Rômulo, Yuri, e Juliana, por compreender a minha dificuldade de conduzir um trabalho de mestrado à distância. Os agradeço por todos os incentivos para que eu pudesse cumprir com minhas obrigações acadêmicas.

À todos os amigos que, de alguma forma, me incentivaram ou me deram força nos momentos mais problemáticos, especialmente os amigos de apartamento Aliandro, Danilo, e toda a turma do projeto Mercosul na Dataprev.

Às pessoas do laboratório Embedded, em particular a turma da sala do projeto COMPOR, Glauber, Fred, Milena, Hyggo, Emerson e Leandro Sales, que faziam daquela sala uma das mais divertidas do laboratório. Um agradecimento especial para Kyller, por sua valorosa ajuda com a modelagem em CPN, e a Danilo e Marcus Morais, por todas as suas dicas e palpites sobre meus problemas e dúvidas sobre Bluetooth.

Ao grande guru Odilon, por todas as dicas e lições sobre C++. Dificilmente eu teria conseguido implementar o protótipo usado nos testes sem a assistência desse experiente programador e sábio profissional.

Às pessoas que compõem a COPIN, em particular Aninha, pessoa espetacular, amiga, compreensiva, atenciosa e prestativa.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Em 1991 Mark Weiser expôs ao mundo sua visão do que seria a computação no século XXI [74], originalmente chamada de *computação ubíqua* e hoje mais conhecida como *computação pervasiva* [57]. Weiser vislumbrou um mundo no qual diferentes tecnologias estariam embarcadas em objetos do dia a dia como roupas, eletrodomésticos e automóveis, criando assim *ambientes inteligentes* onde esses objetos seriam capazes de colaborar entre si para oferecer serviços personalizados às pessoas.

As idéias de Weiser eram, entretanto, avançadas demais para a época na qual foram inicialmente concebidas, considerando a disponibilidade de *hardware* e *software* daquela época. Em 1991, os dispositivos computacionais disponíveis eram grandes demais, caros demais e com pouco poder de processamento e armazenamento, se comparados aos dispositivos disponíveis atualmente. Estes eram, sem dúvida, obstáculos à realização da computação pervasiva. Além disso, outras restrições tecnológicas tais como a ausência de redes sem fio, baterias com baixa autonomia e ausência de softwares adequados [25], tornavam impossível a concretização das idéias de Weiser.

Porém, os avanços tecnológicos dos últimos anos proporcionaram a miniaturização de componentes eletrônicos, possibilitando o desenvolvimento de dispositivos cada vez menores, ao mesmo tempo que o poder dos processadores aumenta a cada nova geração. O espaço de armazenamento dos computadores aumentou de poucos *kilobytes* em 1991 para centenas de *gigabytes*, comuns hoje em dia. Baterias para dispositivos móveis tornaram-se menores e com maior autonomia. Tecnologias para comunicação sem fio como *Wi-Fi* [72] e *Bluetooth* [64] surgiram e amadureceram, enquanto novas tecnologias como *ZigBee* [79] e

WiBree [75], continuam a surgir. Essa evolução veio acompanhada da gradual redução dos custos de desenvolvimento e produção em massa de dispositivos que utilizam essas tecnologias, tornando-os mais acessíveis e cada vez mais presentes na sociedade. Graças a essas novas tecnologias, dispositivos como sensores, telefones celulares, *smartphones*, *notebooks*, *Internet Tablets*, e PDAs [1], surgiram e/ou evoluíram desde 1991. Esses dispositivos são hoje as peças centrais para a concretização da computação pervasiva.

Paralelamente ao desenvolvimento da computação pervasiva, um novo paradigma de rede, focado na comunicação sem fio de curto alcance, vêm atraindo pesquisadores e indústria nos últimos anos. Redes pessoais sem fio [49], WPANs (*Wireless Personal Area Networks*), são redes de curto alcance, em torno de 10 m, cujo centro é o usuário. O cenário de uso geral é aquele onde dispositivos dentro da área de cobertura da WPAN possam comunicar-se diretamente entre si ou com recursos do mundo exterior (recursos de fora da WPAN) através de dispositivos que ofereçam esse encaminhamento de dados. Possíveis aplicações para WPANs incluem monitoração médica [29; 28], ambientes inteligentes [45; 12], e entretenimento [33].

A pesquisa em torno de WPANs intercala-se com o desenvolvimento do Bluetooth. De fato, o Bluetooth têm sido o ponto de partida para diversos estudos e protótipos no contexto de WPANs [27; 26]. Originalmente desenvolvido para substituir cabos na troca de dados de curto alcance entre dispositivos, o Bluetooth está cada vez mais presente em dispositivos eletrônicos. Dentre as razões que o tornam, atualmente, a tecnologia ideal para o desenvolvimento de WPANs pode-se citar seu baixo consumo de energia, baixo preço, e presença de mercado. Sendo a mobilidade de usuários a principal característica das WPANs, e dadas as características dos potenciais dispositivos clientes (pequenos dispositivos móveis com pouca capacidade de processamento, pouca memória, largura de banda restrita, etc), um número de dificuldades são identificadas quando se pensa no desenvolvimento de aplicações direcionadas para WPANs [49], como por exemplo [43; 44]:

- dispositivos clientes, ainda, com diversas restrições computacionais e de usabilidade;
- adequação de protocolos e formatos de dados para uso com tais dispositivos;

[1] *Personal Digital Assistant*

- ambientes com mudanças constantes na oferta de serviços, sejam estas na disponibilidade de algum serviço ou na sua qualidade;
- intermitências na conexão entre provedor e consumidor dos serviços, devido movimentação dos dispositivos cliente, e a eventual necessidade de aplicações poderem lidar com esses períodos de queda de conexão;
- para serviços mais complexos, transferência transparente de serviços entre diferentes pontos de acesso sem fio à medida que usuários se movem através de suas áreas de cobertura (gerência de *handoff*);
- segurança e privacidade para dados trocados via conexões sem fio e para o acesso/uso de informações sobre o perfil dos usuários;
- necessidade de adaptar conteúdo em tempo de execução devido à heterogeneidade de dispositivos e/ou mudanças que possam impactar na qualidade do serviço oferecido;
- tirar proveito da alta disponibilidade de recursos das redes cabeadas, como conexões de rede mais velozes, espaço de armazenamento e capacidade computacional, em benefício dos pequenos dispositivos móveis clientes.

Dispositivos móveis dentro da área de cobertura da WPAN podem comunicar-se diretamente entre si ou com recursos do mundo exterior (recursos fora da WPAN) através de pontos de acesso que ofereçam esse tipo de encaminhamento de dados. Para um uso contínuo e transparente dos serviços oferecidos pelos pontos de acesso, ou por provedores acessados através destes, faz-se necessária a adequada gerência de *handoff*. *Handoff* é o processo pelo qual conexões, rotas de dados, e estados associados à provisão de algum serviço são transferidos entre pontos de acesso à medida que o usuário se move entre suas áreas de cobertura [78; 63]. Gerenciar *handoffs* significa prover os meios pelos quais essas trocas de ponto de acesso são tornadas transparentes para usuários finais e/ou aplicações mais altas na pilha de *software*.

Soluções para gerenciamento de *handoffs* são desenvolvidas a partir de pilares como: i) características da tecnologia de rede sem fio utilizada, e ii) características/requisitos das aplicações e casos de uso finais. Ou seja, o projeto de uma solução para gerência de *handoffs* pode ser mais ou menos árduo conforme a própria tecnologia de rede sem fio ofereça

ou não facilidades para o processo de *handoff*, por exemplo, suporte para medida precisa da localização de dispositivos ou reserva antecipada de recursos no novo ponto de acesso. O projeto de uma solução para gerenciamento de *handoffs* é fortemente influenciado pelas características/requisitos das aplicações finais do sistema. Aplicações com alguma tolerância a eventuais perdas totais de conexão e/ou oscilações na qualidade da conexão, como e-mail ou navegação pela *web*, tendem a não acrescentar muita complexidade à solução final para gerência de *handoffs*. Por outro lado, aplicações com fortes requisitos quanto à disponibilidade de conexão e de sua qualidade trazem complexidade extra ao desenvolvimento da solução de *handoff*. Aplicações de tempo-real, em especial aquelas que demandam a transferência de dados via rede em tempo-real (vide Seção 2.4.1), são aplicações com limites bem definidos quanto aos intervalos para envios/chegada de dados. Perdas totais de conexão ou grandes oscilações na qualidade do canal sem fio tendem a comprometer a eficácia dessas aplicações por quebrar requisitos temporais associados às suas trocas de dados. Assim, protocolos bem sucedidos para um determinado cenário (tecnologia de rede + aplicações alvo), podem provar-se inadequados em outros cenários.

Neste trabalho, o interesse está na gerência de *handoffs* entre dispositivos clientes e pontos de acesso em WPANs baseadas em Bluetooth. Mais precisamente, estamos interessados num protocolo de *handoff* que viabilize o uso de aplicações que demandam a troca de dados em tempo-real pelas interfaces Bluetooth. Ao mesmo tempo que esse tipo particular de aplicação abre caminho para o desenvolvimento de uma série de novas aplicações em ambientes inteligentes, tais quais as descritas no Apêndice B, por outro lado aumenta-se também os requisitos sobre a solução final. Conforme discutido anteriormente, esse *trade off* é conseqüência das características desse tipo de aplicação de tempo-real, que demanda o envio/recepção de dados com alguma pontualidade, e são sensíveis à oscilações em parâmetros como taxa de perda de dados ou dados fora de tempo.

1.1 Problemática e Relevância da Pesquisa

Por ser a mobilidade de usuários e dispositivos a principal característica das WPANs, aplicações desenvolvidas para esse tipo de ambiente devem ser, desde seu projeto, preparadas para lidar com problemas relacionados com a constante intermitência na disponibilidade de

recursos. Intermitências na disponibilidade de recursos incluem, por exemplo, a entrada e saída de dispositivos na WPAN, levando ou trazendo consigo serviços de interesse de outros dispositivos da rede sem fio. A forma de se lidar com essa característica das WPANs depende dos requisitos das aplicações finais. Enquanto algumas aplicações, como *e-mail* ou sincronização de dados, tendem a ser mais tolerantes à falta de conexão com a Internet ou redes locais, aplicações baseadas na troca de dados em tempo-real, como aplicações de *streaming* (Seção 2.4.1), podem ter seus desempenhos fortemente comprometidos diante da falta, mesmo que temporária, de conexão com redes externas. Para aplicações cujos requisitos se encaixem no segundo caso, o suporte de um protocolo de *handoff* é requisito para seu funcionamento adequado em WPANs.

Apesar de sua adequação para comunicação *ad-hoc* e alinhamento com o modelo de rede das WPANs, Bluetooth foi inicialmente desenvolvido para substituir cabos, provendo um canal sem fio para a comunicação entre dispositivos próximos. Como tal, não possui facilidades para o gerenciamento de *handoffs* além de suas operações padrão para localização e conexão com dispositivos próximos (*inquiry* e *paging*, respectivamente). Além disso, o tráfego de dados dessas operações possui prioridade sobre o tráfego de dados das aplicações pela interface Bluetooth. Ou seja, no contexto de aplicações de tempo-real, também é necessário ponderar o uso dessas operações sobre o desempenho das aplicações. O foco da atenção no projeto do protocolo de *handoff*, portanto, está em definir as configurações e procedimentos necessários para a troca de conexões entre pontos de acesso com o mínimo de impacto sobre o desempenho das aplicações, particularmente, aquelas que demandam a troca de dados em tempo-real.

O interesse por protocolos de *handoff* em Bluetooth não é recente, onde propostas têm sendo publicadas desde 2002 [13; 30; 20; 11; 35; 10; 31; 70], sendo a problemática em torno desse tema já discutida pelo menos desde 2000 [69; 3]. Entretanto, dentre as propostas atualmente conhecidas, nenhuma possui experimentos suficientes para comprovar sua adequação para o uso com aplicações de tempo-real. Por exemplo, dentre os trabalhos mais expressivos nessa área, a proposta de Sang-Hun Chung et al. [13] inclui o uso nos dispositivos móveis de operações não padronizadas pelo Bluetooth, o que pode tornar inviável sua implantação nesse tipo de dispositivo. Os trabalhos de Aman Kansal [30] e Ming-Chiao Chen [11], por outro lado, baseiam-se apenas em operações padronizadas da tecnologia. En-

tretanto, no primeiro trabalho o procedimento de *handoff* apenas é disparado após a perda total de conexão do dispositivo. Além disso, os dispositivos móveis precisam estar no modo contínuo de escuta por tentativas de conexão (vide Seções 2.5.3 e A). Isso significa que o dispositivo têm sua largura de banda comprometida, o que causa impacto direto no desempenho de aplicações de tempo-real.

Para evitar a perda total de conexão do dispositivo, na proposta de Ming-Chiao Chen novas conexões são estabelecidas antes que as anteriores sejam perdidas. Os pontos negativos deste trabalho incluem o papel do dispositivo móvel no procedimento de *handoff* (estes dispositivos são responsáveis por monitorar a necessidade e disparar processos de *handoff*), e o número de conexões simultâneas que precisam ser mantidas pelo dispositivo. Ou seja, de acordo com a solução descrita, cada dispositivo cliente está conectado ao mesmo tempo não só em um ponto de acesso específico, mas neste ponto de acesso e em todos os seus vizinhos ao mesmo tempo. Esse aumento no número de conexões reduz a escalabilidade do sistema, pois mesmo que não esteja em movimento um dispositivo ocupa recursos de todas os pontos de acesso próximos. Além disso, não é descrito no trabalho o protocolo pelo qual dispositivos podem se mover através das áreas de cobertura dos pontos de acesso enquanto um fluxo de dados com características de tempo-real é transferido pela interface Bluetooth. O modelo de movimentação assumido neste trabalho assume um padrão de movimento de usuários simplista onde, uma vez iniciado o procedimento de *handoff*, o usuário/dispositivo não mais pode mudar de direção. Comum a todos esses trabalhos é o fato de que em nenhum deles foram apresentadas comprovações sobre sua eficiência no contexto de aplicações de tempo-real.

1.2 Objetivos

O objetivo deste trabalho é propor um protocolo de *handoff* para WPANs baseadas em Bluetooth e demonstrar sua adequação para o uso com aplicações que demandam a troca de dados via rede em tempo-real. A solução final deve levar em consideração a escassez de recursos dos potenciais dispositivos clientes, evitando portanto sobrecarregar esses dispositivos com tarefas relacionadas à gerência de *handoffs*. O protocolo deve, por fim, basear-se apenas em operações padronizadas pela especificação do Bluetooth, viabilizando assim seu uso com

qualquer dispositivo programável equipado com uma interface Bluetooth padronizada.

1.3 Estrutura da Dissertação

O restante deste trabalho está estruturado da seguinte maneira:

- No **Capítulo 2** são introduzidos os principais temas que norteiam este trabalho.
- No **Capítulo 3** é apresentada a proposta deste trabalho para atender os objetivos descritos na Seção 1.2.
- Em seguida, no **Capítulo 4** são des-critos aplicações, cenários, modelos, e resultados de experimentos, usados para a validação do protocolo apresentado no Capítulo 3.
- Por fim, no **Capítulo 5** são feitas as considerações finais sobre o trabalho. São apresentadas as conclusões e planos de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são introduzidos os principais temas relacionados ao trabalho apresentado nesta dissertação. Na Seção 2.1 é apresentada uma visão geral sobre a computação pervasiva, enquanto na Seção 2.2 são discutidas as principais tecnologias para transmissão sem fio, com enfoque em WPANs. Os principais conceitos envolvendo *handoffs* e gerência de *handoffs* são discutidos na Seção 2.3. Aplicações de tempo-real são introduzidas na Seção 2.4, juntamente com uma discussão sobre aplicações de *streaming* devido ao seu particular alinhamento com o trabalho aqui apresentado. Por fim, na Seção 2.5 é feita uma introdução sobre a tecnologia Bluetooth. São discutidas suas principais características, funcionamento das camadas da pilha padrão, e as operações de *inquiry* e *paging* (principais funções no contexto do trabalho apresentado), além de resultados de experimentos reais que ilustram o impacto dessas operações sobre o fluxo de dados em tempo-real.

2.1 Computação Pervasiva

A visão geral sobre a computação pervasiva é aquela onde a computação é embarcada em objetos do dia-a-dia, como roupas, cômodos/ambientes, automóveis e eletrodomésticos. Juntos, estes objetos e ambientes podem colaborar entre si, através da troca de diferentes informações contextuais, em benefício das pessoas [74; 58]. Por meio do uso da proatividade, de informações atuais sobre o contexto dos usuários e de dados sobre os seus perfis, serviços personalizados podem ser oferecidos às pessoas, no lugar e na hora certos.

A computação pervasiva realiza-se através dos ambientes inteligentes, ou ambientes per-

vasivos [58; 24]. Estes ambientes são caracterizados pela interação entre dispositivos embarcados, dispositivos móveis, infra-estrutura cabeada e usuários. A forma mais comum, e mais simples, para usuários interagirem com ambientes inteligentes é através de dispositivos móveis pessoais, como PDAs ou *smartphones*, que podem carregar informações sobre seus perfis. Pequenos subconjuntos destas informações podem ser passados, sem intervenção humana, para provedores de serviços em ambientes pervasivos assim que o usuário se aproxima ou entra no ambiente. De posse dessas informações, estes provedores podem então adequar seus serviços aos desejos/necessidades de cada usuário [53]. Alternativamente, o próprio ambiente pode já conhecer as preferências dos seus usuários, e apenas consultá-las quando a presença de alguém conhecido é percebida através de sensores [59]. Independente de como ambientes inteligentes tomam ciência dos perfis dos usuários ou como eles detectam suas presenças, o princípio básico da computação pervasiva é tornar a computação invisível à percepção humana, ou pelo menos torná-la pouco evidente. Em outras palavras, usuários devem se beneficiar de decisões tomadas por estes ambientes sem que para isso precisem ser consultados a todo momento sobre o que eles gostariam, e como gostariam, que o ambiente fizesse algo por eles [58].

A computação pervasiva possui um forte relacionamento ancestral com, pelo menos, duas outra áreas da computação [57]: *sistemas distribuídos* e *computação móvel*. Vários dos problemas com os quais a computação pervasiva precisa lidar podem ser mapeados em problemas já identificados e estudados nestas áreas. Da área de sistemas distribuídos é herdado todo o conhecimento para lidar, por exemplo, com comunicação remota, tolerância a falhas, alta disponibilidade de recursos, acesso remoto a informações e segurança. Da computação móvel aproveitam-se, por exemplo, as experiências em acesso a redes cabeadas via dispositivos móveis, sistemas projetados para economizar energia de baterias e localização de dispositivos móveis.

Entretanto, mesmo aproveitando soluções desenvolvidas em anos de pesquisas nestas áreas, pesquisadores da área de computação pervasiva ainda precisam resolver problemas inerentes aos objetivos deste novo paradigma. Tornar-se uma "tecnologia que desaparece" requer que a computação pervasiva se integre na vida das pessoas de tal maneira que estas continuem desempenhando suas tarefas corriqueiras, desta vez assistidas por algum sistema computacional, mas sem alterar a forma como as tarefas eram realizadas antes, sem auxílio

computacional. Ou seja, os sistemas computacionais se adaptam às pessoas, e não o contrário. A partir do momento que movimento é parte integral da vida cotidiana das pessoas, tais sistemas precisam dar suporte à mobilidade de seus usuários. Caso contrário, usuários vão tomar ciência da tecnologia assim que perceberem sua ausência ao se moverem de um cômodo para outro, por exemplo.

Dentre os novos problemas com os quais a computação pervasiva precisa lidar, podem ser destacados [57]:

- **Escalabilidade**

  À medida que ambientes inteligentes tornam-se mais sofisticados, a quantidade de interação entre os dispositivos móveis dos usuários e os ambientes tende a aumentar. Isto traz diversas implicações para os projetos de tais sistemas; e.g., largura de banda limitada, consumo de energia e capacidade de absorção de clientes. O problema torna-se mais complicado quando considera-se um número cada vez maior de usuários nestes ambientes.

- **Heterogeneidade**

  Dispositivos móveis modernos são extremamente diferentes entre si tanto em termos de configuração quanto em termos de recursos. Eles podem ter telas de diferentes tamanhos, com resoluções e número de cores distintos. Estes dispositivos podem ainda diferenciar-se pelo tipo de tecnologia sem fio usada, capacidade de processamento, quantidade de memória, método de entrada, sistema operacional e plataforma de *hardware*. Criar sistemas capazes de lidar com tamanha heterogeneidade de dispositivos clientes pode ser uma tarefa árdua a depender das características do serviço anunciado. Por exemplo, provedores de serviços que necessitem instalar nos dispositivos móveis alguma aplicação cliente específica, provavelmente terão que lidar com diferentes versões da mesma aplicação cliente, como forma de lidar com a heterogeneidade dos clientes em potencial, o que pode tornar proibitiva a implantação de tal serviço [32].

- **Invisibilidade**

  No cenário ideal vislumbrado por Weiser a computação se torna invisível à percepção

humana. Na prática, atualmente, o que se buscam são sistemas que requerem um mínimo de intervenção/distração humana para realizarem seu trabalho. Humanos podem, eventualmente, intervir em decisões tomadas por algum sistema como forma de "calibrar" o sistema para uma determinada situação/condição. Entretanto o que se procura são maneiras pouco intrusivas para a disponibilização e ajuste de serviços. Para atender continuamente às expectativas das pessoas, ambientes e objetos devem ser capazes de se auto-ajustarem sem a necessidade de exigirem a atenção das pessoas. Auto-ajustes e auto-aprendizado podem ser implementados em diferentes níveis. Por exemplo, dispositivos com conexão sem fio podem se auto-configurar quando na presença de algum ponto de acesso. Isso retira do usuário a tarefa maçante de buscar pontos de acesso, definir endereço de rede, mascaras de rede e *gateways* padrão, por exemplo.

- **Ciência e Gerência de Informações sobre o Contexto**

  Outro requisito básico da computação pervasiva é a necessidade de ambientes inteligentes perceberem informações contextuais e raciocinarem sobre elas de maneira proativa. Esta capacidade de percepção, ou ciência de contexto [16], é uma característica intrínseca de ambientes inteligentes. Mas implementar esse tipo de percepção traz uma séria de complicações, tais como: localização de dispositivos e usuários, processamento de informações em tempo-real, e mesclagem de dados vindos de diferentes fontes (e.g. sensores e perfis). Decisões tomadas automaticamente em ambientes inteligentes devem ser precisas e baseadas em dados atuais, caso contrário os erros cometidos podem implicar na distração dos usuários.

Na Figura 2.1 é ilustrado o relacionamento entre a computação pervasiva e as áreas de sistemas distribuídos e computação móvel. Na figura não é expressa a relação temporal entre as áreas, mas sim ligações lógicas. A computação pervasiva beneficiou-se de diversas experiências previamente adquiridas nos campos da computação móvel e sistemas distribuídos. A computação móvel, por sua vez, também aproveitou conhecimentos adquiridos pela pesquisa em torno de sistemas distribuídos. O que o símbolo multiplicador da figura quer dizer é que a complexidade das soluções para os mesmos problemas aumenta à medida que estas são adaptadas para novos requisitos, que não existiam quando as soluções foram originalmente aplicadas.

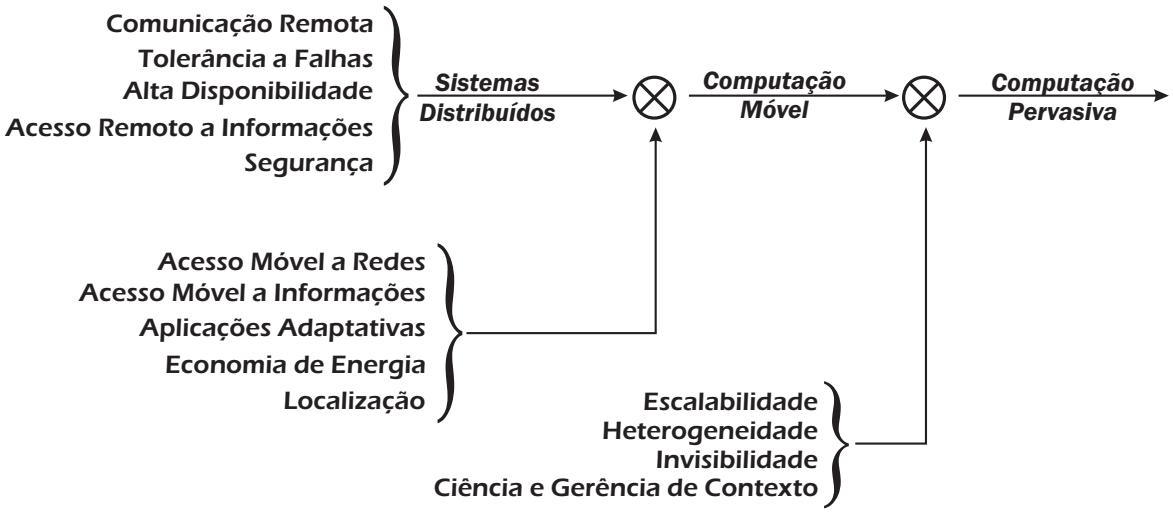


Figura 2.1: Relacionamento entre a computação pervasiva e as áreas de sistemas distribuídos e computação móvel (adaptado do trabalho de Mahadev Satyanarayanan em [58])

2.2 Tecnologias para Transmissão Sem Fio

Existe atualmente uma grande diversidade na oferta de tecnologias para comunicação sem fio, cada uma com suas próprias características e limitações e voltadas para um nicho de mercado específico. Tecnologias sem fio vêm sendo empregadas para diversas finalidades, como conectar computadores, monitoração de ambientes, troca oportunística de dados, e telefonia, para citar algumas. Devido à incessante demanda da sociedade por novos serviços, serviços personalizados, melhor desempenho e melhor qualidade na comunicação sem fio, a pesquisa em tecnologias para troca de dados sem fio atinge novos patamares a cada nova tecnologia anunciada.

Como os diferentes usos de tecnologias sem fio podem sugerir (telefones sem fio, telefonia celular, redes metropolitanas sem fio, redes sem fio pessoais, etc), cada tecnologia é direcionada para conjuntos bem definidos de aplicações. Adquirir e implantar uma infraestrutura GPRS [1], por exemplo, com certeza não é a melhor solução para o usuário doméstico que quer compartilhar acesso à Internet entre os computadores de sua residência. Portanto, desde seu projeto, cada nova tecnologia foca as necessidades e características dos cenários de uso finais, como aplicações, ambiente, e requisitos não funcionais. Durante essa fase de projeto, provavelmente o primeiro aspecto a ser levado em consideração na especificação de

[1] *General Packet Radio Service* [22]

uma nova tecnologia de rede é sua abrangência. Nesse contexto, as principais tecnologias para transmissão sem fio podem ser classificadas em três categorias [72]: redes de longo alcance sem fio (*Cellular Systems*); redes locais sem fio (WLANs [2]); e redes pessoais sem fio (WPANs [3]). Redes celulares são bem conhecidas do público desde o início de sua popularização na década de 80. Elas lidam, essencialmente, com o tráfego de voz e são direcionadas para telefonia [22]. Antenas de redes celulares podem atender clientes que estejam num raio de dezenas de quilômetros. WLANs podem ser vistas como extensões das já conhecidas redes locais, LANs, onde a troca de informações entre um subconjunto dos nodos da rede acontece via ondas de rádio, ao invés de cabos. Comumente, o alcance de redes WLAN gira em torno das dezenas de metros para ambientes internos e centenas de metros para ambientes externos. Por fim, por estarem no foco deste trabalho, as WPANs serão discutidas separadamente a seguir, na Seção 2.2.1.

2.2.1 WPANs

Redes pessoais sem fio, WPANs [50], são um conceito razoavelmente recente. Apesar de, por exemplo, o IEEE trabalhar na normatização de redes WPAN desde 1999 [72], apenas mais recentemente as discussões sobre esse tema tomaram força. O principal fator motivador para essa quebra de inércia foi a popularização do Bluetooth em diferentes dispositivos, desde simples celulares à computadores de mesa. Além disso, o modelo de rede das WPANs também possui uma boa relação com requisitos da computação pervasiva, que também vêm atraindo atenções e ganhando espaço nos últimos anos [58].

Redes WPAN são redes de abrangências ainda menores que redes WLAN, e têm no usuário o ponto central de uma área de cobertura de poucos metros (também conhecida como POS, ou *Personal Operating Space*). À medida que o usuário se move, sua rede WPAN move-se junto com ele (Figura 2.2). A idéia geral em torno de WPANs é que algum dispositivo móvel carregado pelo usuário estabeleça e desfaça conexões com outros dispositivos dentro de sua POS à medida que o usuário se move para mais próximo ou mais distante destes aparelhos. Tais dispositivos podem incluir tanto dispositivos que estejam fixos no ambiente, como impressoras, câmeras, e computadores de mesa, quanto outros dispositivos que

[2] *Wireless Local Area Networks*

[3] *Wireless Personal Area Networks*

também se encontram em movimento, como celulares, *smartphones*, tocadores de música e aparelhos de GPS [4] que, por exemplo, podem estar nos bolsos de outras pessoas.

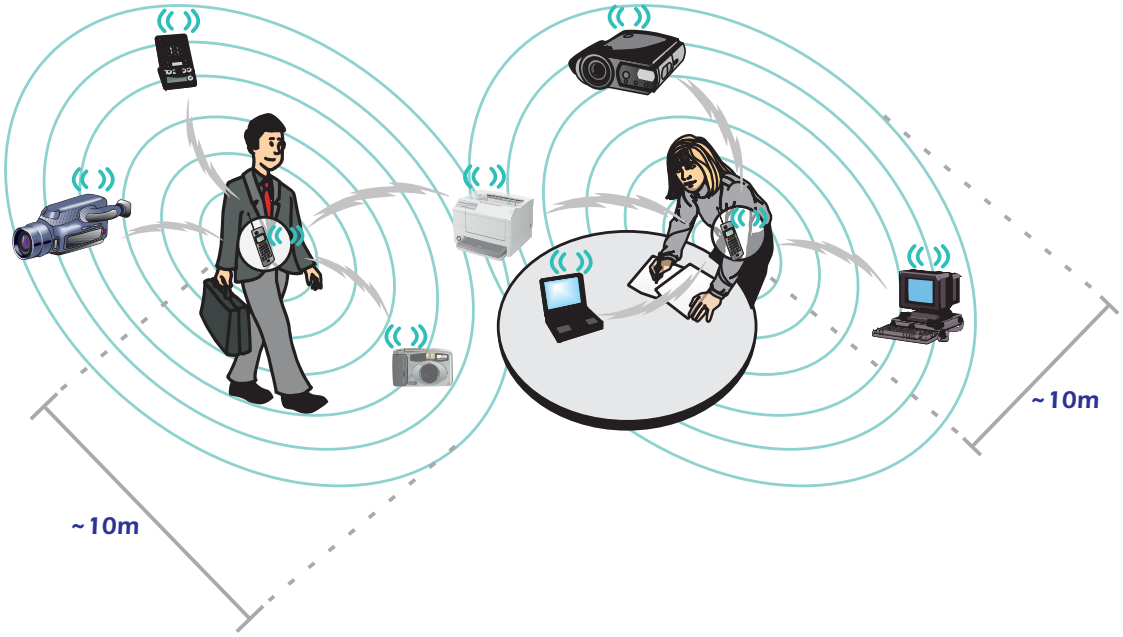


Figura 2.2: Redes pessoais sem fio - WPANs

Atualmente, a principal tecnologia usada em WPANs é Bluetooth. Devido às suas características, Bluetooth vem ganhando cada vez mais presença no mercado, onde equipa uma diversidade de equipamentos. Tais equipamentos incluem fones de ouvido, telefones celulares, telefones VoIP [5], relógios, auto-falantes, smartphones, PDAs, Internet Tablets, computadores portáteis, periféricos (e.g. teclados e *mouses*), impressoras, multifuncionais, e automóveis, dentre outros. Algumas características de Bluetooth que contribuem para essa presença de mercado incluem:

Largura de banda: As versões 1.1 e 1.2 de Bluetooth possuem largura de banda nominal de 1 Mbps. As versões mais atuais, 2.0 e 2.1, suportam até 3 Mbps de velocidade máxima.

Baixo custo: Como Bluetooth nasceu para ser um substituto de cabos, cada par de interfaces não podem custar muito mais que o cabo que elas substituem.

[4] *Global Positioning System*.
[5] *Voice over IP*.

Facilidade de uso: Conectar dois dispositivos via Bluetooth precisa ser tão simples quando ligar as extremidades de um mesmo cabo entre eles.

Baixo consumo de energia: Desde seu projeto Bluetooth foi direcionado para uso com dispositivos móveis que possuem autonomia limitada devido o uso de baterias. Assim, um objetivo fixo no projeto e evolução de Bluetooth é manter baixo o consumo de energia.

***Hardware* leve e pequeno:** Como Bluetooth foi desde o inicio projetado para ser incorporado em dispositivos móveis, seu *hardware* foi pensado para ser leve e pequeno. Desta forma, dificilmente o desenho de algum produto precisará ser alterado devido a incorporação de Bluetooth.

Confiança na transmissão de dados: A camada de enlace do Bluetooth garante entrega confiável de dados. Conexões sem fio Bluetooth precisam ser tão confiáveis quanto os cabos que elas substituem.

Apesar de sua grande presença no mercado, Bluetooth não é a única opção existente para implementar WPANs. Outras alternativas com fatias menores do mercado de WPANs, ou ainda em desenvolvimento, incluem ZigBee [79] e WirelessUSB [77]. Mesmo que direcionadas para o mercado de redes sem fio de curto alcance, tanto ZigBee quanto WirelessUSB destacam-se por possuirem características significativamente mais sofisticadas que Bluetooth. As principais características dessas tecnologias são enumeradas na Tabela 2.1.

	Freqüência	**Largura de Banda Máxima**	**Economia de Energia** [a]	**Alcance**	**Máximo Conexões Simultâneas**
Bluetooth 1.1/1.2	2.4 GHz	1 Mbps	⋆⋆	10 cm - 100 m	7
Bluetooth 2.0/2.1	2.4 GHz	3 Mbps	⋆⋆	10 cm - 100 m	7
ZigBee	2.4 GHz	259 Kbps	⋆⋆⋆⋆⋆	até 30 m	65.535
WirelessUSB	3.1 GHz - 10.6 GHz	480 Mbps	⋆⋆	até 50 m	127

[a] Quanto mais estrelas maior a economia de energia.

Tabela 2.1: Resumo das principais características das principais tecnologias para WPANs.

ZigBee, por exemplo, sobressai-se pelo seu consumo de energia extremamente baixo em relação ao Bluetooth, que por sua vez já é uma tecnologia com baixo consumo de energia. Em compensação possui largura de banda bem inferior à largura de banda do Bluetooth. WirelessUSB, por outro lado, possui ampla largura de banda enquanto mantém consumo de energia próximo ao consumo do Bluetooth. Em resposta a estes avanços de tecnologias similares, o Bluetooth SIG [6] vêm trabalhando na próxima versão do Bluetooth que, assim como WirelessUSB, usará a especificação UWB [7] [14] como sua camada de acesso ao meio, obtendo assim o mesmo desempenho que WirelessUSB. Além disso, já existem trabalhos em andamento para a integração entre Bluetooth e WiBree [8] [75] como forma de reduzir o consumo de energia, e largura de banda do Bluetooth para níveis próximos aos níveis praticados pelo ZigBee.

2.3 Gerência de *Handoff*

Mobilidade é a característica mais importante em sistemas baseados em tecnologias sem fio.

Handoff é o processo pelo qual conexões, rotas de dados e estados associados à provisão de algum serviço são transferidos entre pontos de acesso à medida que o usuário se move entre suas áreas de cobertura [78; 63]. Freqüentemente, o disparo do processo de *handoff* é iniciado pelo dispositivo móvel quando este atravessa um determinado limite de distância entre si próprio e seu atual ponto de acesso, ou ainda através de heurísticas baseadas na deterioração da qualidade do sinal da conexão atual.

Existem diversos critérios de classificação de estratégias de *handoff*. Um dos principais classifica os *handoffs* pelo número de conexões simultâneas com os pontos de acesso de origem e de destino no momento de sua troca (*hard handoffs* x *soft handoffs*). Em *hard handoffs*, também conhecidos como "quebrar antes de conectar", todas as conexões estabelecidas entre o dispositivo móvel e o ponto de acesso de origem precisam ser fechadas antes que sejam estabelecidas novas conexões entre o dispositivo e o novo ponto de acesso. Em

[6] *Bluetooth Special Interest Group* - Consórcio de fabricantes responsáveis pela especificação do Bluetooth.

[7] *Ultra Wide Band*. Especificação de camada física para comunicação de alta velocidade em curtas distâncias.

[8] Tecnologia complementar ao Bluetooth que especifica um modo ultra econômico de consumo de energia comparável ao consumo do ZigBee.

soft handoffs, também conhecidos como "conectar e depois quebrar", conexões simultâneas com os dois pontos de acesso são mantidas e usadas simultaneamente durante a troca. A probabilidade de perda de dados em *hard handoffs* é maior que em *soft handoffs* devido à temporária ausência total de conexão.

O projeto de infra-estruturas que suportem *soft handoffs* depende do uso de interfaces de rede que suportem múltiplas conexões simultâneas ou da disponibilidade de dispositivos multimodais [9] [55]. O impacto da perda temporária de conexão com a rede depende do tipo de aplicação usada nos dispositivos móveis. Aplicações onde o atraso nas respostas de requisições não compromete seu funcionamento: e.g., navegação pela Internet, ou aplicações com uso esporádico da conexão, como *e-mail*, são tolerantes a perdas temporárias de acesso à rede, desde que em seu projeto seu comportamento tenha sido configurado para situações assim (e.g., aguardar algum intervalo antes de tentar acessar a rede novamente ao invés de, por exemplo, acusar algum erro para o usuário e entrar num modo *off line*). Por outro lado, são comuns atualmente aplicações onde a menor interrupção no tráfego de dados pode prejudicar mais seriamente seu desempenho; e.g., aplicações de *streaming* (vide Seção 2.4.1).

Outra grande classe para classificação de *handoffs* diz respeito às tecnologias de rede usadas nos pontos de acesso de origem e de destino (*handoffs horizontais* x *handoffs verticais*). *Handoffs* onde a mesma tecnologia de rede sem fio é usada nos dois pontos de acesso são ditos horizontais: por exemplo, conexão com os dois pontos de acesso via Bluetooth. Caso as tecnologias de rede envolvidas sejam diferentes, como ponto de acesso de origem usando Bluetooth e de destino usando *Wi-Fi*, então o *handoff* é dito vertical. Classificações para *Handoffs* incluem ainda *handoffs* controlados pelo ponto de acesso ou pelo dispositivo móvel (*mobile controlled* x *network controlled*) [63], e *handoff* intra-célula, inter-célula, e inter-regiões (*micromobility* x *macromobility* x *global mobility*) [56].

Existem atualmente diversas abordagens e estratégias para gerenciamento de *handoff* em redes de comunicação de dados. Diferentes trabalhos propõem suas próprias infra-estruturas, desenvolvidas a partir de suposições distintas sobre seus cenários de uso finais. Algumas dessas suposições incluem tipo de tecnologia de rede sem fio [13; 52], tipo de aplicação [5; 52], tipo de dispositivo [42; 3], tipo de protocolo usado pelas aplicações [66; 62], e dife-

[9] Dispositivos equipados com diversas tecnologias sem fio, e.g. GPRS, Bluetooth, e WiFi, e com capacidade de decidir qual interface usar em diferentes circunstâncias.

rentes padrões de movimento [78], dentre outros. A especialização dessas soluções de gerenciamento de *handoff* para cenários de uso particulares dificulta, ou mesmo inviabiliza, sua aplicação direta em outros cenários.

A gerência de *handoff*, entretanto, é um problema mais amplo, envolvendo diversas camadas de software [63]. A transferência de conexão física entre pontos de acesso é apenas o inicio de uma série de procedimentos que visam manter virtualmente constante o uso de algum serviço. Aplicações, ou camadas intermediárias, em algum momento precisam lidar com os eventos de estabelecimento e/ou perda de conexões com pontos de acesso. Nesse contexto, existem diversas propostas para executar *handoffs* no contexto da troca de dados entre pares numa rede. Estas propostas operam em diferentes camadas da pilha de protocolos de rede. A maioria delas são baseadas em modificações do Mobile IP [48] e, portanto, implementadas na camada de rede [40; 47]. Outras propostas para gerência de *handoffs* no nível de troca de dados incluem estratégias baseadas em SIP [10] [54] (operando na camada de aplicação) [4; 62] ou em SCTP [11] [67] (operando na camada de transporte) [36].

2.4 Aplicações de Tempo-Real

Sistemas computacionais de tempo-real [38] são sistemas cujas tarefas dependem não apenas do resultado lógico da computação, mas também do cumprimento de requisitos temporais para a obtenção deste resultado. Ou seja, uma resposta correta mas que desrespeita algum requisito temporal associado à aplicação é considerada uma resposta não correta. O não cumprimento de algum desses requisitos temporais pode causar desde inofensivas pequenas degradações de qualidade, como ruídos numa ligação telefônica, a conseqüências de maior magnitude, como acidentes fatais devido à falhas no sistema ABS [12] de um automóvel. O requisito básico de um sistema de tempo-real é prover maneiras de assegurar que tarefas sejam executadas respeitando suas restrições temporais.

Sistemas de tempo-real dividem-se entre *hard real-time* e *soft real-time*. Nos dois casos existem restrições temporais quanto à execução de alguma tarefa, por exemplo, momento futuro quando uma resposta é aguardada e limites de tolerância máximo e mínimo para a

[10] *Session Initiation Protocol.*

[11] *Stream Control Transmission Protocol.*

[12] *Antilock Braking System.*

chegada dessa resposta. Em sistemas *hard real-time*, os prazos devem ser cumpridos, sob pena de um resultado inaceitável ocorrer. Em sistemas *soft real-time*, normalmente os prazos devem ser satisfeitos, mas se um certo número de prazos for perdido o sistema ainda pode ser considerado operacionalmente aceitável.

Neste trabalho o foco está sobre aplicações *soft real-time* cujos requisitos temporais estejam sobre a transmissão e/ou recepção de dados via rede, mais especificamente, pela interface Bluetooth de dispositivos móveis. Provavelmente a classe de aplicações que melhor se alinha com o trabalho aqui apresentado é a classe de aplicações baseadas em *streaming*, descritas a seguir.

2.4.1 Aplicações de *Streaming*

Aplicações de *streaming* são aquelas onde conteúdo, em geral multimídia (áudio e/ou vídeo), é transferido e reproduzido em tempo-real entre provedores e consumidores numa rede [34]. Diferente do paradigma "tradicional", onde arquivos precisam ser baixados completamente para os computadores dos clientes para então serem reproduzidos, conteúdo multimídia distribuído via *streaming* é reproduzido ao mesmo tempo em que é baixado.

Aplicações multimídia que lidam com a reprodução de áudio e vídeo, por exemplo, são um tipo de aplicação de tempo-real [38]. Como tal, possuem algumas propriedades temporais que precisam ser respeitadas durante sua execução para que seu papel seja cumprido de forma adequada. Em aplicações multimídia como reprodutores de áudio e/ou vídeo, a restrição está em manter constante, dentro do *software*, o fluxo entre o módulo responsável por ler o arquivo com os dados, módulos intermediários que decodificam através de CODECs [13] a informação lida do arquivo, e o módulo final, responsável por desenhar algo na tela ou tocar algum som nos auto-falantes, por exemplo. Manter o fluxo constante significa controlar a quantidade de informação lida/decodificada por intervalo de tempo. Os dados do arquivo precisam ser lidos e decodificados na mesma velocidade em que são reproduzidos. Informação pode ser perdida caso a leitura/decodificação aconteça mais rápido que o tempo necessário para reproduzi-la (*buffer overflow*). Perda de informação, nesse contexto, torna-se

[13] *COders/DECoders*. *Softwares* que implementam diferentes algoritmos para comprimir conteúdo multimídia. CODECs podem implementar algoritmos *loss-less*, onde nenhum dado é perdido durante a compressão, ou *lossy*, onde dados são perdidos durante a compressão.

perceptível para o usuário final na forma de ruídos no áudio sendo tocado ou quadros incompletos e/ou congelamentos no vídeo. De forma análoga, paradas temporárias na reprodução do áudio/vídeo são esperadas caso a taxa de leitura/decodificação seja mais lenta que a taxa de reprodução (*buffer underflow*).

Conceber uma aplicação de *streaming* significa, essencialmente, quebrar a única aplicação que existia anteriormente para a reprodução de arquivos multimídia locais em duas aplicações: uma servidora e outra cliente. Além disso, novos módulos precisam ser desenvolvidos e incorporados às cadeias de módulos das duas aplicações. Estes novos módulos devem prover as funcionalidades de empacotar dados do áudio/vídeo lido para envio via rede e, na outra ponta, ler esses dados a partir da rede e desempacotá-los, para então serem entregues ao módulo responsável pela sua reprodução.

Para o empacotamento dos dados para envio via rede, é necessário dividir a informação a ser transferida entre diversos pacotes do protocolo de transporte escolhido. Considerado apenas o uso de redes TCP/IP, as opções atuais de protocolo de transporte são TCP e UDP. Ambos os protocolos possuem virtudes e problemas que devem ser levados em conta no momento de sua escolha. A característica comum aos dois é que nenhum deles foi projetado para trabalhar com aplicações de *streaming* [14].

TCP implementa transmissão confiável de dados, garantindo a entrega dos dados na sua ordem correta. Para cada pacote TCP enviado, o remetente aguarda confirmação de recebimento. Eventuais pacotes perdidos (não confirmados dentro do período estipulado) são re-enviados antes que o próximo pacote da fila de envio seja enviado. Em cada tentativa frustrada de enviar algum pacote, o TCP aguarda um tempo aleatório crescente antes de fazer a próxima tentativa [46]. UDP, por outro lado, implementa entrega não confiável de dados, o que traz pouca sobrecarga à transmissão de dados. Entretanto, não existem garantias nem de que os dados enviados serão, em algum momento, recebidos do outro lado, nem que esses dados vão chegar na mesma ordem em que foram enviados. Em aplicações de *streaming*, a eventual perda de dados é preferível à completa parada da renderização dos dados devido atraso no envio de grandes quantidades de informação. Ou seja, se por um

[14] Novos protocolos de transporte para a pilha TCP/IP, particularmente DCCP (*Datagram Congestion Control Protocol*) e SCTP (*Stream Control Transmission Protocol*), vêm sendo desenvolvidos para atender a demanda crescente por esse tipo de aplicação.

lado a eventual perda de pacotes UDP pode provocar ruídos ou quadros incompletos, por outro lado o atraso de pacotes TCP implica não apenas o atraso de pacotes isolados, mas o atraso no envio de todos os dados que seguem aquele pacote. Devido às características dos dois protocolos, mais as características de aplicações de *streaming*, o UDP freqüentemente é escolhido, dentro da pilha TCP/IP, como o protocolo de transporte da aplicação.

Em resumo, numa aplicação de *streaming* de conteúdo multimídia, as mesmas restrições temporais das aplicações multimídia convencionais continuam valendo, mas com o acréscimo de se lidar com a transmissão de dados via redes de melhor esforço [15], como UDP/IP. No contexto de aplicações multimídia as principais conseqüências de se lidar com esse tipo de rede são: oscilações de *jitter*, eventuais perda de dados, dados entregues fora de ordem, replicação de dados, e, para o caso de aplicações com requisitos temporais ainda maiores (e.g. telefonia), atraso. Justamente para atacar alguns dos problemas relacionados à transmissão de *streams* multimídia via redes de melhor esforço foi definida a suíte de protocolos RTP (RTP, RTCP, e RTSP).

O RTP [60], *Real-Time Transport Protocol*, é voltado para o transporte de dados com características de tempo-real, como os de aplicações multimídia. No RTP, cada trilha do conteúdo a ser enviado via *streaming* é transmitida numa sessão RTP distinta. Trilhas são diferentes tipos de mídia (áudio, vídeo, legendas, etc) que precisam ser sincronizados no momento da renderização. Normalmente todas essas trilhas são codificadas juntas num mesmo arquivo, mas também é possível que elas sejam obtidas a partir de origens distintas, como arquivos diferentes ou câmeras e microfones ao vivo. As facilidades que o RTP acrescenta ao *streaming* de conteúdo multimídia são: identificação do conteúdo de cada pacote (e.g. CODEC utilizado), enumeração de pacotes para eventual re-ordenação de dados e eliminação de duplicatas, *timestamps* pra sincronização de trilhas, e monitoração do envio de dados.

O funcionamento ideal de aplicações de *streaming* baseadas em RTP envolve, na verdade, dois protocolos intrinsecamente relacionados: o RTP em si, responsável pelo transporte dos dados; e o RTCP (*RTP Control Protocol*) [60], responsável por monitorar a presença de participantes e parâmetros de qualidade de sessões RTP. Para cada sessão RTP da qual participa,

[15] Redes baseadas em datagramas e em serviços sem conexão onde não existem garantias quanto à entrega de dados.

cada participante envia periodicamente para os demais participantes relatórios RTCP sobre as atuais condições de sua recepção do *stream*. Assim, cada participante de uma sessão RTP têm informações suficientes para inferir métricas de qualidade sobre o tráfego de *streams* entre o servidor e qualquer participante da sessão. Em geral, o maior interessado nessas informações é o próprio servidor, que pode usá-las para decidir pela redução da qualidade do *stream* em caso congestionamento na rede (revelado pela taxa de perdas relatadas pelos membros da sessão).

O último protocolo da suíte, RTSP [61] (*Real-Time Streaming Protocol*), pode ser usado pelos clientes de sessões RTP para controlar o envio de dados. Para isso, o RTSP oferece funcionalidade semelhante à de um controle remoto, onde o cliente pode comandar o início de uma transmissão, pará-la temporariamente e reiniciar em seguida. Freqüentemente, outros protocolos são usados em conjunto com a suíte RTP para a implantação de serviços de *streaming* mais sofisticados. Por exemplo, o SDP [23] (*Session Description Protocol*) é usado para descrever sessões RTP antes que o cliente inicie o *streaming*, e.g. descrevendo CODECs usados, taxas de transmissão das diferentes trilhas, endereços e portas dos servidores, etc. O SIP [54] também encontra seu espaço nesta combinação de tecnologias. Projetado para o estabelecimento, modificação, e finalização de sessões multimídia via Internet, o SIP pode ser usado em substituição ao RTSP. Entretanto, o uso mais comum do SIP é em combinação com os protocolos da suíte RTP, para o convite de novos usuários para sessões RTP previamente existentes e para gerência de mobilidade de aplicações e usuários.

2.5 Bluetooth

Bluetooth, também referenciado como padrão 802.15.1 do IEEE [2], é um padrão de tecnologia de rede sem fio desenvolvida para servir como alternativa aos cabos usados para troca de dados entre dispositivos próximos [7]. Suas características chave são sua robustez para a transmissão de dados em canais sem fio, pouca complexidade, baixo custo e baixo consumo de energia.

O Bluetooth opera na faixa de freqüências não licenciadas ISM [16] que vai de 2.400 MHz a 2.483,5 MHz na maioria dos países. Para evitar interferências com dispositivos próximos

[16] *Industrial, Scientific, and Medical.*

trabalhando na mesma faixa de freqüências ISM, um esquema de FHSS [17] é usado com freqüências variando de $2.402 + k$ MHz, onde $0 \leq k \leq 78$. A taxa normal de troca de freqüências são 1600 trocas por segundo. O período de tempo entre trocas consecutivas, onde a interface passa trabalhando numa única freqüência, é chamado de *slot*.

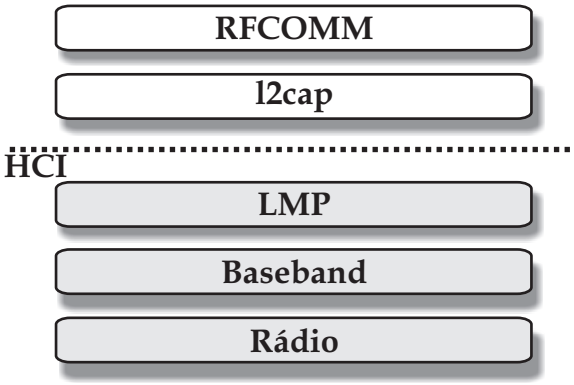


Figura 2.3: Pilha padrão Bluetooth.

O Bluetooth possui uma estrutura padronizada de protocolos, ilustrada na Figura 2.3. Na figura, os módulos abaixo da interface HCI [18] são implementados dentro da própria interface. Os demais módulos são implementados via *software*. Além das camadas que compõem a estrutura padrão de protocolos do Bluetooth, também são definidos uma série de perfis para a tecnologia. Cada perfil define como um tipo especifico de aplicação pode ser implementada, descrevendo quais componentes padrão da estrutura de protocolos devem ser usados, e como.

A camada *baseband* controla o rádio da interface. O esquema usado para a troca constante de freqüências é definido por essa camada, que também toma parte em mecanismos de baixo nível para encriptação de dados e segurança de conexões. A seleção de pacotes da camada física também é feita pela camada *baseband*. Existem dois tipos possíveis de conexões entre dois dispositivos Bluetooth: SCO e ACL. Conexões SCO (*Synchronous Connection Oriented*) são essencialmente usadas para tráfego de voz. Conexões ACL (*Asynchronous Connection Less*) são usadas para troca de dados quaisquer entre aplicações.

A camada *baseband* é a responsável pelo estabelecimento e manutenção de conexões físicas entre os dispositivos. Para tal, ela é a responsável por operações como *inquiry*, para localizar dispositivos na vizinhança, *paging*, para sincronização/conexão entre dispositivos,

[17] *Frequency Hopping Spread Spectrum.*
[18] *Host Controller Interface.*

e seleção de pacotes de camada física de acordo com as condições do canal sem fio. Existem definidos diversos tipos de pacotes de camada física, onde, para cada tipo, variam parâmetros como carga útil suportada, tamanho dos cabeçalhos, e carga extra de dados para correção de erros. Além dos canais físicos de comunicação que são iniciados e mantidos pela camada *baseband* quando usuários solicitam conexão com outros dispositivos, a camada *baseband* gerencia também por padrão cinco outros canais padronizados. Esses canais são usados para, por exemplo, busca de dispositivos próximos e estabelecimento de conexões.

A organização básica para a conexão de dispositivos Bluetooth são as *piconets*. Uma *piconet* é um grupo de dispositivos que compartilham um mesmo canal físico. Em Bluetooth um canal físico é caracterizado pelas freqüências, e o padrão de troca delas, utilizadas para comunicação entre dispositivos. Cada *piconet* com m dispositivos possui necessariamente um dispositivo mestre e $m-1$ escravos. *Piconets* podem ter até sete escravos conectados ao mesmo mestre ao mesmo tempo. Dispositivos numa *piconet* podem comunicar-se entre si através de conexões ACL ou SCO, intermediadas pelo mestre. O acesso ao canal físico por cada escravo é coordenado pelo mestre, que de tempos em tempos libera o uso do canal para cada escravo. Na estrutura de protocolos do Bluetooth, a camada LMP [19] é a responsável pela manutenção das *piconets*.

O l2cap (*Logical Link Control and Adaptation Protocol*) é o protocolo mais baixo da pilha padronizada Bluetooth a implementar a comunicação fim a fim entre processos em diferentes dispositivos Bluetooth. As características básicas desse protocolo são a segmentação e re-montagem de dados, e ajustes de alguns parâmetros de qualidade de serviço. No primeiro caso, o l2cap automaticamente quebra cada PDU [20] em tamanhos que podem ser encaixados nas cargas úteis dos pacotes de camada física. PDUs l2cap podem ser de até 64 Kb enquanto pacotes de camada física da versão 2.0 do Bluetooth suportam, no máximo, 1023 *Bytes* [64]. No processo reverso, vários pacotes de camada física recebidos são guardados até que uma mensagem l2cap completa possa ser lida a partir deles. O protocolo l2cap permite ainda que alguns parâmetros de qualidade de serviço sejam definidos, como largura de banda de pico, atraso no envio de mensagens, e variações no atraso no envio de mensagens. O l2cap provê o serviço dentro dos requisitos solicitados desde que as atuais

[19] *Link Manager Protocol*.
[20] *Protocol Data Unit*.

condições do canal físico permitam.

Por fim, no *Host Controller Interface* são definidos todas as funções pelas quais aplicações controlam a interface Bluetooth. Esse desacoplamento entre especificação e implementação permite que diferentes interfaces sejam usadas sem que o *software* precise ser alterado.

2.5.1 Estabelecendo Conexões *Baseband*

O estabelecimento de conexões em Bluetooth acontece em duas etapas: descoberta de dispositivos; e sincronização. A etapa de descoberta de dispositivos próximos (operação de *inquiry*) é necessária quando o endereço do dispositivo ao qual se deseja conectar não é conhecido. Comumente esta operação leva em média 10,24 s.

Uma vez descoberto o endereço do dispositivo ao qual se deseja conectar, seja via *inquiry* ou via outros meios, o próximo passo é a conexão entre os dois dispositivos (operação de *paging*). Conectar dois dispositivos, no contexto de Bluetooth, significa fazer com que os dois dispositivos usem o mesmo padrão para a troca de freqüências, ou seja, que tenham sincronia na troca de freqüências. Numa *piconet*, os escravos devem ajustar seus relógios internos para "bater" no mesmo ritmo que o relógio do mestre, além de usarem o mesmo padrão de troca de freqüências usado pelo mestre. Os tempos de conexão podem variar conforme os relógios internos dos dispositivos estejam mais ou menos fora de sincronia e de acordo com o modo usado pelo dispositivo que responde o *paging* para escutar tentativas de conexão.

2.5.2 A Operação de *Inquiry*

O *inquiry* é a operação definida na especificação do Bluetooth para a busca de outros dispositivos Bluetooth nas proximidades. Para a operação de *inquiry*, um canal dedicado com 32 freqüências é usado para troca de pacotes de consulta e resposta. Para que dispositivos possam ser descobertos pela operação de *inquiry*, é necessário que estes entrem periodicamente no estado *inquiry scan*. Nesse estado, dispositivos suspendem a troca de dados das aplicações para escutarem por eventuais pacotes de *inquiry* no canal dedicado.

O processo de *inquiry* pode ser descrito conforme segue. Sejam $BT_{inquiry}$ o disposi-

tivo que inicia o *inquiry* e $BT_{inquiry\_scan}$ o único dispositivo que irá responder à consulta de $BT_{inquiry}$. Para poder responder à consulta de $BT_{inquiry}$, $BT_{inquiry\_scan}$ deve entrar periodicamente no estado *inquiry scan*. Por padrão, dispositivos Bluetooth entram no estado *inquiry scan* por 11,25 ms a cada 2,56 s [64]. Para o processo de *inquiry*, as 32 freqüências do canal dedicado a *inquiries* são divididas entre dois trens de 16 freqüências. O *inquiry* funciona a partir do envio, por $BT_{inquiry}$, de pacotes de *inquiry* no canal dedicado, usando um padrão de mudança de freqüências duas vezes mais rápido que o padrão normal do Bluetooth. Assim, num único *slot* são enviados pacotes de *inquiry* em duas freqüências. No *slot* seguinte, $BT_{inquiry}$ aguarda possíveis respostas dos dois pacotes que acabaram de ser enviados. Por questões de robustez da busca, os envios de pacotes de *inquiry* em cada freqüência são repetidos 256 vezes em quatro rodadas. Ao escutar um pacote de *inquiry* em um de seus chaveamentos para o estado *inquiry scan*, $BT_{inquiry\_scan}$ aguarda um tempo aleatório de até 1.023 *slots* (640 ms) para então entrar no estado *inquiry response*. Caso, enquanto no estado *inquiry response*, $BT_{inquiry\_scan}$ receba outro pacote de *inquiry*, então este será respondido para $BT_{inquiry}$, 1 *slot* depois, com um pacote com as informações necessárias por $BT_{inquiry}$ para estabelecer uma conexão com $BT_{inquiry\_scan}$.

De acordo com os procedimentos descritos pela equipe do Bluetooth SIG em [64], o tempo total de gasto pela operação de *inquiry* $\tau_{inquiry}$ pode ser deduzido como:

$$\tau_{inquiry} = ((\frac{\delta_{slot}}{2} \times 32) \times 256) \times 4 \tag{2.1}$$

onde, δ_{slot} é o tamanho do *slot* Bluetooth, 32 é o número de freqüências, 256 é o número de repetições de transmissões para cada freqüência, e 4 é o número de rodadas onde o o envio de dados é repetido em todas as freqüências. Substituindo o valor do *slot* na equação 2.1 têm-se:

$$\begin{aligned} \tau_{inquiry} &= ((625\mu s \times 16) \times 256) \times 4 && (2.2)\\ &= 10.240.000\mu s && (2.3)\\ &= 10,24s && (2.4) \end{aligned}$$

O número de *slots* usados $T_{inquiry}$ pode ser encontrado a partir do tempo total gasto. Partindo de 2.3 têm-se:

$$T_{inquiry} = \tau_{inquiry}\mu s/\delta_{slot} \quad (2.5)$$

$$T_{inquiry} = 10.240.000\mu s/\delta_{slot} \quad (2.6)$$

$$= 10.240.000\mu s/0,625\mu s \quad (2.7)$$

$$= 16.384 \quad (2.8)$$

O custo de se responder *inquiries*, em termos de uso de *slots*, é dividido entre o tempo gasto escutando por mensagens de *inquiry* e respondendo essas mensagens. No seu modo de operação normal, interfaces Bluetooth periodicamente suspendem o tráfego de dados para escutar eventuais mensagens de *inquiry* num canal dedicado a cada 2,56 s. Em cada parada para escutar nesse canal são usado 18 slots (11,25 ms). De acordo com a especificação do Bluetooth, caso alguma mensagem de *inquiry* seja captada, a interface aguarda um tempo aleatório de até 1.023 *slots* (640 ms). Ao final desse intervalo, a interface enviará a resposta do *inquiry* um *slot* após receber mais uma mensagem de *inquiry*. Considerando que $BT_{inquiry\_scan}$ opera em modo normal de escuta por mensagens de *inquiry*, o número de *slots* T_{inq_resp} usados para responder o *inquiry* pode ser estimado como:

$$T_{inq\_resp} = N_{backoff} + 2 \quad , \quad 0 \leq N_{backoff} \leq 1.023 \quad (2.9)$$

onde $N_{backoff}$ é o número aleatório de *slots* que a interface irá aguardar antes de tentar responder uma mensagem de *inquiry* recebida, mais os dois *slots* necessários para receber novo pacote de *inquiry* e enviar a resposta em seguida. Assim, o custo de se responder *inquiries* oscila entre 2 e 1.025 *slots*. O tempo gasto respondendo *inquiries* τ_{inq_resp} pode ser deduzido a partir do número de *slots* usados conforme equação 2.10. Sendo T_{inq_resp} uniformemente distribuído entre 2 e 1.025, têm-se que $1,25ms \leq \tau_{inq\_resp} \leq 640ms$.

$$\tau_{inq\_resp} = T_{inq\_resp} \times \delta_{slot} \quad , \quad 2 \leq T_{inq\_resp} \leq 1.025 \quad (2.10)$$

2.5.3 A Operação de *Paging*

O *paging* é o processo pelo uma conexão física é estabelecida entre dois dispositivos Bluetooth. Durante a operação de *paging* é feita a sincronização dos relógios dos dois disposi-

tivos, assim como eles passam a trocar as freqüências do canal entre si usando o mesmo padrão pseudo-aleatório. O procedimento de *paging* acontece de maneira similar ao procedimento de *inquiry*. Para estabelecer conexão com outro dispositivo, o dispositivo que inicia o *paging* deve transmitir mensagens de *page* para o dispositivo alvo. Assim como no *inquiry*, estas mensagens são enviadas usando um padrão especial para troca de freqüências, duas vezes mais rápido que o convencional. Dispositivos para serem "conectáveis", precisam também, periodicamente, escutar no canal dedicado para *pagings* por tentativas de conexão de outros dispositivos. Antes de iniciar o *paging*, as 32 freqüências do canal dedicado aos *pagings* são divididas entre dois trens de 16 freqüências. Diferente do *inquiry*, no *paging* é suficiente repetir as transmissões em cada freqüência 128 vezes.

Os chaveamentos da interface para escutar o canal de *pagings* são regulados por dois parâmetros: *page scan interval* e *page scan window*. O primeiro parâmetro diz com que periodicidade a interface irá escutar no canal de *pagings*. O segundo diz por quanto tempo a interface permanecerá ouvindo no canal de *pagings* em cada chaveamento. De acordo com a especificação, o primeiro parâmetro pode assumir quaisquer valores entre 18 e 4096 *slots*, enquanto o segundo parâmetro pode assumir valores entre 17 e 4046 *slots*, desde que seu valor seja menor ou igual ao valor do *page scan interval*.

O processo de *paging* pode ser descrito da seguinte maneira. Sejam BT_{paging} o dispositivo que inicia o *paging*, e BT_{page_scan} o dispositivo alvo da conexão. Para receber conexões de outros dispositivos, BT_{page_scan} periodicamente suspende as transmissões de dados das aplicações para escutar o canal dedicado aos *pagings*. O padrão em Bluetooth é que dispositivos entrem no estado *page scan* por 11,25 ms a cada 1,28 s. Enquanto isso, BT_{paging} inicia o processo de conexão, transmitindo pacotes de *page* no canal dedicado e escutando eventuais respostas no *slot* seguinte a cada transmissão. Quando BT_{page_scan}, em um de seus chaveamentos para o canal de *pagings*, recebe um pacote de *page*, o pacote é respondido para BT_{paging}, seguido de uma troca padronizada de pacotes para a conclusão da conexão entre os dispositivos [7].

O custo em tempo de se fazer um *paging* depende do quanto sincronizados estão os dois dispositivos. O número de *slots* necessários T_{paging} para concluir o *paging*, do lado da interface que inicia o processo, depende do número da tentativa k de envio de mensagem de *paging* numa dada freqüência e pode ser calculado segundo 2.11.

$$T^{k}_{paging} = k + 5 \quad , \quad 1 \leq k \leq 4.095 \tag{2.11}$$

Caso as interfaces estejam em perfeita sincronia, uma mensagem de *paging* enviada no primeiro *slot* do processo será respondida no *slot* seguinte e o restante do processo de *paging* acontecerá com mais 5 trocas de mensagens entre os dispositivos, cada uma em um *slot*. O pior caso acontece quando apenas a última mensagem enviada, no *slot* 4.095, é respondida. Os número de *slots* usados portanto varia entre $6 \leq T_{paging} \leq 4.100$. O custo do *paging* em tempo é função do número de *slots*, conforme 2.12. Assim, os tempos podem variar entre 3,75 ms e 2,56 s. Entretanto, empiricamente verifica-se que o tempo médio de *pagings* gira em torno de 1,28 s [13].

$$\tau_{paging} = T_{paging} \times \delta_{slot} \quad , \quad 6 \leq T_{paging} \leq 4.100 \tag{2.12}$$

Na ausência de erros no canal, o custo para BT_{page_scan} responder um pedido de *paging* é constante, somando um total de $T_{page\_scan} = 5,5$ *slots* [7]. O total de tempo τ_{page_resp} gasto com a interface transmitindo e escutando no canal de *paging* portanto é de aproximadamente 3,4 ms, conforme 2.13.

$$\tau_{page\_resp} = T_{page\_scan} \times \delta_{slot} \tag{2.13}$$

2.5.4 *Inquiry* e *Paging* X Tráfego de Dados em Tempo-Real

Como discutido anteriormente, as operações de *paging* e *inquiry*, assim como os procedimentos associados aos respectivos estados de resposta, possuem prioridade no uso da interface Bluetooth. Como resultado, aplicações que necessitem do uso contínuo da interface tendem a ser prejudicadas devido à multiplexação do canal entre os dois tráfegos.

Nesse contexto, alguns experimentos foram conduzidos como forma de ilustrar o impacto das operações de *paging* e *inquiry* sobre o tráfego de dados em tempo-real pelas interfaces Bluetooth. Para obter os resultados a seguir foram usados três computadores, BT_1, BT_2, e BT_3, cada um com uma interface Bluetooth 2.0. Nos gráficos a seguir são plotadas as

larguras de banda instantâneas de um fluxo constante de aproximadamente 400 Kbps passando pelas interfaces Bluetooth. As larguras de banda foram medidas no nível de recepções de mensagens num *socket l2cap*. As operações de Bluetooth avaliadas foram: iniciar um *inquiry*, responder um *inquiry*, iniciar um *paging*, e responder um *paging*.

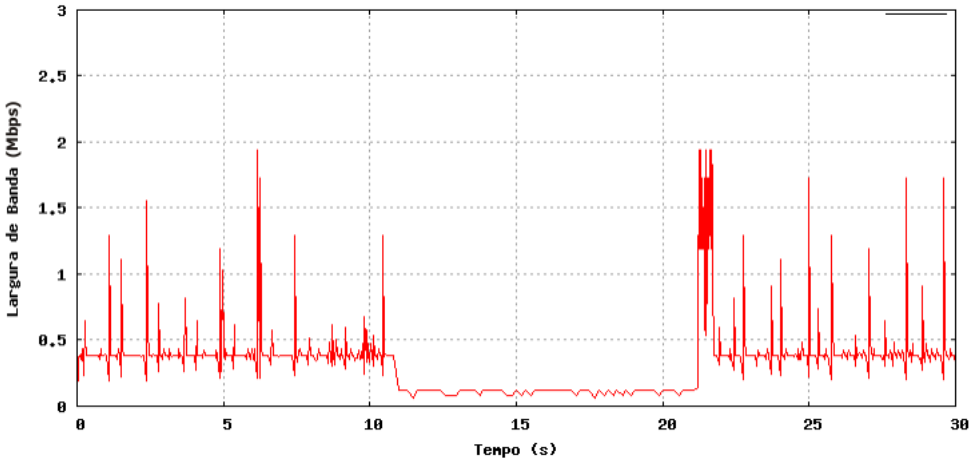


Figura 2.4: Impacto do *inquiry* num fluxo de 400 Kbps.

Um exemplo do impacto da operação de *inquiry* sobre um fluxo constante é mostrado na Figura 2.4. Neste exemplo, BT_1 transmite um fluxo para BT_2 e inicia, aproximadamente no momento 10 s, um *inquiry* com a duração padrão de aproximadamente 10 s. Para garantir a descoberta de todos os dispositivos em sua vizinhança, o *inquiry* em BT_1 consumirá 16.384 *slots* para enviar mensagens de *inquiry* e escutar eventuais respostas, totalizando 10,24 s com a interface ocupada com o *inquiry*. Durante esse tempo, longo para uma aplicação com características de tempo-real, a disponibilidade da interface para a aplicação é reduzida devido à prioridade da operação de *inquiry*.

Na Figura 2.5 é ilustrado o impacto da operação de *paging* sobre o mesmo fluxo. Assim como anteriormente, BT_1 transmite um fluxo constante para BT_2. Com o fluxo em curso, aproximadamente no momento 10 s, BT_1 faz um *paging* para BT_3 e as oscilações na qualidade da transmissão dos dados são medidas sobre dados que chegam em BT_2. Diferente do *inquiry*, onde o custo da operação é constante [21], no *paging* o tamanho da ocupação do canal varia conforme os dois dispositivos estejam fora de sincronia, dependendo também das configurações usadas no dispositivo alvo para responder *pagings*. Especificamente no teste

[21] O número de *slots* necessários pela interface que inicia o *inquiry* é constante considerando que o tempo padrão de busca seja usado como condição de parada.

retratado na Figura 2.5, é perceptível uma queda no fluxo normal dos dados da aplicação pouco depois que a operação de *paging* é iniciada.

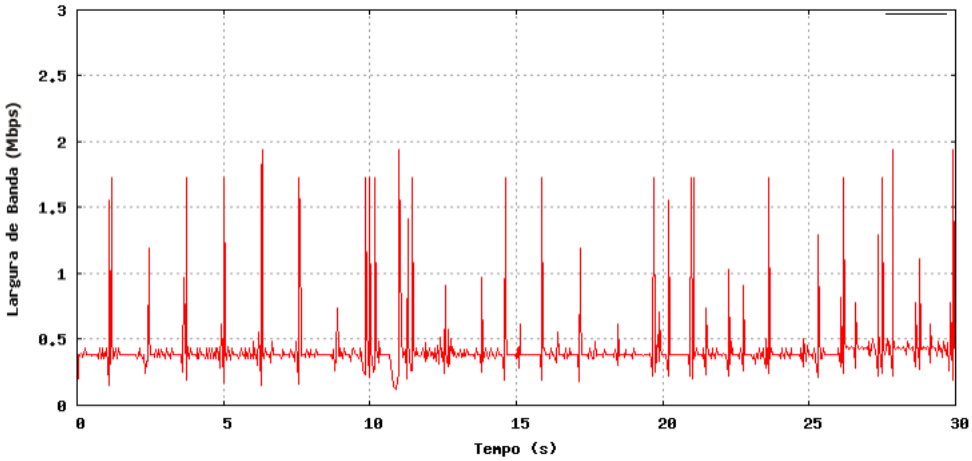


Figura 2.5: Impacto do *paging* num fluxo de 400Kbps.

Conforme visto na seção anterior, o número de *slots* necessários para responder *inquiries* e *paging* é muito menor que o número de *slots* necessários para disparar essas operações. Nas Figuras 2.6 e 2.7 são mostrados os gráficos com os dados do desempenho do mesmo fluxo mediante ocupação da interface Bluetooth com respostas para um *inquiry* e um *paging*, respectivamente. Para o teste de resposta de *inquiry*, BT_1 transmite um fluxo para BT_2. No momento 10 s, BT_3 inicia um *inquiry*, mas somente BT_1 está configurado para responder *inquiries*. No gráfico plotado na Figura 2.6 é ilustrado o desempenho do fluxo conforme verificado em BT_2. Por fim, para o teste de resposta de *paging*, BT_1 transmite o mesmo fluxo constante para BT_2. No momento 10 s, BT_3 inicia um *paging* para BT_1 e o desempenho do fluxo, conforme verificado em BT_2 é ilustrado na Figura 2.7. Nesses dois últimos testes, não houve mudança perceptível no desempenho dos fluxos.

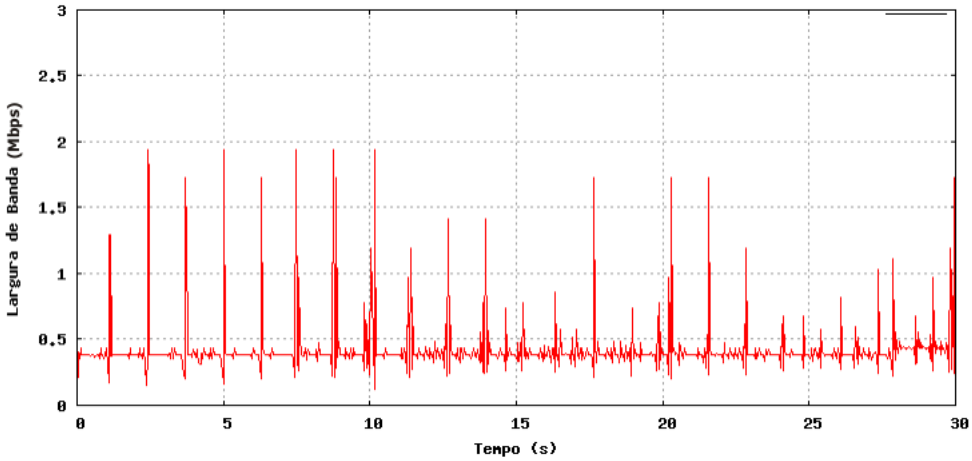


Figura 2.6: Impacto do *inquiry scan* num fluxo de 400Kbps.

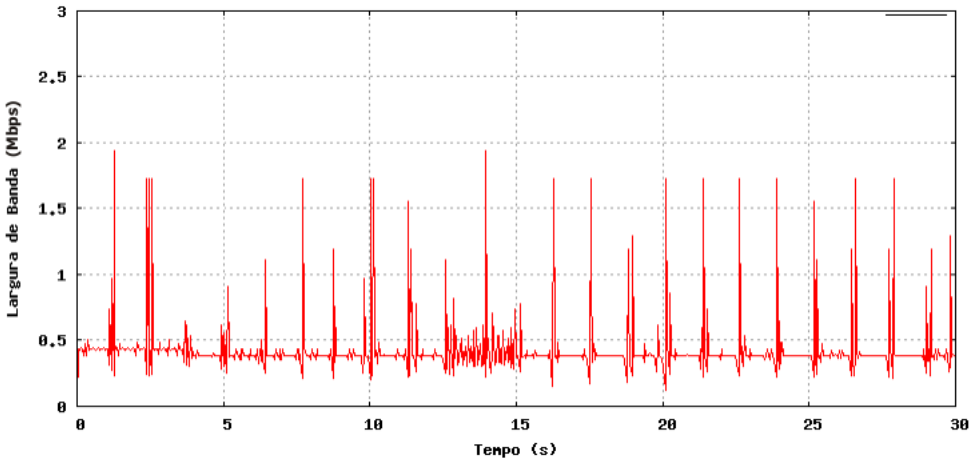


Figura 2.7: Impacto do *page scan* num fluxo de 400Kbps.

Capítulo 3

Gerenciando *Handoffs* para Bluetooth no Contexto de Aplicações de Tempo-Real

Neste capítulo é apresentada a proposta deste trabalho para atender os objetivos levantados na Seção 1.2. O ponto chave para a definição da solução foi definir um protocolo para gerência de *handoff* que minimizasse o impacto das operações de Bluetooth, necessárias para o *handoff*, sobre o tráfego de dados em tempo-real.

Na solução proposta são definidas duas entidades principais: estações base (EBs), e dispositivos móveis (DMs). Seus papéis e organizações físicas e lógicas são descritas a seguir.

3.1 Visão Geral

Por ser uma tecnologia desenvolvida para substituir cabos, Bluetooth não oferece nenhuma facilidade para o gerenciamento de *handoff* [9]. Dessa forma, eventuais protocolos para gerenciamento de *handoff* precisam ser implementados acima da pilha padronizada, usando funções previamente definidas na interface HCI, ou em perfis já existentes, de forma a tornar transparente a mudança de ponto de acesso. Uma vez feita a troca de ponto de acesso, cabe às aplicações (ou camadas intermediárias) implementar os mecanismos para redirecionamento de tráfego e eventuais tratamentos sobre os dados enviados/recebidos (como dados fora de tempo, replicados, ou perdidos).

No contexto deste trabalho definimos uma infra-estrutura, ilustrada na Figura 3.1, onde são representadas as entidades físicas que compõem a solução final. As EBs são os pontos

de acesso do sistema. Cada EB possui interfaces para ambas as redes cabeada e Bluetooth. EBs comunicam-se com EBs vizinhas via rede cabeada. Ao posicionar EBs próximas umas às outras, de forma a sobrepor suas áreas de cobertura, aumenta-se o espaço físico contínuo coberto pelo sistema. À rede de todas as EBs que, juntas, cobrem o mesmo espaço físico contínuo denomina-se domínio de mobilidade. EBs são ditas vizinhas quando existe uma intersecção entre suas áreas de cobertura.

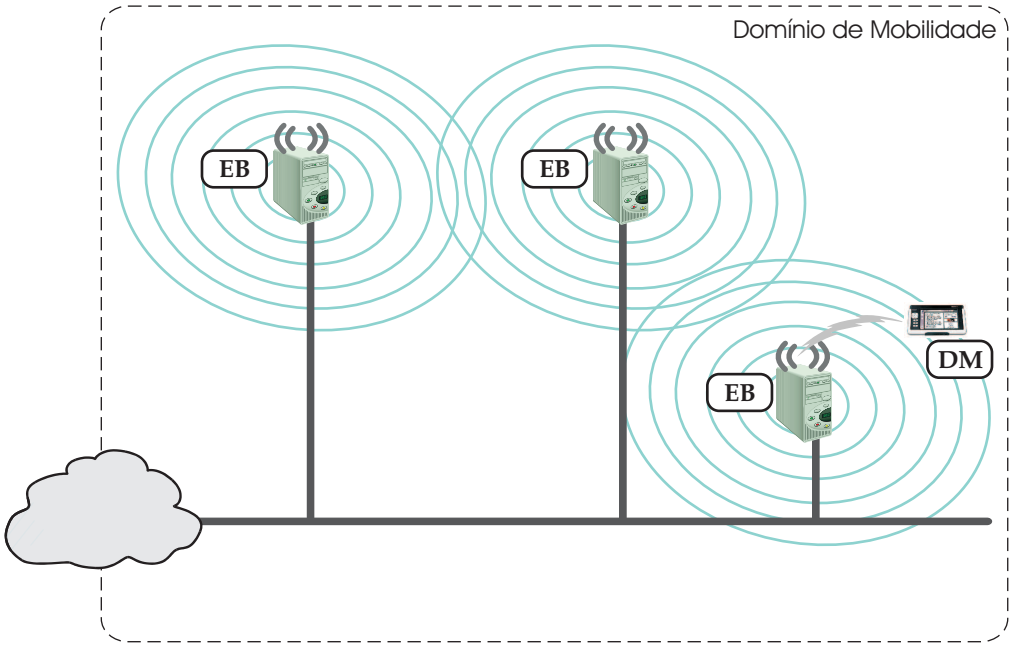


Figura 3.1: Visão geral do sistema.

Na solução introduzida nesse trabalho tira-se proveito da característica de múltiplas conexões simultâneas de Bluetooth para definir um algoritmo de gerência de *soft handoffs*, reduzindo a probabilidade de perda de dados e de grandes oscilações de *jitter*. O processo de *handoff* é disparado e gerenciado pelas EBs quando DMs, devido à movimentação dos usuários, ultrapassam uma determinada distância limiar entre si próprios e as EBs às quais estão atualmente conectados. De forma inversa, um processo de *handoff* bem sucedido termina quando DMs voltam a aproximar-se das EBs.

3.1.1 Estados de um DM

DMs podem assumir diferentes estados enquanto conectados às EBs do domínio de mobilidade. A principal função dessa representação de estados é servir como base para a decisão

das EBs quanto ao disparo do processo de *handoff* para algum DM em movimento, ou por sua saída de algum processo de *handoff* já ativo. Na Figura 3.2 são representados os estados e transições de estados associados a DMs.

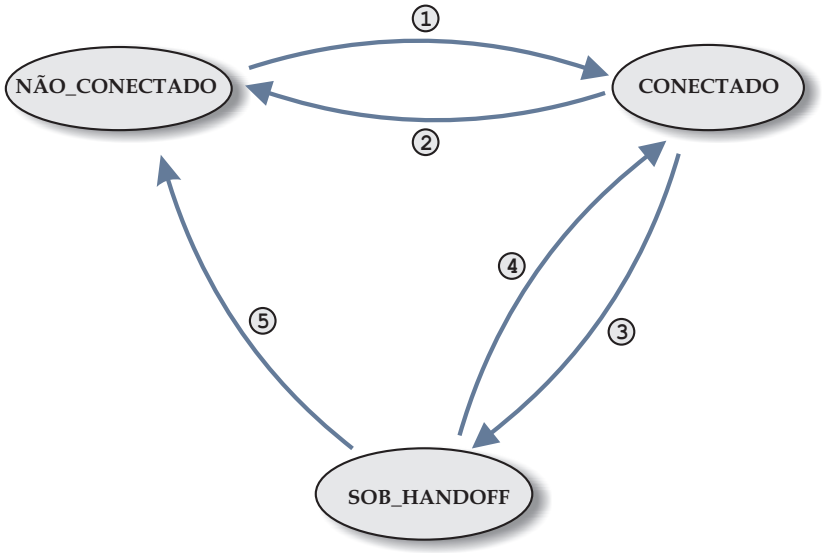


Figura 3.2: Estados e transições de estados associados a DMs.

Um dispositivo que não está conectado a nenhuma EB é um dispositivo externo ao sistema e, portanto, encontra-se no estado NÃO_CONECTADO. Após conseguir conexão com alguma EB, o DM passa a assumir o estado CONECTADO (1). Do estado CONECTADO, DMs podem desconectar-se do sistema (retornando ao estado NÃO_CONECTADO), eventualmente sem a necessidade de fazer *handoffs* (2), entretanto, o esperado é que DMs movimentem-se após o estabelecimento de alguma conexão. Ao ultrapassarem o ponto a partir do qual o processo de *handoff* é disparado, DMs têm seu estado atualizado para SOB_HANDOFF (3), onde podem estar conectados a mais de uma EB. DMs retornam ao estado CONECTADO quando continuam se movendo e perdem contato com uma das EBs às quais estavam conectados (4). Devido a processos de *handoff* mau sucedidos, ou ainda à movimentação para regiões não cobertas por nenhuma EB, DMs perderão contato total com as EBs do sistema, voltando ao estado NÃO_CONECTADO (5).

3.1.2 O Registro de um DM

Para manter controle sobre a movimentação de DMs, as EBs constantemente trocam entre si informações sobre DMs monitorados por elas. Estas informações são usadas por cada

EB para gerenciar registros numa base de dados local sobre os DMs ligados diretamente a elas ou em sua vizinhança, ou seja, monitorados por EBs vizinhas (Figura 3.3). Para dar suporte à decisão de participação ou não em *handoffs*, três informações sobre os DMs são trocadas entre as EBs: identificador do DM; seu estado no sistema (segundo a EB que envia a notificação); e EBs em contato com o DM.

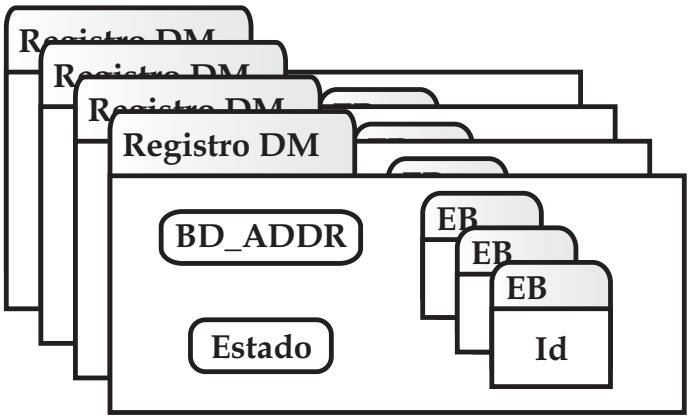


Figura 3.3: Registros de DMs e suas informações.

O identificador de um DM é seu endereço BD_ADDR. Este endereço, semelhante ao endereço MAC em redes *ethernet*, possui 48 *bits* divididos em 6 octetos que identificam de forma única diferentes interfaces Bluetooth. O endereço DB_ADDR de um DM torna-se conhecido no sistema quando este conecta-se pela primeira vez em uma EB. O endereço BD_ADDR é usado para agilizar a conexão com novas EBs durante o *handoff*, sem a necessidade de demorados *inquiries* (vide Seção 3.2.2). Como um DM pode estar conectado a mais de uma EB por vez (durante o *handoff*, excepcionalmente), uma lista de identificadores de EBs é mantida junto com o registro do DM. Um identificador de EB acumula as propriedades de unicamente identificar EBs na rede cabeada e de servir como base para a comunicação entre elas (ou seja, seu endereço IP). Por fim, no registro é mantido o estado atualizado do DM no sistema, conforme os estados descritos na Seção 3.1.1. O estado NÃO_CONECTADO possui tratamento especial na atualização do estado do DM. Conforme descrito na Seção 3.3, a notificação do estado NÃO_CONECTADO serve para atualizar a lista de EBs às quais o DM está conectado.

3.2 Gerência de *Handoff* - Transições entre EBs

Toda a coordenação envolvida no processo de monitoração de DMs e gerenciamento de *handoffs* ocorre nas EBs, poupando assim os DMs de quaisquer custos envolvidos nestes processos. Nesse contexto, custos podem ser mapeados para recursos computacionais, e.g. processamento e memória para executar processos extras, e temporais, e.g. atrasos envolvidos na sinalização entre DM e EB durante o *handoff*.

Para gerenciar *handoffs*, EBs precisam manter controle sobre os DMs em sua vizinhança, onde manter controle significa manter atualizados registros com as identificações de DMs e monitorar suas mudanças de estado e lista de EBs às quais estão conectados. Uma vez conectados a alguma EB do domínio de mobilidade, DMs não precisam seguir nenhum procedimento para manterem-se conectados às EBs do sistema, mesmo durante *handoffs*. Assim, apenas são definidas entidades lógicas do lado das EBs, conforme ilustrado na Figura 3.4.

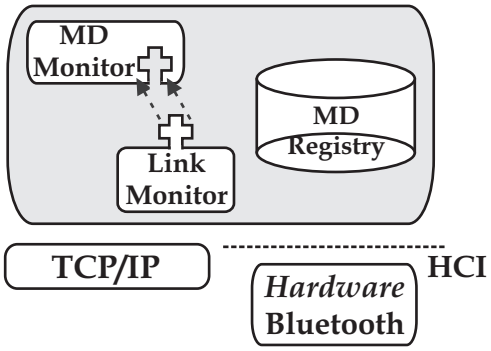


Figura 3.4: Principais entidades lógicas das EBs.

No *MD Registry* são mantidos registros, conforme descrito na Seção 3.1.2, sobre DMs ligados à propria EB ou em EBs próximas. O *Link Monitor* constantemente verifica a conexão entre DM e EB, reportando o estado atual para um *MD Monitor* associado ao DM observado. O funcionamento dessas entidades é descrito nas seções seguintes.

3.2.1 Monitorando DMs

Cada EB é responsável por monitorar mudanças de estados em DMs em contato direto com elas e propagar essas informações para EBs vizinhas. Uma vez conectado a uma EB, o DM passa a ter sua localização constantemente monitorada. São dois os objetivos da moni-

toração de DMs: perceber quando o DM cruza o ponto limiar para disparo/cancelamento de processos de *handoff*; e detectar a perda de conexão entre EB e DM.

Idealmente, a monitoração de DMs envolveria informações precisas sobre suas direções e velocidades. Entretanto, devido limitações dos mecanismos atualmente disponíveis para localização em Bluetooth, o máximo que se pode buscar, sem comprometer recursos das interfaces, são estimativas da distância em linha reta entre dois dispositivos [6], conforme ilustrado na Figura 3.5.

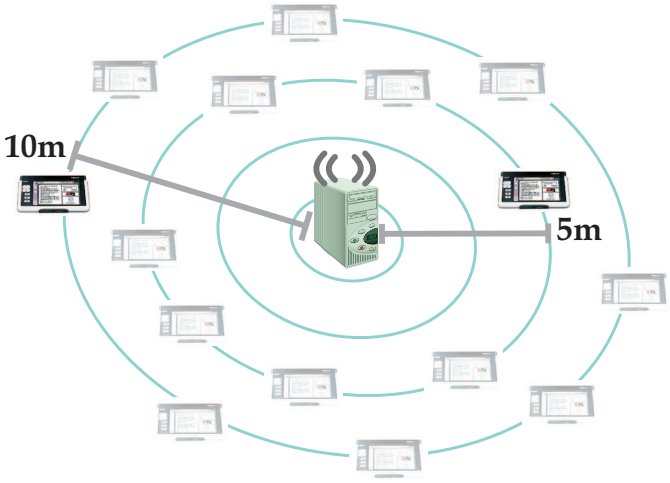


Figura 3.5: Localização de dispositivos Bluetooth.

Existem diferentes meios para se estimar a distância entre dois dispositivos Bluetooth, como RSSI [1] ou qualidade do canal, conforme discutido por Timothy Bielawa em [6], e por Javier García Castaño em [9]. A detecção de perda de conexão também é dependente de implementação, freqüentemente sendo usadas heurísticas baseadas em ajustes do *link supervision timeout* [70; **?**]. Entretanto, a pesquisa em torno de localização em Bluetooth é vasta e complexa [6], existindo atualmente diversas propostas de abordagens para localização de dispositivos Bluetooth [18; 19; 17] onde, em cada solução, variam parâmetros como complexidade, exatidão, e viabilidade técnica. Dessa maneira, e devido o fato de que neste trabalho o foco é definir um procedimento de *handoff*, serão abstraídos aqui quaisquer detalhes relacionados à implementação dos mecanismos para estimativa de distância entre EBs e DMs e detecção de perda de conexão. Desta forma, define-se um contrato com um módulo externo que implementa o comportamento desejado para localização dos DMs.

[1] *Received Signal Strength Indication*

O *MD Monitor* [2], cujo funcionamento é ilustrado no Algoritmo 3.1, é responsável por decidir que ações tomar mediante mudanças de estados verificadas em DMs monitorados. A funcionalidade de detectar as mudanças de estado do DM é delegada para um módulo externo, uma implementação de *Link Monitor*, de acordo com os estados e transições descritos na Seção 3.1.1.

Algoritmo 3.1: Monitorando mudanças de estado em DMs.

```
1  SE(novoEstadoDM = NÃO_CONECTADO) ENTAO
2    SE(estadoAnteriorDM = SOB_HANDOFF) ENTAO
3        aguardaContatoDeVizinhaOuTimeout(dadosEB, VERDADEIRO)
4    SENAO
5        defineEstado(dadosEB,NAO_CONECTADO)
6        notificaEBsVizinhas(dadosEB)
7    FIM SE
8  SENAO SE(estadoAnteriorDM = CONECTADO) E (novoEstadoDM = SOB_HANDOFF)
       ENTAO
9      defineEstado(dadosEB,SOB_HANDOFF)
10     notificaEBsVizinhas(dadosEB)
11     aguardaEncaminhamentoDeMensagem(dadosEB, VERDADEIRO)
12 SENAO SE(estadoAnteriorDM = SOB_HANDOFF) E (novoEstadoDM = CONECTADO)
       ENTAO
13     defineEstado(dadosEB,CONECTADO)
14     notificaEBsVizinhas(dadosEB)
15 FIM SE
```

O algoritmo é executado segundo um modelo de *callback* a partir do Link Monitor. Ao se conectar pela primeira vez ao sistema, cada DM possui estado CONECTADO. Quando o DM, devido à movimentação do usuário, ultrapassa o raio de distância da EB a partir do qual o processo de *handoff* é disparado, o MD Monitor é notificado, o estado do DM atualizado para SOB_HANDOFF e notificado para as EBs vizinhas. A EB também se prepara para repassar a primeira notificação que receber sobre o DM que acabou de iniciar *handoff* (mais detalhes são descritos na Seção 3.3).

Caso o DM, após atravessar o ponto limiar para disparo do *handoff*, retorne para próximo de sua EB atual, então o Link Monitor reportará a mudança de estado, que será atualizada

[2] *Mobile Device Monitor*.

pelo MD Monitor e notificada para as EBs vizinhas como CONECTADO. O comportamento é o mesmo para quando um DM, concluindo um processo de *handoff*, ingressa na área de cobertura de uma nova EB e continua movendo-se em sua direção, perdendo contato com a EB anterior e mantendo conexão apenas com a EB destino.

Toda mudança de estado de um DM monitorado por uma determinada EB deve ser atualizada na base de dados local daquela EB e comunicada às suas EBs vizinhas. Isso inclui a perda de conexão com DMs. Dessa forma, mesmo quando uma EB percebe que não mais está em contato com algum DM, seu estado é atualizado localmente para NÃO_CONECTADO e propagado para as EBs vizinhas. Uma situação excepcional acontece quando DMs saem da área de cobertura da EB após assumirem estado SOB_HANDOFF. Especificamente, o procedimento de *handoff* para um dado DM em movimento pode falhar caso a mensagem de perda de contato com ele seja recebida pela EB de destino, segundo a trajetória do dispositivo, antes que esta consiga contactar o DM. Para evitar problemas de perda de contato com o DM devido à ordem de troca de mensagens entre EBs (vide Seção 3.3), quando EBs percebem a perda de contato com DM nessa situação essa mensagem não é imediatamente enviada para suas vizinhas. Ao invés disso, a EB aguarda até que chegue uma mensagem de contato de uma de suas vizinhas com o DM em questão, ou até que um determinado período de tempo transcorra. O ajuste deste *timeout* é uma configuração de implantação do sistema. Entretanto, valores entre 5 e 10 segundos são ajustes razoáveis para esse parâmetro.

No Algoritmo 3.1 são ilustrados os procedimentos tomados pelo MD Monitor mediante as mudanças de estados de interesse. A execução do algoritmo envolve apenas a decisão de que ação deve ser tomada. Como não existem laços nem recursões, o algoritmo possui complexidade O(1), executando em tempo constante. As complexidades das funções referenciadas no algoritmo são dependentes de implementação.

3.2.2 Atribuindo Responsabilidades e Configurando Operações

Quem Deve Ser o Mestre?

Ao iniciar a definição de responsabilidades e configurações das entidades do sistema, a primeira pergunta a ser respondida é quem, dentre EBs e DMs, deve ser mestre nas *piconets*. Como já discutido na Seção 2.5, Bluetooth emprega um esquema de *pooling* para controle

de acesso ao meio. Numa *piconet*, todos os escravos têm seus relógios sincronizados com o relógio do mestre, de forma a viabilizar a tarefa do mestre de arbitrar os acessos ao meio de cada dispositivo na rede. Dispositivos Bluetooth podem também fazer parte de mais de uma *piconet* ao mesmo tempo. Quando isso acontece, o dispositivo negocia com os mestres de cada *piconet* quando irá participar de cada uma, alternando períodos onde estará sincronizado com o relógio do mestre de cada *piconet* [64]. Em termos práticos, isso significa perda de tempo e, conseqüentemente, de largura de banda, para o tráfego de dados em cada *piconet*. Além disso, a especificação de Bluetooth é bastante vaga quando descreve o comportamento de dispositivos que participam de mais de uma *piconet*. Apenas é exigido que um esquema de TDM [3] seja empregado para alternar períodos de participação do dispositivo em cada *piconet*. Como veremos na Seção 4.1.2, essa flexibilidade da especificação permite que diferentes fabricantes de interfaces criem mecanismos mais ou menos eficientes para divisão de tempo entre *piconets*, o que pode viabilizar ou não diferentes tipos de aplicações no contexto do trabalho aqui apresentado.

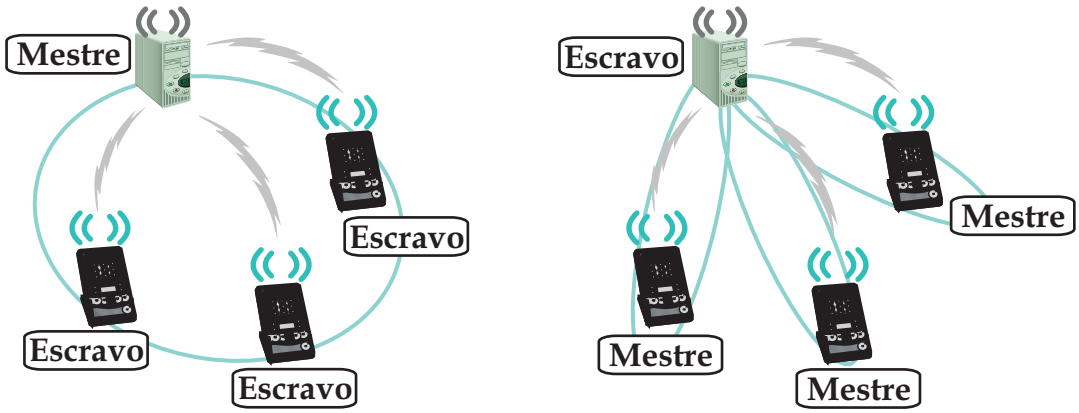


Figura 3.6: Atribuições dos papéis de mestre e escravo (adaptado do trabalho de Aman Kansal em [30]).

Conforme já discutido por Aman Kansal em [30], existem vantagens e desvantagens em definir EBs como mestres e DMs como escravos e vice versa (Figura 3.6). Tornar DMs mestres nas *piconets* com EBs significa poupar DMs dos desperdícios relacionados com o chaveamento entre diferentes *piconets*, além de, num primeiro momento, ser a solução mais escalável do ponto de vista do número de clientes atendidos pela EB, já que não existem

[3] *Time Division Multiplexing*

restrições explicitas quanto o número de conexões simultâneas que um dispositivo pode ter ativas ao mesmo tempo enquanto escravo. Por outro lado, essa arquitetura transfere para as EBs o gargalo de desempenho, já que o número de DMs conectados ao mesmo tempo numa EB tende a ser potencialmente maior que o número de EBs às quais um DM está conectado ao mesmo tempo.

Ao definir DMs como escravos nas *piconets* com EBs, restringe-se o número máximo de clientes conectados ao mesmo tempo às EBs para 7 (restrição já conhecida das *piconets*) e transfere-se para os DMs a responsabilidade de chavear seu tempo de escuta entre as EBs que estiver conectado. Como resultado, DMs precisam chavear seu tempo apenas enquanto fazem *handoff*, e entre um número reduzido de EBs. Os critérios para divisão de recursos da EB entre os DMs passam a ser apenas os critérios já conhecidos do Bluetooth, como número de participantes da *piconet*, tipo de conexão (simétrica ou assimétrica), e tipos de pacotes escolhidos pela camada física [30].

Handoffs sem *Inquiries*

De acordo com a especificação de Bluetooth, e conforme ilustrado nos experimentos da Seção 2.5.4, existem custos (de quantidade de *slots* usados) associados às operações de *inquiry*, *paging*, e os respectivos estados de resposta. Esses custos podem ser calculados baseado nas descrições destes estados na especificação (conforme as equações apresentadas nas Seções 2.5.2 e 2.5.3), apesar de que resultados de experimentos práticos podem diferir dos resultados esperados ou simulados devido a uma série de fatores, como condições do canal sem fio, qualidade das interfaces, interferência com aparelhos próximos (e.g. fornos microondas, telefones sem fio, ou pontos de acesso WiFi), etc.

Para o projeto de um protocolo de *handoff* com baixo impacto sobre o tráfego de dados em tempo-real, é necessário evitar as operações com alto consumo de *slots*, particularmente o *inquiry*, já que o *paging* é operação obrigatória para a conexão de dispositivos Bluetooth. No protocolo apresentado na Seção 3.3, o processo de *inquiry* é descartado durante os *handoffs*. Ao invés disso, EBs trocam entre si, via rede cabeada, a informação sobre o endereço BD_ADDR do DM sob *handoff* sendo, então, necessária apenas a operação de *paging*, que é iniciada pela EB. Nesse arranjo de responsabilidades a única tarefa do DM para se manter conectado ao sistema é manter-se no estado *page scan*, como forma de manter-se "conec-

tável" durante o *handoff*.

Reduzindo a Quantidade de *Slots* Necessários para os *Handoffs*

Conforme discutido no tópico anterior, uma vez eliminada a operação de *inquiry* do processo de *handoff*, o custo para o DM durante as transições entre EBs resume-se ao custo de responder o *paging* da EB. De acordo com a especificação de Bluetooth, e conforme mostrado na Equação 2.13 da Seção 2.5.3, o custo de responder um *paging* é o menor dentre os estados relacionados com as operações necessárias para o *handoff*. Esse baixo custo reflete-se numa curta obstrução, da ordem de 5,5 *slots* (não considerando eventuais erros nas trocas de pacotes entre dispositivos), no trânsito de dados pela interface Bluetooth do DM. Entretanto, o impacto total no tráfego de dados também depende da perda de desempenho da interface da EB durante o processo de *paging*, que pode variar entre 6 e 4100 *slots* (Equação 2.11), mas em média 1.28 s (aproximadamente 2048 *slots*), dependendo do desvio de sincronia entre os dispositivos.

Para obter *handoffs* ainda mais eficientes, do ponto de vista do uso de recursos das interfaces Bluetooth das EBs e DMs, a solução direta é diminuir o tempo necessário para o procedimento de *paging*. Para diminuir o número de *slots* usados no *paging* existem duas alternativas: ter relógios sincronizados nos dois dispositivos; e/ou configurar a janela e o intervalo do *paging* no DM. Como não se pode interferir nos relógios internos das interfaces, a única solução para reduzir os tempos dos *pagings* é calibrar seus parâmetros de maneira apropriada.

Os dois parâmetros do *paging*, *page scan interval* e *page scan window*, definem, respectivamente, com que freqüência e por quanto tempo uma interface escutará por tentativas de *paging* no canal dedicado para essa operação. Conforme mostrado na Figura 3.7, extraída do trabalho de Ming-Chiao Chen et al. em [11], os melhores desempenhos para a operação de *paging* são alcançados quando se aumenta o *page scan window*, e se reduz o *page scan interval*, onde mudanças no *page scan interval* parecem ter um impacto muito maior do que mudanças no *page scan window*. Ou seja, melhores resultados são alcançados quando se aumenta o tempo que a interface passa ocupada apenas escutando por pedidos de *paging*. A conseqüência direta dessa abordagem é seu impacto direto no desempenho de aplicações de tempo-real, que vão ter menos *slots* livres da interface do DM para o trânsito de seus dados.

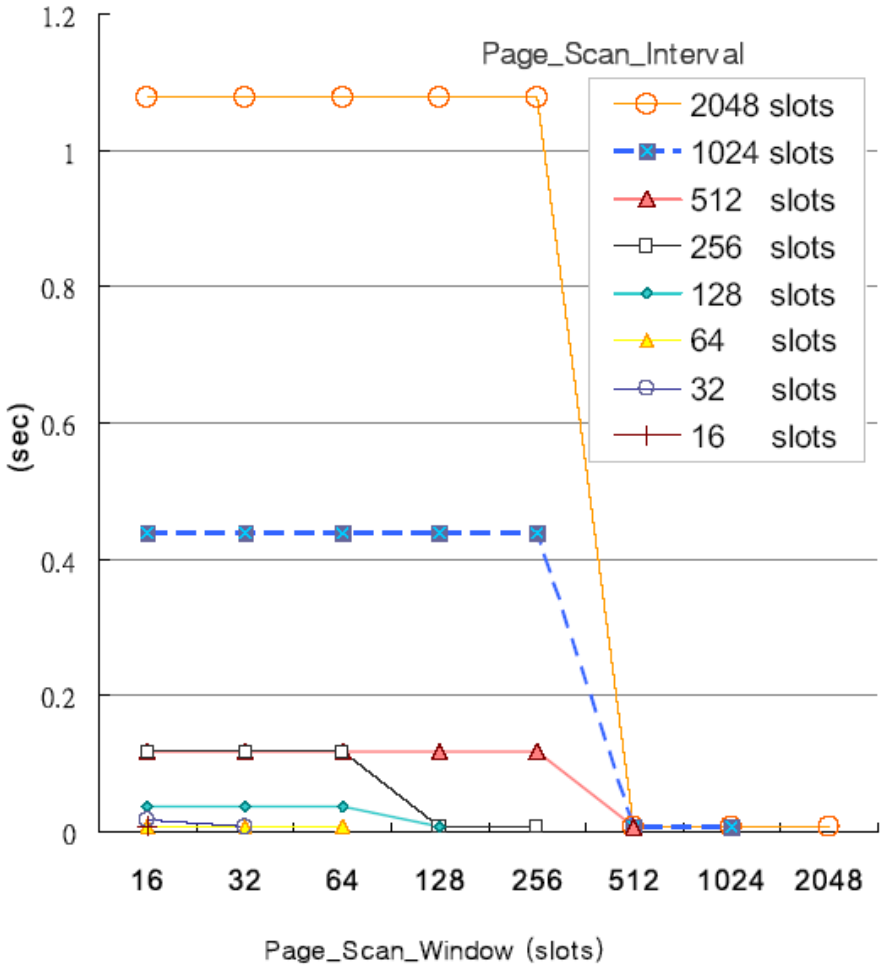


Figura 3.7: Desempenho da operação de *paging* com diferentes parâmetros (extraída do trabalho de Ming-Chiao Chen et al. em [11]).

Dessa maneira, os ajustes dos parâmetros do *paging* devem ser cuidadosamente definidos, de forma a alcançar os melhores valores sem comprometer os requisitos temporais da aplicação do usuário. A definição de valores ideais para esses parâmetros, entretanto, é uma tarefa complexa o suficiente para ser tratada num trabalho à parte. Por isso, não é proposto aqui nenhum modelo para definição de valores ótimos para os parâmetros do *paging* levando-se em conta os requisitos de cada fluxo de dados em tempo-real eventualmente em curso pela interface Bluetooth.

Hard ou *Soft Handoffs*?

Do ponto de vista de número de conexões ativas durante a troca de ponto de acesso, *handoffs* podem ser divididos entre *soft* e *hard* (Seção 2.3). De maneira geral, *soft handoffs* diminuem a probabilidade de ruptura total de conectividade durante o *handoff*, o que também diminui a probabilidade de perda de dados [37]. Em compensação, o custo de se criar infra-estruturas de suporte a *soft handoffs* tende a ser mais alto devido à complexidade extra das interfaces sem fio para suportarem múltiplas conexões simultâneas. *Hard handoffs*, por outro lado, exigem interfaces menos complexas, mas precisam acontecer de forma rápida o suficiente para não serem notados. *Soft handoffs* relaxam o requisito por *handoffs* extremamente rápidos dos *hard handoffs*, o que traz alguma "folga" para que, por exemplo, aplicações tenham tempo de enviar sinalizações para o redirecionamento do tráfego de dados para o novo ponto de acesso.

No contexto específico de Bluetooth, a tecnologia já foi projetada de forma a suportar conexões simultâneas, e o baixo custo das interfaces também foi um requisito em seu projeto. As perdas de se lidar com *soft handoffs* no contexto de Bluetooth dependem do esquema de TDM usado para multiplexar o acesso às *piconets* das quais o DM faz parte. Porém, é difícil estimar tais perdas, já que o esquema de TDM utilizado é dependente de implementação e, provavelmente, segredo de negócio do fabricante da interface. Entretanto, eventuais estratégias ruins de TDM não significam necessariamente um problema do protocolo de *handoff* como um todo, mas o conjunto de aplicações finais pode ser influenciado de alguma maneira devido essa característica das interfaces (vide Seções 4.1.2 e 5.1).

Detectando a Necessidade de *Handoff*

EBs podem ser as responsáveis por monitorar e disparar processos de *handoff*, assim como DMs também podem. Existem, na verdade, diversas tarefas que precisam ser realizadas para uma troca transparente de EB durante *handoffs*, como perceber a necessidade de *handoff* ou contactar a EB de destino para solicitar conexão com o DM. Existem inúmeras maneiras de distribuir essas responsabilidades entre as entidades do sistema, onde a decisão dita "correta" deve levar em consideração os requisitos do problema e o impacto das decisões tomadas sobre esses requisitos.

Em razão do enfoque deste trabalho na característica de mobilidade do problema tratado, espera-se que os dispositivos clientes de eventuais ambientes inteligentes baseados na infra-estrutura aqui proposta sejam pequenos dispositivos pessoais, tais como PDAs, *Internet Tablets*, ou *smartphones*. Uma das características comuns a esses dispositivos é sua baixa autonomia, devido ao uso de baterias, baixo poder computacional (e.g. memória e processamento), e largura de banda reduzida. Assim, alocar a execução de funcionalidades relacionadas à gerência de handoff a tais dispositivos significa consumir parte dos recursos já limitados, além de um provável impacto em sua autonomia relacionada ao consumo extra de energia.

Dessa maneira, optou-se aqui por remover dos DMs o máximo possível de funcionalidades ligadas ao processo de *handoff*, jogando para EBs e rede cabeada a maior parte do custo relacionado à gerência de *handoffs* em benefício dos DMs. Dessa maneira, EBs monitoram o estado de DMs, controlam suas mudanças de estado, trocam entre si, via rede cabeada, informações sobre essas mudanças de estado e tentam contato com DMs que se movem entre suas áreas de cobertura. Aos DMs, mais precisamente às aplicações executando nos DMs, cabe apenas a responsabilidade de perceber o estabelecimento e perda de conexões e lidar com o tratamento de dados em tempo-real transmitidos/recebidos por, eventualmente, mais de uma conexão ativa.

Comandos HCI Necessários

A solução para *handoffs* eficientes, do ponto de vista de impactos sobre o tráfego de dados em tempo-real, introduzida neste trabalho, baseia-se apenas em comandos e características obrigatórios do Bluetooth, de acordo com sua especificação. Os procedimentos descritos neste trabalho são independentes de dispositivo, bastando que existam nas EBs e DMs interfaces Bluetooth e uma pilha de acesso ao *hardware* que implemente, pelo menos, os comandos HCI padrões descritos na Tabela 3.1. Esses comandos são suficientes para a gerência de *handoff* como definido neste trabalho. Outros comandos e eventos HCI são necessários, por exemplo, para implementar o Monitor de Link ou o mecanismo usado nos DMs para detectar o estabelecimento e perda de conexões.

Comando	Função	Contexto
HCI_Create_Connection	Estabelece uma conexão baseband entre dois dispositivos (*paging*).	Invocado por EBs durante o *handoff*.
HCI_Write_Page_Scan_Activity	Define os parâmetros do *paging*.	Pode ser chamado pelos DMs para melhorar o desempenho dos *pagings*.

Tabela 3.1: Comandos HCI usados.

Discussões Adicionais

As configurações e alocações de responsabilidades discutidas nesta seção têm como objetivo, principalmente, reduzir o tempo necessário para conexão entre DMs e EBs durante o *handoff*. Existem, entretanto, outras soluções para se buscar *handoffs* ainda mais rápidos. Segundo os trabalhos de Aman Kansal [30] e Sang-Hun Chung et al. [13], e à própria especificação do Bluetooth, uma boa estimativa do tamanho da diferença de sincronização entre os relógios dos dois dispositivos (*clock offset*) pode agilizar o processo de *paging* entre os dispositivos. Entretanto, tal ganho não está claro nem na especificação, que não faz comentários sobre economia de *slots* envolvida nesse processo, nem nos trabalhos que também propõem o uso de estimativas dos *offsets*. Nestes trabalhos, o uso de relógios sincronizados é sempre empregado junto com outras estratégias para aumento de desempenho, não ficando claro qual o ganho de se ter relógios sincronizados. Além disso, em testes próprios realizados para se investigar a real utilidade de se usar estimativas de diferença entre relógios, não foi percebida mudança visível no desempenho dos *pagings*.

Outra linha de trabalho, pouco comum, para melhorar o desempenho dos *handoffs* consiste em usar operações não padronizadas do Bluetooth para, por exemplo, escolher a EB mais próxima durante o *handoff* [13]. Como nestes trabalhos predomina a apresentação de resultados a partir de dados de simuladores, torna-se fácil aplicar e estudar o sucesso ou não de operações não padronizadas. Entretanto, para implantar esse tipo de solução mudanças podem ser necessárias desde a pilha Bluetooth até o *firmware* da interface para que as soluções propostas cumpram com suas tarefas. Esse custo extra para implantação destas soluções possui grandes chances de ser uma questão proibitiva, a depender dos cenários de

uso finais.

Neste trabalho optou-se por definir uma solução utilizando apenas operações padrão do Bluetooth, definidas na camada HCI. O objetivo é viabilizar a implantação da solução aqui proposta em sistemas embarcados, onde uma atualização de pilha ou de *firmware* é muito mais complexa que a instalação de *softwares* em camada de aplicação.

3.3 Gerência de *Handoff* - Mantendo Contato com DMs

Adicionalmente às definições de configurações e responsabilidades discutidos na seção anterior, nesta seção é descrito o protocolo pelo qual DMs são mantidos conectados às EBs à medida que movem-se através de suas áreas de cobertura. O processo de *handoff* é disparado e/ou mantido por uma determinada EB mediante duas situações:

1. quando o estado de algum DM em sua vizinhança muda de CONECTADO para SOB_HANDOFF;

2. quando algum DM está no estado SOB_HANDOFF e ligado, ao mesmo tempo, a duas de suas EBs vizinhas.

O processo de *handoff* para um DM termina quando este volta ao estado CONECTADO ou perde conexão com o sistema (estado NÃO_CONECTADO). A única informação que a EB tem sobre a movimentação do DM é uma estimativa da distância em linha reta entre si e o DM. Por isso, as notificações de mudança de estado de DMs são enviadas para todas as EBs vizinhas. A decisão de participar do *handoff* de algum DM, ou ainda de cancelar/manter participação em *handoffs* já existentes, é tomada localmente por cada EB ao receber notificações sobre mudanças de estados de DMs em sua vizinhança. O processo pelo qual EBs atualizam seus registros locais e tomam essas decisões é descrito no Algoritmo 3.2.

Algoritmo 3.2: Atualizando registros de DMs

```
1 SE(aguardandoEncaminhamentoParaDM(dadosEB)) ENTAO
2    aguardaEncaminhamentoParaDM(dadosEB, FALSO)
3    notificaEBsVizinhas(dadosEB)
4 FIM SE
5 SE(aguardandoContatoDeVizinha(dadosEB)) ENTAO
```

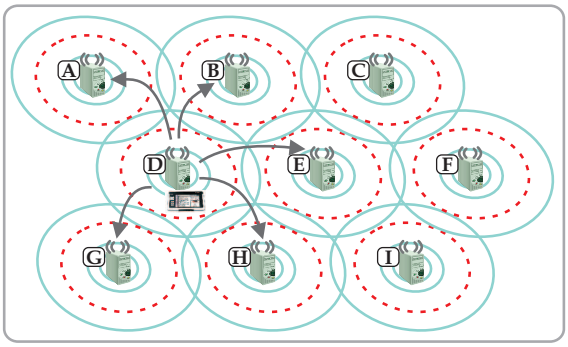
```
6    defineEstado(dadosEB,NAO_CONECTADO)
7    notificaEBsVizinhas(dadosEB)
8  FIM SE
9  SE(registroLocal.jaExisteRegistro(dadosEB)) ENTAO
10   estadoAtualDM ← estadoAtual(dadosEB)
11   estadoAnteriorDM ← registroLocal.estadoAtual(dadosEB)
12   ebRemetente ← ebRemetente(dadosEB)
13   // adiciona ou remove a EB remetente como conectada ou não ao DM,
         dependendo do estado reportado. Notificações redundantes não são
         aplicadas
14   registroLocal.atualizaRegistro(dadosEB)
15   SE(!listaEBsVizinhas.contem(ebRemetente)) ENTAO
16        escalonadorHandoffs.cancelaPedidoHandoff(dadosDM)
17   SENAO SE(ebRemetente ≠ ESTA) ENTAO
18        SE(estadoAtualDM ≠ estadoAnteriorDM) ENTAO
19             SE(estadoAtualDM = CONECTADO) OU (estadoAtualDM = SOB_HANDOFF
                   ) ENTAO
20                  SE(estadoAnteriorDM = CONECTADO) E (estadoAtualDM =
                        SOB_HANDOFF) ENTAO
21                       escalonadorHandoffs.geraPedidoHandoff(dadosDM)
22                  SENAO SE(estadoAnteriorDM = SOB_HANDOFF) E (estadoAtualDM
                        = CONECTADO) ENTAO
23                       escalonadorHandoffs.cancelaPedidoHandoff(dadosDM)
24                  FIM SE
25             SENAO //estadoAtualDM = NÃO_CONECTADO
26                  SE(!registroLocal.estaConectadoEmAlgumaEB(dadosEB)) ENTAO
27                       //também cancela eventual handoff em andamento
28                       registroLocal.removeRegistro(dadosEB)
29                  FIM SE
30             FIM SE
31        SENAO SE(estadoAtualDM = estadoAnteriorDM) E (estadoAtualDM =
              SOB_HANDOFF) ENTAO
32             //gera um pedido novo ou mantém o já existente
33             escalonadorHandoffs.geraPedidoHandoff(dadosDM)
34        FIM SE
35   FIM SE
36 SENAO
```

```
37    registroLocal.adicionaRegistro(dadosEB)
38 FIM SE
```

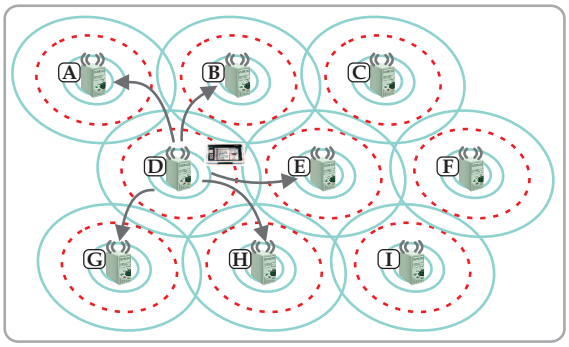
A distribuição das informações que ajudarão EBs a decidirem por sua participação ou não em algum processo de *handoff* começa quando DMs entram no sistema. Quando isto acontece, a EB que recebeu o novo cliente comunica às suas vizinhas que está em contato com um novo DM, e que este se encontra no estado CONECTADO. Enquanto conectados a alguma EB, DMs têm seus estados constantemente monitorados conforme descrito no Algoritmo 3.1. Quaisquer mudanças de estado verificadas por EBs são atualizadas localmente e comunicadas para suas vizinhas. Cada EB vizinha, ao receber essa notificação, compara o estado notificado para aquele DM contra seu último estado conhecido localmente. Caso verifiquem uma das situações onde o processo de *handoff* é disparado, um pedido de *handoff* é enviado para um escalonador local, responsável por gerenciar os pedidos de *handoff* executados por aquela EB. Pedidos de *handoff* já existentes podem ser mantidos ou cancelados, de acordo com as notificações recebidas. De forma excepcional, a EB que percebe o inicio de um *handoff* para um DM anteriormente conectado a apenas si própria, encaminhará para todas as suas vizinhas a primeira notificação que receber sobre contato de alguma vizinha com o DM sob *handoff*. O objetivo é evitar que algumas EBs continuem indefinidamente tentando contato com um DM que já foi contactado por outra EB. Assim como o Algoritmo 3.1, o Algoritmo 3.2 não possui laços nem recursões. Portanto, a complexidade do Algoritmo 3.2 é O(1), sendo este sempre executado em tempo constante.

Na Figura 3.8 é ilustrado o funcionamento do algoritmo descrito no Algoritmo 3.2. Assim que o dispositivo DM_1 consegue conexão com EB_D, esta envia uma notificação para EB_A, EB_B, EB_E, EB_G, e EB_H, indicando sua conexão com DM_1 e seu estado CONECTADO (3.8(a)). Os registros locais das EBs do sistema, após notificação de EB_D, são descritos na Tabela 3.2.

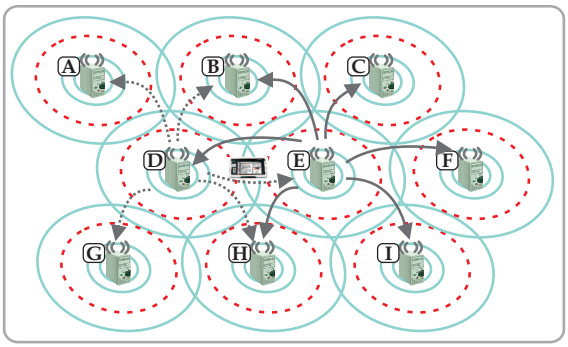
Quando EB_D percebe que DM_1 ultrapassou o ponto a partir do qual o *handoff* é disparado, o estado daquele DM é atualizado em EB_D para SOB_HANDOFF e enviado para as demais EBs (3.8(b)). Cada EB vizinha de EB_D recebe a notificação e atualiza sua base de dados local. Ao perceber a mudança de estado de DM_1 de CONECTADO para SOB_HANDOFF, as EBs EB_A, EB_B, EB_E, EB_G, e EB_H geram pedidos de *handoff* locais para aquele dispositivo. EB_D, por sua vez, não dispara processo de *handoff*, pois já está



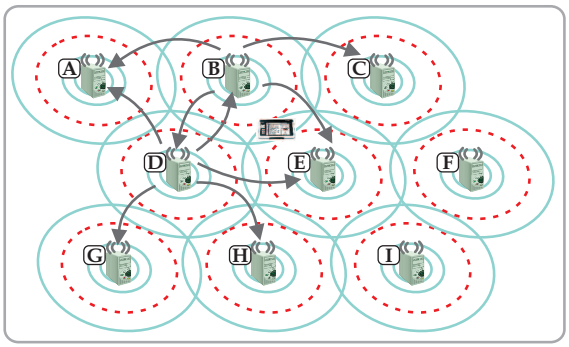
(a) EB_D notifica contato com DM_1 no estado CONECTADO.



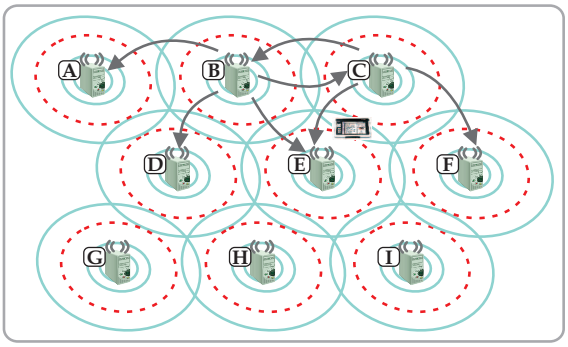
(b) EB_D mudança de estado de DM_1 para SOB_HANDOFF.



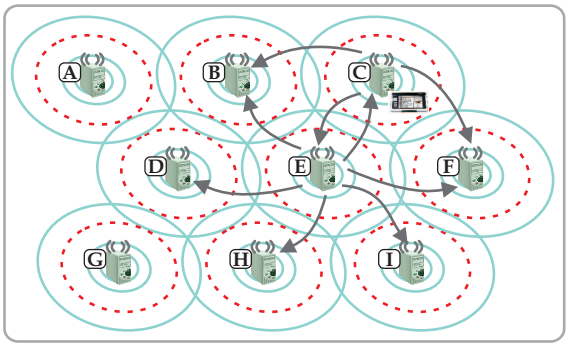
(c) EB_E notifica contato com DM_1 no estado SOB_HANDOFF. DB_D encaminha a mensagem recebida.



(d) EB_B notifica contato com DM_1 no estado SOB_HANDOFF



(e) EB_C notifica contato com DM_1 no estado SOB_HANDOFF



(f) EB_C notifica contato com DM_1 no estado CONECTADO

Figura 3.8: Mantendo o DM conectado durante movimentação.

EB	**Registros**	***Handoff* Ativo**
EB_A	$\{DM_1$, CONECTADO, $[EB_D]\}$	Não
EB_B	$\{DM_1$, CONECTADO, $[EB_D]\}$	Não
EB_C	$\{\}$	Não
EB_D	$\{DM_1$, CONECTADO, $[EB_D]\}$	Não
EB_E	$\{DM_1$, CONECTADO, $[EB_D]\}$	Não
EB_F	$\{\}$	Não
EB_G	$\{DM_1$, CONECTADO, $[EB_D]\}$	Não
EB_H	$\{DM_1$, CONECTADO, $[EB_D]\}$	Não
EB_I	$\{\}$	Não

Tabela 3.2: Registros das EBs após entrada de DM_1 no sistema.

em contato com DM_1, mas prepara-se para repassar às suas vizinhas a primeira notificação que receber sobre DM_1. Novo estado dos registros das EBs pode ser visto na Tabela 3.3.

EB	**Registros**	***Handoff* Ativo**
EB_A	$\{DM_1$, SOB_HANDOFF, $[EB_D]\}$	Sim
EB_B	$\{DM_1$, SOB_HANDOFF, $[EB_D]\}$	Sim
EB_C	$\{\}$	Não
EB_D	$\{DM_1$, SOB_HANDOFF, $[EB_D]\}$	Não
EB_E	$\{DM_1$, SOB_HANDOFF, $[EB_D]\}$	Sim
EB_F	$\{\}$	Não
EB_G	$\{DM_1$, SOB_HANDOFF, $[EB_D]\}$	Sim
EB_H	$\{DM_1$, SOB_HANDOFF, $[EB_D]\}$	Sim
EB_I	$\{\}$	Não

Tabela 3.3: Registros das EBs após mudança de estado de DM_1 para SOB_HANDOFF.

Ao aproximar-se de EB_E, DM_1 é contactado por aquela EB, que envia para suas vizinhas, EB_B, EB_C, EB_D, EB_F, EB_H, e EB_I notificação de contato com DM_1 no estado SOB_HANDOFF (3.8(c)). Neste momento, as EBs EB_A, e EB_G continuariam tentando contato com DM_1, pois não são vizinhas de EB_E e, por isso, não receberam a notificação do contato com DM_1 (Tabela 3.4(a)). Entretanto, EB_D encaminha para suas vizinhas a primeira notificação recebida de alguma vizinha sobre seu contato com DM_1, neste caso, a mensagem de EB_E. Somente as EBs que não têm contato direto com EB_E processam essa mensagem encaminhada. Segundo o Algoritmo 3.2, EB_B e EB_H vão processar a mesma mensagem duas vezes (uma recebida diretamente de EB_E e outra encaminhada por EB_D), o que não deve provocar mudanças no registro de DM_1 em cada uma delas. Já

as EBs EB_A e EB_G, quando processarem a mensagem repassada por EB_D cancelarão o *handoff* em andamento para DM_1. Os estados dos registros das EBs no sistema, antes e depois do repasse da mensagem de EB_E por EB_D, são descritos nas Tabelas 3.4(a) e 3.4(b), respectivamente.

EB	**Registros**	***Handoff* Ativo**
EB_A	{DM_1, SOB_HANDOFF, [EB_D]}	Sim
EB_B	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Sim
EB_C	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_D	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Não
EB_E	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Não
EB_F	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_G	{DM_1, SOB_HANDOFF, [EB_D]}	Sim
EB_H	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Sim
EB_I	{DM_1, SOB_HANDOFF, [EB_E]}	Não

(a) Estado dos registros das EBs do sistema antes do encaminhamento de mensagem de EB_E.

EB	**Registros**	***Handoff* Ativo**
EB_A	{DM_1, SOB_HANDOFF, [EB_D]}	Não
EB_B	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Sim
EB_C	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_D	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Não
EB_E	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Não
EB_F	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_G	{DM_1, SOB_HANDOFF, [EB_D]}	Não
EB_H	{DM_1, SOB_HANDOFF, [EB_D,EB_E]}	Sim
EB_I	{DM_1, SOB_HANDOFF, [EB_E]}	Não

(b) Estado dos registros das EBs do sistema após encaminhamento de mensagem de EB_E.

Tabela 3.4: Estados dos registros da EBs antes e depois do repasse da mensagem de EB_E por EB_D.

Continuando sua trajetória, DM_1 move-se em direção a EB_B, que consegue contato com DM_1 através do pedido de *handoff* ainda ativo para aquele dispositivo. Quando isso acontece, EB_B envia para EB_A, EB_C, EB_D, e EB_E notificação sobre seu contato com DM_1 no estado SOB_HANDOFF. Enquanto DM_1 se movia, EB_D continuou monitorando o dispositivo. Ao perceber que perdeu contato com DM_1, EB_D não envia ainda a notificação devida para suas vizinhas. Ao invés disso, aguarda notificação de contato de uma de suas vizinhas com DM_1. Somente após receber a notificação de EB_B, EB_D envia para suas vizinhas a notificação de que perdeu contato com DM_1 (3.8(d)). Os estados dos registros após essa transição de EBs podem ser vistos na Tabela 3.5.

Neste momento, DM_1 pode continuar movendo-se na direção de EB_E ou EB_B, onde teria contato perdido naturalmente com a EB oposta e assumiria estado CONECTADO, ou ainda pode mover-se em direção a EB_C ou EB_D, que possuem pedidos ativos de *handoff* para DM_1. Assumindo que DM_1 mova-se na direção de EB_C, esta EB notificará EB_B, EB_E, e EB_F sobre seu contato com DM_1 no estado SOB_HANDOFF, enquanto EB_B

EB	**Registros**	***Handoff* Ativo**
EB_A	{DM_1, SOB_HANDOFF, [EB_B]}	Não
EB_B	{DM_1, SOB_HANDOFF, [EB_B,EB_E]}	Não
EB_C	{DM_1, SOB_HANDOFF, [EB_E,EB_B]}	Sim
EB_D	{DM_1, SOB_HANDOFF, [EB_B,EB_E]}	Sim
EB_E	{DM_1, SOB_HANDOFF, [EB_B,EB_E]}	Não
EB_F	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_G	{}	Não
EB_H	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_I	{DM_1, SOB_HANDOFF, [EB_E]}	Não

Tabela 3.5: Registros das EBs após transição de DM_1 de EB_D para EB_B.

aguardará a primeira mensagem de algum contato com DM_1 para então notificar para EB_A, EB_C, EB_D, e EB_E sua perda de contato com aquele dispositivo. Os novos estados dos registros das EBs após mais essa mudança são descritos na Tabela 3.6. Por fim, DM_1 continua sua trajetória em direção a EB_C, que percebe sua aproximação e atualiza localmente o estado local daquele dispositivo para CONECTADO. A mudança é então comunicada às vizinhas EB_B, EB_E, e EB_F, seguida da notificação de EB_E sobre a perda de contato com DM_1. O resultado é a suspensão dos processos de *handoff* ativos para aquele DM em EB_B e EB_F (3.8(f)). Os estados dos registros das EBs do sistema, após essa última mudança, são descritos na Tabela 3.7.

EB	**Registros**	***Handoff* Ativo**
EB_A	{}	Não
EB_B	{DM_1, SOB_HANDOFF, [EB_C,EB_E]}	Sim
EB_C	{DM_1, SOB_HANDOFF, [EB_C,EB_E]}	Não
EB_D	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_E	{DM_1, SOB_HANDOFF, [EB_C,EB_E]}	Não
EB_F	{DM_1, SOB_HANDOFF, [EB_C,EB_E]}	Sim
EB_G	{}	Não
EB_H	{DM_1, SOB_HANDOFF, [EB_E]}	Não
EB_I	{DM_1, SOB_HANDOFF, [EB_E]}	Não

Tabela 3.6: Registros das EBs após transição de DM_1 de EB_B para EB_C.

EB	**Registros**	***Handoff* Ativo**
EB_A	{}	Não
EB_B	{DM_1, CONECTADO, [EB_C]}	Não
EB_C	{DM_1, CONECTADO, [EB_C]}	Não
EB_D	{}	Não
EB_E	{DM_1, CONECTADO, [EB_C]}	Não
EB_F	{DM_1, CONECTADO, [EB_C]}	Não
EB_G	{}	Não
EB_H	{}	Não
EB_I	{}	Não

Tabela 3.7: Registros das EBs após mudança de estado de DM_1 de SOB_HANDOFF para CONECTADO.

3.4 Discussões Adicionais

Um problema adicional é naturalmente levantado ao discutir a solução proposta para gerência de *handoff* em Bluetooth: escalabilidade. É natural perceber que à medida que se aumenta a quantidade de DMs fazendo *handoff* para a mesma EB ao mesmo tempo, aumenta-se a proba-bilidade de *handoff* mau sucedido devido atraso da EB para tentar contato com o cliente. Também é natural perceber que a capacidade de interfaces Bluetooth de até sete conexões simultâneas não é suficiente para evitar uma saturação rápida das EBs. Apesar da questão de escalabilidade ter tido sua parcela de atenção durante a elaboração deste trabalho, entendemos que este problema não está diretamente ligado ao foco do trabalho. Dessa maneira, o conhecimento adquirido sobre como tornar o sistema escalável, do ponto de vista de eficiência dos *handoffs* e capacidade de comportar clientes, não está sendo proposto aqui como parte da solução. Porém, no Apêndice C é apresentado um arcabouço de infra-estrutura, baseada no protocolo aqui proposto, cujo foco são as questões de escalabilidade que acabaram de ser colocadas. A extensão apresentada no Apêndice C baseia-se na possibilidade de aumentar a quantidade de interfaces nas EBs e no acoplamento parcial entre os processos das EBs para gerência de *handoffs* com as aplicações finais do sistema. Com o aumento do número de interfaces nas EBs aumenta-se também a capacidade daquela EB de atender mais clientes ao mesmo tempo. Diminui-se também a probabilidade de *handoffs* mau sucedidos devido atraso da EB em processar pedidos simultâneos pois, com um maior número de interfaces na EB, vários *handoffs* podem ser atendidos ao mesmo tempo. Através

do acoplamento com as aplicações finais do sistema, pode-se implementar mecanismos para controle de admissão nas EBs, o que pode evitar tentativas de contato com DMs sob *handoff* quando não existem recursos suficientes para aquele dispositivo na EB destino.

Capítulo 4

Avaliação da Solução

Conforme visto no Capítulo 3, a apresentação da solução proposta para a gerência de *handoffs* em *Bluetooth* foi dividida em duas partes: as configurações e atribuições de responsabilidades entre EBs e DMs para transições diretas de ponto de acesso; e o protocolo pelo qual EBs disparam/cancelam processos de *handoff* e mantém DMs conectados ao sistema enquanto estes se movem.

A avaliação da solução proposta segue a mesma divisão. Para avaliar a transição direta entre EBs um protótipo foi implementado, enquanto o funcionamento do protocolo das EBs foi modelado e verificado utilizando Redes de Petri Coloridas (CPN [1]). Detalhes da implementação do protótipo e cenários de testes são descritos na Seção 4.1, enquanto a análise sobre o modelo CPN é apresentada na Seção 4.2.

4.1 Transições entre EBs

Um protótipo, contemplando as entidades lógicas, configurações, e responsabilidades definidas na Seção 3.2, foi implementado para avaliar o real impacto da troca de EBs sobre fluxos de dados em tempo-real em trânsito pelas interfaces *Bluetooth* de EBs e DMs. Além da implementação do aparato para gerência de *handoffs* definido na Seção 3.2, também foi desenvolvido um sistema simples para transmitir *streams* de um servidor na rede cabeada para dispositivos móveis que se movem entre as áreas de cobertura de duas EBs. De fato, a infra-estrutura implementada para os testes é uma implementação parcial da infra-estrutura

[1] *Coloured Petri Nets*

descrita no Apêndice C.

Uma visão geral do ambiente desenvolvido para os testes é ilustrada na Figura 4.1. No ambiente montado, EBs e servidor de *streams* estão ligados à mesma rede local. O servidor armazena e transmite via *streaming* conteúdo multimídia pré-gravado. Junto com o *software* para gerência de *handoff* rodando em cada EB funciona uma aplicação externa, responsável por atuar como consumidora do *stream* em benefício do DM (vide Seção 4.1.1 e Apêndice C). Tanto o servidor de *streams* quanto EBs são computadores de mesa convencionais enquanto um *notebook* foi utilizado como DM. Um resumo das características físicas de cada uma destas entidades é descrito na Tabela 4.1.

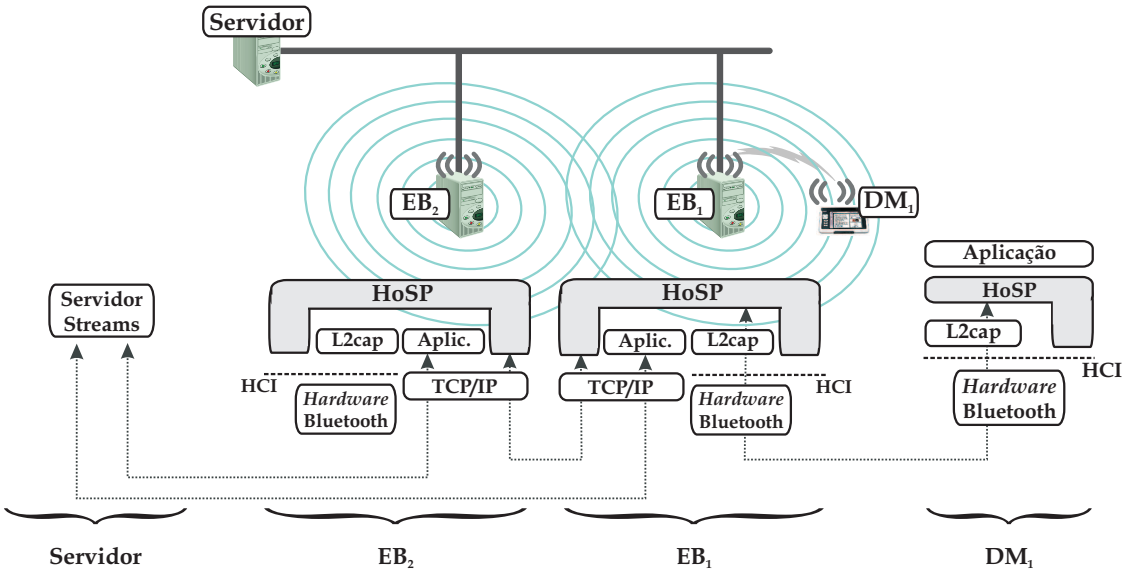


Figura 4.1: Visão geral da estrutura criada para testes.

4.1.1 O Ambiente de Testes

Todos os *softwares* usados foram desenvolvidos em C++ [68]. O servidor de *streams* utilizado é um dos programas exemplo (*testMP3Streamer*) que acompanham a biblioteca Live555 [41]. Este programa exemplo lê repetidamente um arquivo MP3, jogando na rede local um *stream* do seu conteúdo usando RTP/RTCP num endereço *multicast*. Nas EBs, as funcionalidades necessárias para os testes foram divididas em módulos, quase todos implementadas como componentes de uma mesma aplicação, HoSP (sigla para *Handing off Streams Profile*), conforme esquema na Figura 4.2(a), à exceção da aplicação responsável

Entidade	Dispositivo	Configuração	Notas Adicionais
Servidor	Computador de mesa	Athlon 64; Ubuntu Linux; *Kernel* 2.6.15-28; 2GB de RAM; *Ethernet* 100Mbps	
EBs	Computador de mesa	Athlon 64; Ubuntu Linux; *Kernel* 2.6.22-rc6; 2GB de RAM; *ethernet* 100Mbps; Interfaces *Bluetooth* 2.0 + EDR Broadcom modelos USB-200 e USB-250; Pilha *Bluetooth* BlueZ v3.12	Foram instaladas versões específicas do BlueZ [51] e do *Kernel*, diferentes das versões atualmente estáveis para o Ubuntu, para poder usar os recursos de EDR das interfaces *Bluetooth*.
DM	*Notebook*	Pentium 4 HT; Ubuntu Linux; *Kernel* 2.6.22-rc6; 512MB de RAM; Interfaces *Bluetooth* 2.0 + EDR Broadcom modelos USB-200 e USB-250; Interface *Bluetooth* 2.0 + EDR CSR; Pilha *Bluetooth* BlueZ v3.12	

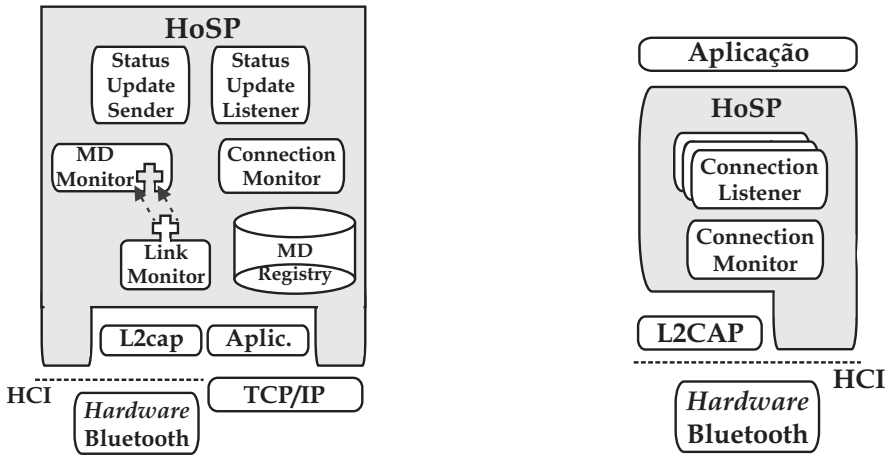
Tabela 4.1: Resumo das características dos dispositivos usados para testes.

por interagir com o servidor de *streams*. Esta aplicação é uma versão modificada de outra aplicação exemplo (*testMP3Receiver*) que acompanha a Live555, alterada para encaminhar para o HoSP os pacotes RTP recebidos a partir do *testMP3Streamer*.

Nas EBs, os módulos *Status Update Sender* e *Status Update Listener* são usados para, respectivamente, enviar e escutar mensagens de mudanças de estados em DMs. O *MD Monitor* foi implementado conforme Algoritmo 3.1. Para implementar o *Link Monitor*, a medida de RSSI [2] foi escolhida como heurística para detecção de mudanças entre os estados CONECTADO e SOB_HANDOFF. Para os testes realizados, o ponto limiar para disparo do *handoff* foi definido em aproximadamente seis metros de distância a partir da EB, sendo o valor do RSSI nesse ponto usado para disparar as transições de estado entre CONECTADO e SOB_HANDOFF. O MD Registry foi implementado como uma lista em memória dos dispositivos atualmente conectados à EB.

Para ter acesso às informações e à conectividade necessária por seus módulos, o HoSP

[2] *Receive Signal Strength Indication.*



(a) Componentes lógicos das EBs usadas para testes.

(b) Componentes lógicos dos DMs usados para testes.

Figura 4.2: Estrutura lógica das EBs implementadas para testes.

têm acesso direto à camada HCI e à rede local TCP/IP. No enlace com DMs a troca de dados é feita via L2CAP. Por fim, o último módulo da EB é a aplicação responsável pelo contato com o servidor de *streams*. Este módulo, implementado a partir do *testMP3Receiver*, encarrega-se de captar na rede TCP/IP os pacotes RTP transmitidos pelo servidor e passá-los para o HoSP para que este faça o encaminhamento dos dados via enlace sem fio com o DM.

Os módulos que compõem o *software* que executa no cliente são ilustrados na Figura 4.2(b). O *Connection Monitor* constantemente verifica o estabelecimento e perda de conexões com EBs. Em sua implementação, o Connection Monitor delega para a ferramenta *hcitool*, que acompanha o BlueZ, a tarefa de consultar a interface *Bluetooth* sobre as conexões atualmente ativas. O Connection Monitor interpreta a saída do *hcitool*, mantendo um pequeno histórico de conexões previamente estabelecidas, suficiente para identificar e lançar eventos de estabelecimento e perda de conexões. O aspecto mais relevante da camada HoSP nos DMs, do ponto de vista dos testes executados, é sua capacidade de criar e destruir *Connection Listeners* à medida que conexões *baseband* são estabelecidas ou perdidas. Cada Connection Listener escuta dados que chegam pela respectiva conexão *baseband*, repassando-os para a aplicação que faz o tratamento apropriado. Por fim, a aplicação final não implementa nem *buffers* de recepção nem qualquer mecanismo para renderização dos dados recebidos. Os pacotes RTP são processados tão logo sejam recebidos, onde o proces-

samento consiste em extrair e escrever sua carga útil na saída padrão.

Cada teste foi acompanhado de uma avaliação subjetiva. No escopo dos testes executados, a avaliação subjetiva consistiu em reproduzir os *streams* recebidos à medida que estes chegavam no dispositivo cliente. As avaliações dos *streams* recebidos são ditas subjetivas pois cabe à pessoa que executa o teste acompanhar a reprodução do *stream* e dar sua opinião sobre a qualidade da reprodução. Para as avaliações subjetivas uma outra aplicação, o *gstreamer* [21], foi utilizada para a renderização dos dados. Para a reprodução dos dados pelo *gstreamer*, a saída da aplicação era direcionada via *pipes* para a entrada padrão do *gstreamer*. Nas avaliações subjetivas o áudio recebido no DM era imediatamente reproduzido em seus auto-falantes. A qualidade da reprodução era então avaliada pelo usuário.

Uma vez discutidos os elementos lógicos implementados/usados/modificados, o funcionamento geral do ambiente de testes pode então ser descrito. O servidor de *streams* continuamente joga na rede local o *stream* do arquivo MP3 lido usando RTP/RTCP via *multicast*. Assim que o DM se conecta à EB, o Connection Monitor da EB detecta a conexão do cliente, abre uma conexão L2CAP com aquele dispositivo e solicita à aplicação o início do repasse do *stream*. Ao mesmo tempo, são criadas instâncias do MD Monitor e do Link Monitor para monitorar mudanças de estado do DM enquanto conectado àquela EB. A aplicação então começa a escutar a rede cabeada pelos pacotes RTP que estão sendo enviados pelo servidor. Cada pacote é então repassado para o HoSP, que encapsula os pacotes RTP em mensagens L2CAP para serem enviadas para o DM. Do lado do DM, cada pacote, assim que recebido pelo HoSP, é repassado para a aplicação. A aplicação verifica o campo *Sequence Number* do pacote RTP [60] para checar numa tabela local se aquele pacote já foi recebido. Caso esta seja a primeira cópia daquele pacote a chegar no DM, seu número é guardado na tabela anteriormente consultada e o pacote é processado pela aplicação.

Quando o Link Monitor reporta uma mudança no estado do DM para SOB_HANDOFF, o MD Monitor envia uma mensagem de mudança de estado para a segunda EB com o BD_ADDR do DM. Por simplicidade, a segunda EB automaticamente dispara o *handoff* para o dispositivo com o BD_ADDR indicado, ao invés de comparar circunstâncias para o disparo ou não de *handoffs*, conforme descrito no Código Fonte 3.2. Assim que consegue conexão com o DM, o HoSP da segunda EB estabelece conexão L2CAP com o dispositivo, requisita para a aplicação o repasse dos dados do *stream* sendo consumido pelo DM, e inicia

o encaminhamento dos dados via enlace sem fio. Do lado do cliente, o Connection Monitor detecta a presença de nova conexão e reporta o evento para o HoSP, que cria um novo Connection Listener para a nova conexão e passa a receber dados vindos da nova EB. Os dados são repassados para a aplicação, que os trata da maneira descrita anteriormente.

4.1.2 Cenários de Testes

Nesta seção são apresentados os resultados obtidos a partir de testes no ambiente descrito na Seção 4.1.1. Para os testes aqui descritos, foi utilizado o mesmo arquivo MP3, codificado numa taxa de 320 Kbps, como fonte do *stream* consumido pelo DM. Para efeitos de comparação com os resultados a seguir, na Figura 4.3 são mostrados o gráfico da chegada de pacotes RTP desse *stream* no DM (Figura 4.3(a)) e o gráfico com os atrasos de cada pacote em relação ao pacote anterior (Figura 4.3(b)). No gráfico de chegada de pacotes, Figura 4.3(a), são mostrados os momentos em que chegaram cada pacote RTP durante o tempo que o teste foi executado. No eixo das ordenadas são representados os números dos pacotes na seqüencia aleatória gerada pelo RTP, e no eixo das abscissas é representado o tempo. O período de tempo ilustrado na Figura 4.3(a) inicia-se no momento zero e encerra-se após 15 segundos de teste. A linha diagonal e sem quebras exibida no gráfico demonstra a recepção perfeita do *stream* usado nos testes e pode ser usada como parâmetro para comparação com os demais gráficos. O resultado do segundo gráfico, ilustrado na Figura 4.3(b), complementa a informação do primeiro gráfico. Na Figura 4.3(b) são ilustradas as variações nos intervalos de chegada dos mesmos pacotes representados na Figura 4.3(a). Neste gráfico, no eixo das abscissas são representados os números dos pacotes RTP recebidos durante o teste. No eixo das ordenadas são representados os intervalos entre as chegadas de pacotes consecutivos da seqüencia. Isto é, para cada pacote recebido durante o teste foi medida a diferença entre o momento de sua chegada e o momento da chegada do pacote anterior da seqüencia. No gráfico são ilustrados os atrasos de todos os pacotes de seqüencia. Naturalmente existe alguma variação entre os tempos de chegada de diferentes pacotes. Entretanto, existe alguma tolerância quanto a estas variações devido ao tempo que o conteúdo de cada pacote leva pra ser processado pela aplicação que faz sua reprodução. O resultado ilustrado na Figura 4.3(b) representa uma reprodução perfeita do *stream* usado. Estes gráficos são utilizados aqui como medida de comparação. A informação sobre a qualidade da reprodução do *stream* não é de-

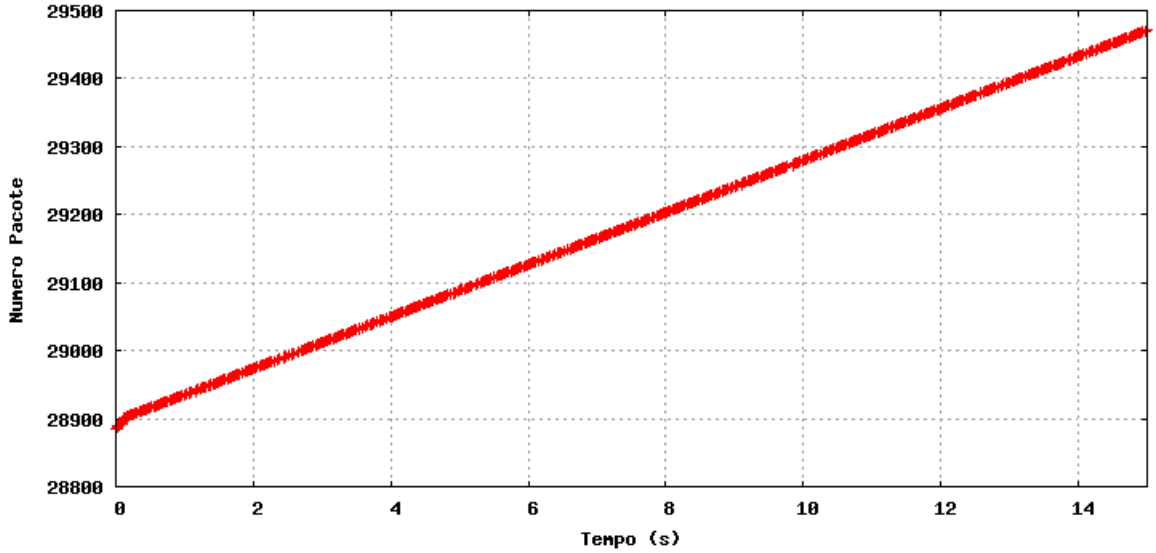
duzida diretamente a partir do gráfico, mas sim da avaliação subjetiva que acompanhou o teste que gerou os gráficos da Figura 4.3, assim como e os demais testes.

Para gerar esses gráficos comparativos foram utilizadas tanto no DM quanto na EB interfaces Broadcom modelo USB-200. Os dados recebidos nesse teste foram redirecionados para o *gstreamer*, que reproduziu perfeitamente o *stream* do MP3.

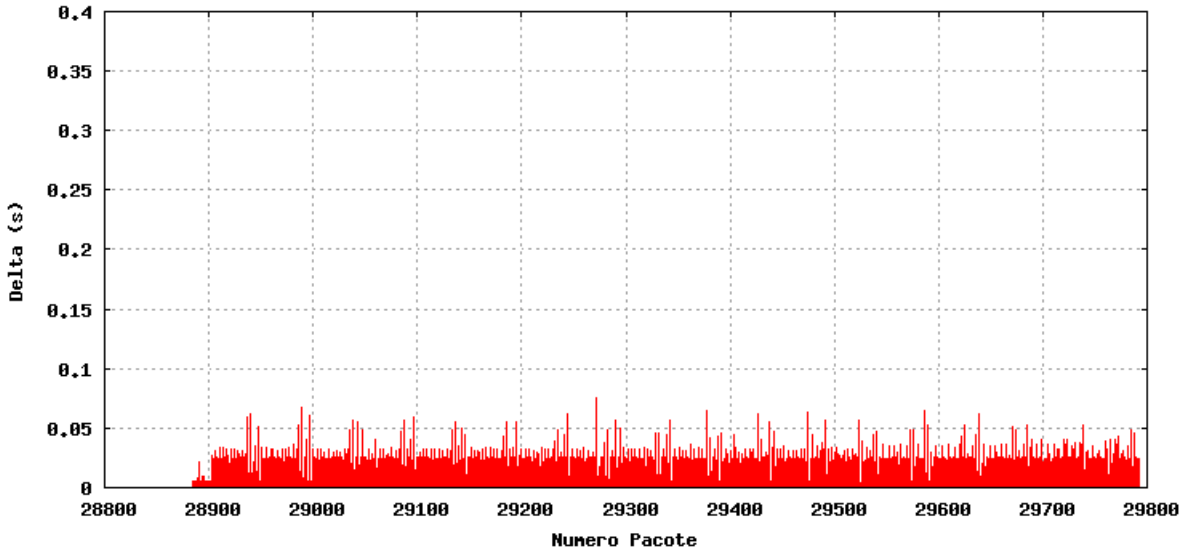
Antes do teste definitivo de *handoff*, foram feitos alguns experimentos para encontrar valo-res para o *page scan window* e *page scan interval* de forma a manter suave a transmissão do *stream* e diminuir o tempo de *paging* durante *handoffs*. Na Figura 4.4 são ilustrados os mesmos gráficos de chegada de pacotes e de atrasos inter-pacotes, mostrados anteriormente, para os valores escolhidos de *page scan window* e *page scan interval* (18 e 256 *slots*, respectivamente). Uma relação completa dos testes para determinar os parâmetros do *paging* usados nos testes a seguir pode ser encontrada no Apêndice A, junto com uma breve discussão sobre o critério usado para a escolha destes valores. Mesmo com o visível aumento na oscilação nos intervalos entre chegadas de pacotes, a avaliação subjetiva do *stream* que originou os gráficos da Figura 4.4 revelou uma reprodução suave do áudio, sem falhas perceptíveis.

Para o teste de *handoff*, cada teste aconteceu da seguinte maneira: primeiro o DM conecta-se à primeira EB e inicia a recepção do *stream*. Um usuário então move-se junto com o DM para distante da EB até o ponto onde o *handoff* era disparado. A primeira EB, após notificar para a segunda sobre a mudança de estado do DM, iniciava um temporizador de 5s para desconecta-la do DM. O usuário conhece o local de sua trajetória onde o *handoff* é disparado, indicado com uma marcação no chão. Uma vez ultrapassado esse ponto, o usuário mudava sua direção, voltando para próximo da segunda EB.

O resultado do teste é mostrado na Figura 4.5. Para esse teste, foram utilizadas interfaces Broadcom modelo USB-200 tanto nas EBs quanto no DM. Neste teste, o *handoff* começa aproximadamente no momento 10s e dura em torno de 5s, quando a conexão com a primeira EB é perdida e o DM permanece conectado apenas à segunda EB. Na Figura 4.5(a) são mostradas as chegadas de cada pacote RTP da seqüencia e sua origem (primeira EB ou segunda EB), enquanto na Figura 4.5(b) são mostrados os intervalos entre as chegadas de cada pacote. Durante o período onde o DM está conectado a mais de uma EB, são mostrados também na Figura 4.5(a) os pacotes descartados. Conforme os gráficos sugerem, se comparados



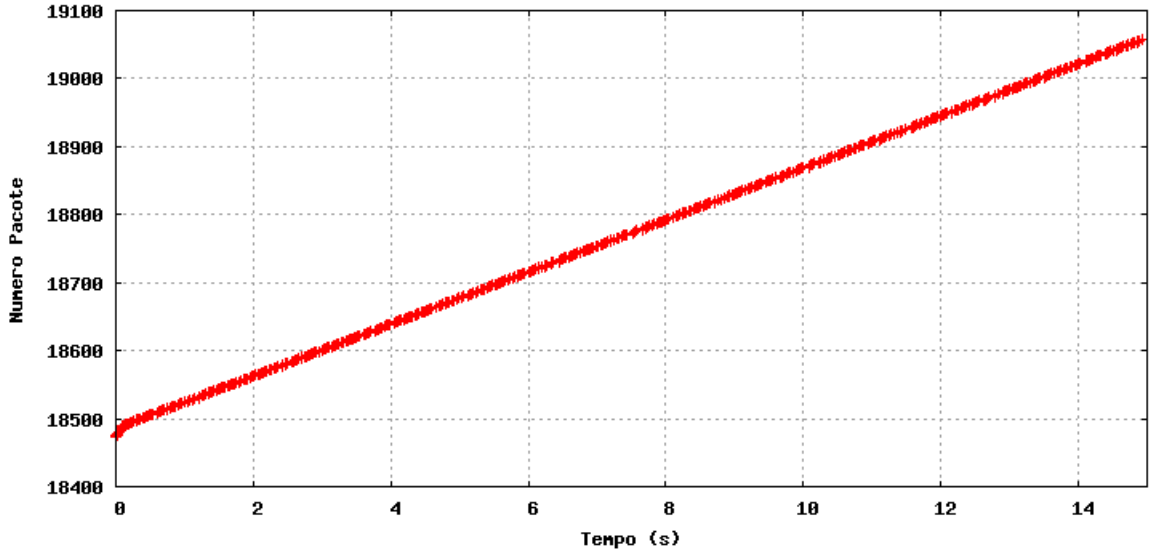
(a) Chegada dos pacotes RTP no DM.



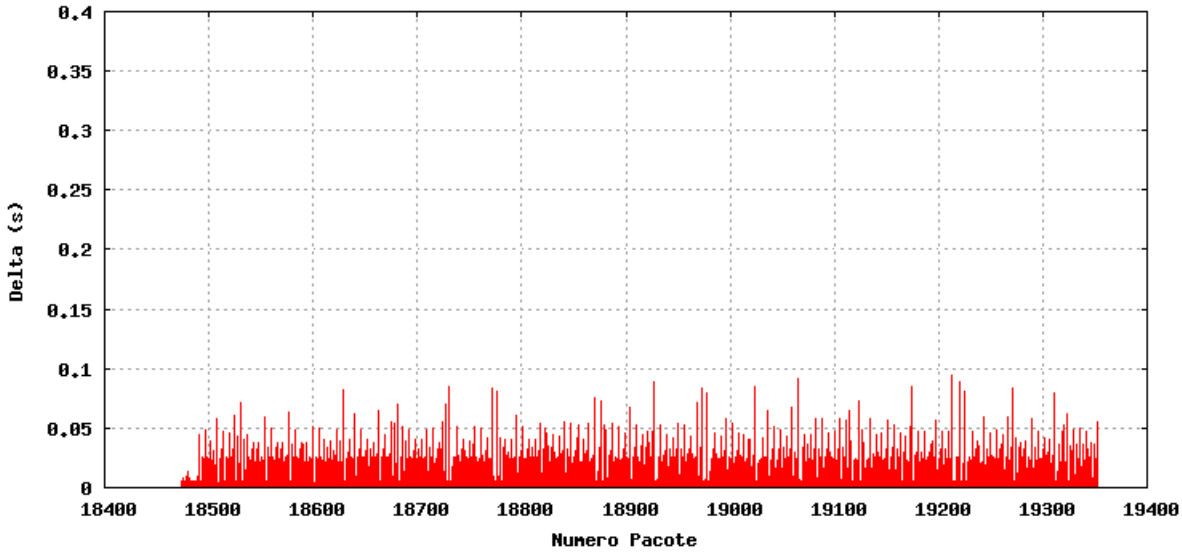
(b) Intervalos entre as chegadas de cada pacote RTP no DM.

Figura 4.3: Dados de referência da transmissão do *stream* de 320 Kbps para o DM, sem *handoff*.

com os gráficos da Figura 4.4, não houve abalo perceptível na recepção do *stream*, mesmo durante o *paging* e o período de *handoff*. O fluxo de pacotes manteve-se constante enquanto o DM estava conectado às duas EBs, e a filtragem de pacotes redundantes no DM cumpriu com seu papel, descartando pacotes RTP que chegaram em duplicata e/ou atrasados. A avali-



(a) Chegada de pacotes RTP usando no DM *page scan window* = 18, e *page scan interval* = 256

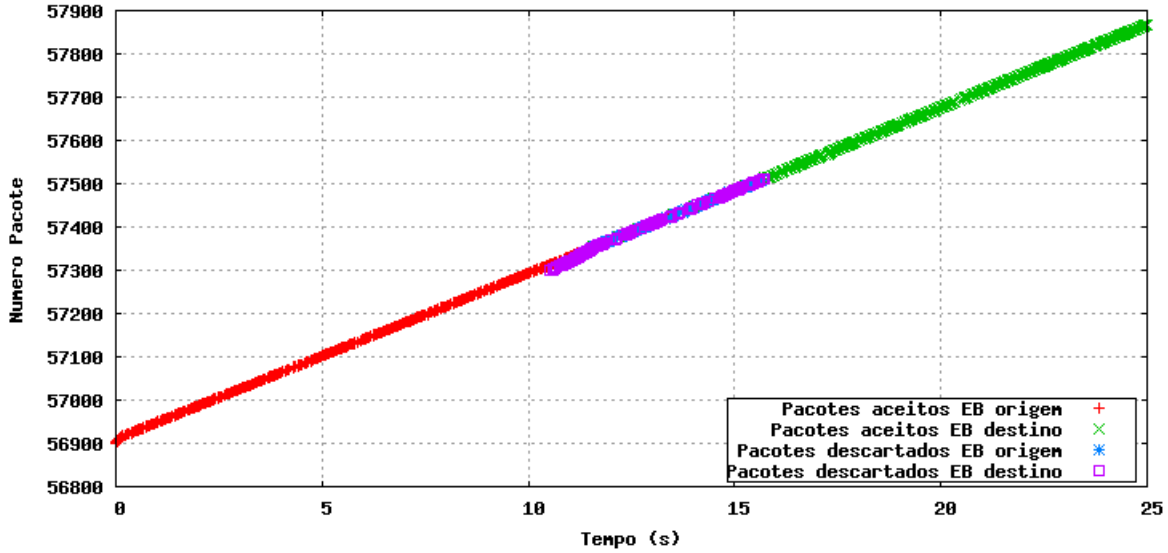


(b) Intervalos entre as chegadas de pacotes RTP usando no DM *page scan window* = 18, e *page scan interval* = 256

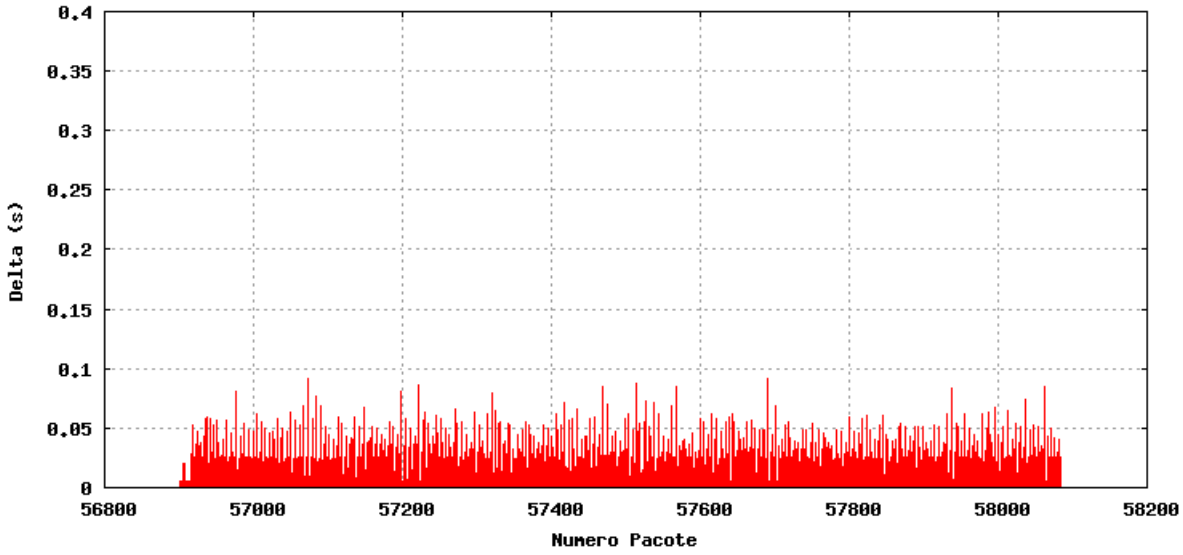
Figura 4.4: Dados sobre a chegada dos pacotes RTP após ajuste dos parâmetros do *paging*.

ação subjetiva confirmou uma reprodução suave do *stream*, sem alterações perceptíveis no áudio recebido.

Este mesmo teste foi repetido usando interfaces Broadcom modelo USB-250 e o Internet Tablet N800 da Nokia. Este último possui interface Bluetooth 2.0 + EDR integrada. Nos



(a) Chegada dos pacotes RTP no DM durante *handoff*.



(b) Intervalos entre as chegadas de cada pacote RTP no DM durante *handoff*.

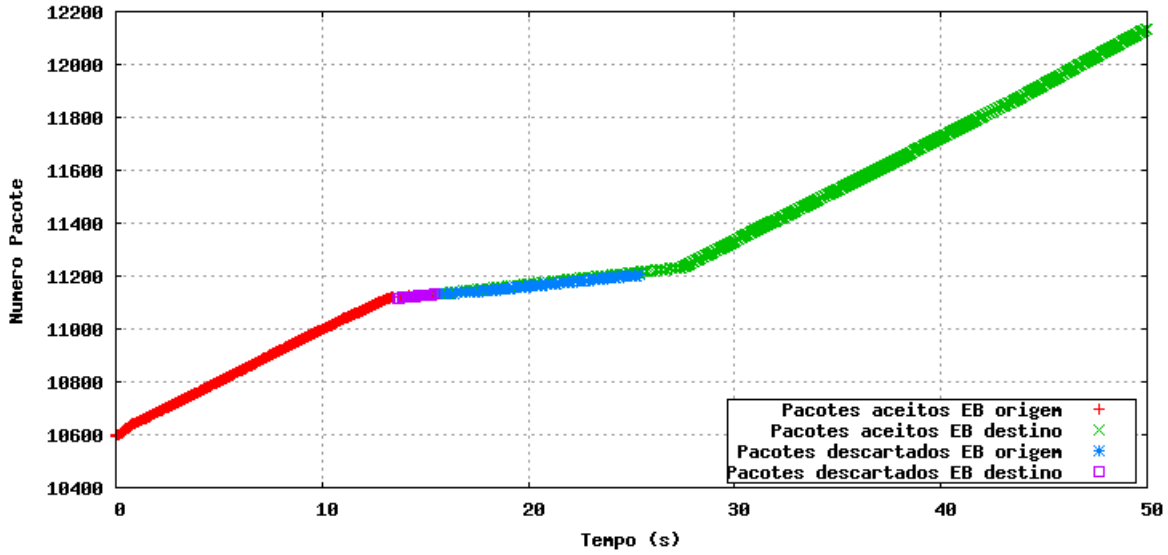
Figura 4.5: Dados da transmissão do *stream* de 320 Kbps para o DM durante *handoff*.

dois casos, os resultados foram semelhantes aos resultados ilustrados na Figura 4.5. Devido à grande semelhança dos gráficos gerados a partir desses testes com os gráficos da Figura 4.5, sua exposição neste trabalho foi considerada redundante.

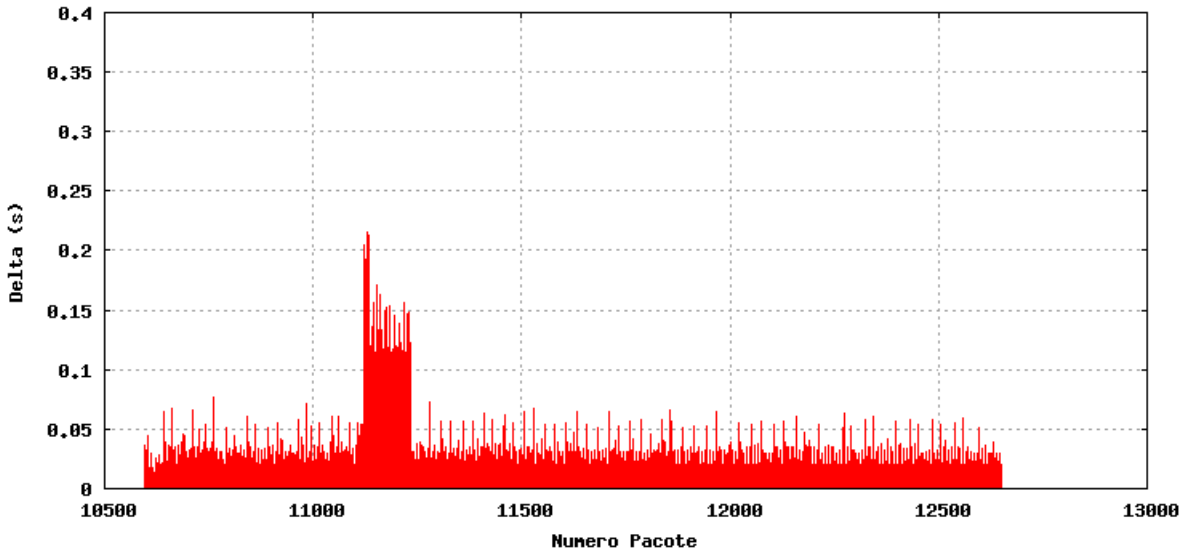
Num segundo teste, a interface *Bluetooth* do DM foi substituída pela interface CSR [3]. Os

[3] *Cambridge Silicon Radio*.

resultados desse novo teste, usando o mesmo cenário descrito anteriormente (com exceção da interface que foi substituída) são mostrados na Figura 4.6.



(a) Chegada dos pacotes RTP no DM durante *handoff* usando interface CSR.



(b) Intervalos entre as chegadas de cada pacote RTP no DM durante *handoff* usando interface CSR.

Figura 4.6: Dados da transmissão do *stream* de 320 Kbps para o DM durante *handoff* usando interface CSR.

Neste experimento, a recepção do *stream* começa de maneira normal, similar aos testes anteriores. Entretanto, durante o período onde o DM está conectado às duas DMs, a recepção

de dados é alterada por uma redução na largura de banda de cada conexão. No gráfico, a redução na largura de banda de recepção do DM é revelada pela diminuição da inclinação da linha do gráfico em relação ao eixo das abscissas. Uma linha com menor inclinação em relação ao eixo x significa que menos pacotes estão sendo recebidos por unidade de tempo. Durante todo o tempo que o DM esteve conectado a mais de uma EB a largura de banda de recepção do DM foi reduzida. Esta redução na largura de banda é evidenciada também no segundo gráfico, ilustrado na Figura 4.6(b). Nesta figura, os pacotes recebidos enquanto o DM estava conectado a mais de uma EB apresentam maiores intervalos de inter-chegadas. Esse aumento é ilustrado pelo pico no gráfico. A avaliação subjetiva para esse teste revelou consideráveis ruídos e pausas na reprodução do áudio durante o *handoff*. A queda na qualidade das conexões *baseband*, verificada na Figura 4.6, provavelmente é conseqüência de uma estratégia de TDM fraca implementada na interface CSR, reflexo de uma interface de má qualidade.

4.2 Validação do Protocolo

Nesta seção são apresentadas a especificação e validação formais, usando modelos de redes de Petri coloridas, do protocolo proposto na Seção 3.3. Para a edição e análise dos modelos foi utilizada a ferramenta CPN Tools [4]. A importância da especificação formal está no fato de garantir que o protocolo se comporta da forma esperada em qualquer situação de funcionamento e prever possíveis comportamentos emergentes em situações não previstas na definição do mesmo. Além disso, um modelo formal serve de documentação precisa para divulgação, publicação, reprodução, mudanças, e adaptações do protocolo, visto que permite um melhor entendimento deste.

4.2.1 Modelagem

Na modelagem do protocolo, os DMs estão inicialmente desconectados. Eventualmente estes dispositivos podem conectar-se a uma EB. A partir do momento que o dispositivo está conectado a uma EB, ele pode se movimentar, eventualmente disparando o processo de *handoff*. Como visto no Capítulo 3, disparar o processo de *handoff* envolve procedimentos

[4] http://wiki.daimi.au.dk/cpntools/cpntools.wiki

de mais baixo nível, como a estimação da distância entre EB e DM, que são abstraídos do modelo. É modelado, entretanto, o conceito de estado do dispositivo, que é acompanhado através dos registros das EBs. Os estados podem ser *OFF* (desconectado), *CONNECTED* (conectado) e *HANDOFF*. A mudança de estados é percebida pela EB onde o DM está conectado que, por sua vez, atualiza seu registro e comunica as EBs vizinhas. Portanto, são tratados no modelo a atualização dos registros e as mensagens de notificação.

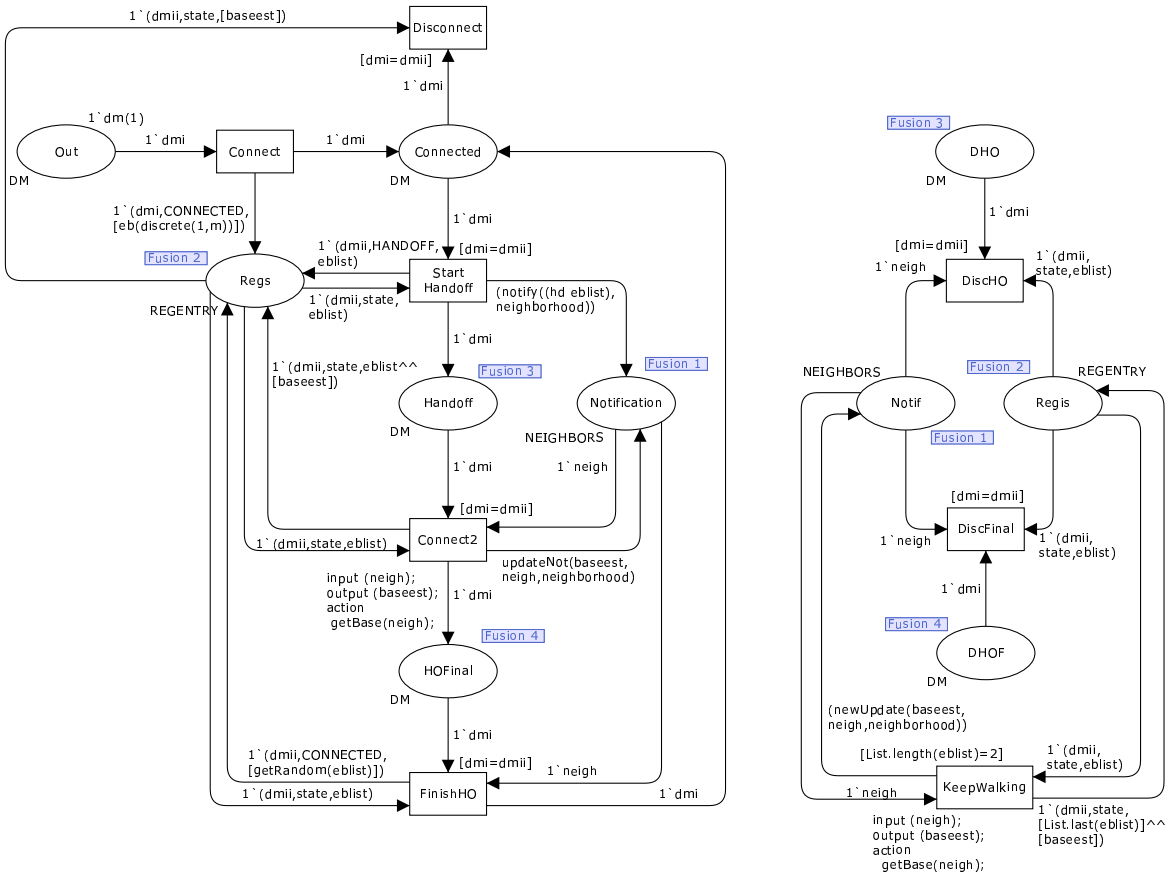


Figura 4.7: Modelo CPN para o protocolo de *handoff* entre EBs.

Na Figura 4.7 é ilustrado o modelo em redes de Petri para o protocolo. O lugar inicial do modelo é *Out*, significa que o dispositivo está desconectado, ou seja, fora da rede. A transição *Connect* representa o procedimento de conexão do dispositivo, o que é feito aleatoriamente a qualquer EB presente no sistema, através da distribuição discreta *discrete(1,m)* onde *m* é o número de EBs presentes no sistema. Neste ponto, o estado do dispositivo é atualizado para conectado no lugar *Regs*, que armazena os registros com os DMs, seus estados, e EBs onde estão conectados.

Uma vez no estado conectado, o dispositivo pode se des-conectar, evento representado pelo disparo da transição *Disconnect*. O dispositivo pode ainda se movimentar e disparar o processo de *handoff*, evento representado pelo disparo da transição *Start Handoff*. Quando isto ocorre, o estado do dispositivo é atualizado para *handoff* e é enviado um conjunto de notificações a todas as EBs vizinhas da EB à qual o dispositivo está localizado. Desta forma essas vizinhas tentam se conectar ao dispositivo. No lugar *Notification* é guardada uma estrutura com a EB na qual o DM está atualmente conectado, mais a lista de todas as EBs que estão tentando contato com o DM.

Neste ponto o dispositivo pode realizar duas transições: se desconectar, através do disparo da transição *DiscHO*; ou se conectar a um vizinho, através do disparo da transição *Connect2*. Ao se conectar a um vizinho o registro é atualizado para refletir o fato de que o dispositivo está agora conectado a duas EBs ao mesmo tempo. Além disso, a lista de EBs com pedidos de *handoff* ativos para o DM também é alterada e passa a ser a intersecção de vizinhos dessas duas estações base, que mantém ativos seus pedidos de *handoff* para o dispositivo em questão.

Quando o dispositivo está conectado a duas estações base, três situações podem ocorrer. Assim como nos outro casos, o dispositivo pode se desconectar, através do disparo da transição *DiscFinal*. O dispositivo pode ainda finalizar o *handoff*, ou seja, ficar com apenas uma conexão ativa, aquela com a EB mais próxima, fato representado pelo disparo da transição *FinishHO*. Neste caso, o registro é atualizado removendo da lista de estações base a anterior e ficando apenas a atual, o estado é atualizado para conectado, e a lista de EBs com pedidos de *handoff* ativos para o DM é esvaziada. O último caso acontece quando o dispositivo continua se movendo, mas não em direção exata à vizinha. Neste caso, pode ocorrer de o dispositivo não finalizar o *handoff*, sair da área de cobertura da primeira e entrar na área de cobertura de uma terceira estação, que está na intersecção das duas primeiras. Este fato é representado pelo disparo da transição *Keep Walking*, que atualiza o registro do dispositivo para representar o fato que mudou a lista de estações base as quais o dispositivo está conectado e, conseqüentemente, a lista de notificações deve ser atualizada para a nova intersecção de vizinhos.

É importante notar que o modelo reflete as situações de que o dispositivo pode se desconectar a qualquer momento do sistema e que pode ficar indefinidamente em *handoff* sem

nunca finalizar um *handoff*. Este último caso reflete a possibilidade geral de o dispositivo se movimentar aleatoriamente na rede, nas áreas de sombra das estações base. Apesar de não termos dados estatísticos da ocorrência de determinadas situações em redes sem fio, o objetivo do protocolo é ser genérico e prover conexão em qualquer circunstâncias de movimentação, exceto nos casos em que o próprio dispositivo se desconecta da rede ou em que não existem mais vagas disponíveis para conexão na *piconet* de uma determinada estação base.

Na próxima seção são apresentados os cenários e resultados da análise usando o modelo de redes de Petri apresentado nesta seção e ilustrado na Figura 4.7.

4.2.2 Validação

A validação do modelo apresentado na Seção 4.2.1, e conseqüentemente do protocolo representado por ele, aconteceu em duas etapas: geração e análise de diagramas de seqüencia de mensagens, MSCs [5], para diversos cenários de simulação; e verificação formal do modelo por propriedades de interesse.

Cada MSC é gerado a partir de um cenário de simulação do modelo CPN. Cada transição no modelo CPN é convertido em uma seqüencia de mensagens nos MSCs. O cenário de simulação escolhido para exemplificar a análise do modelo CPN via MSC é o mesmo cenário ilustrado na Seção 3.3. Neste cenário o dispositivo *Dev* conecta-se inicialmente à EB BS_D. O dispositivo move-se então em direção a BS_E, iniciando o processo de *handoff*, e continua movendo-se em direção a BS_C mas não em linha reta, mas pela borda da área de cobertura de BS_E. O cenário da simulação encerra-se quando o *Dev* aproxima-se de BS_C e assume o estado CONECTADO nesta EB. Na Figura 4.8 é mostrado o MSC gerado para este cenário a partir do modelo da Seção 4.2.1.

Primeiro, *Dev* conecta-se ao sistema na EB BS_D, evento ilustrado no MSC pela mensagem *Connect* enviada de *Dev* para BS_D. Após a conexão de *Dev*, BS_D notifica seus vizinhos BS_A, BS_B, BS_E, BS_G, e BS_H sobre seu contato com *Dev* e que este se encontra no estado CONECTADO. No MSC essas mensagens são representadas pelas mensagens *Dev connected*. Todas essas mensagens no MSC são resultado do disparo da transição *Connect* no modelo ilustrado na Figura 4.7. *Dev* então começa a se mover pela área de cobertura de

[5] *Message Sequence Charts*

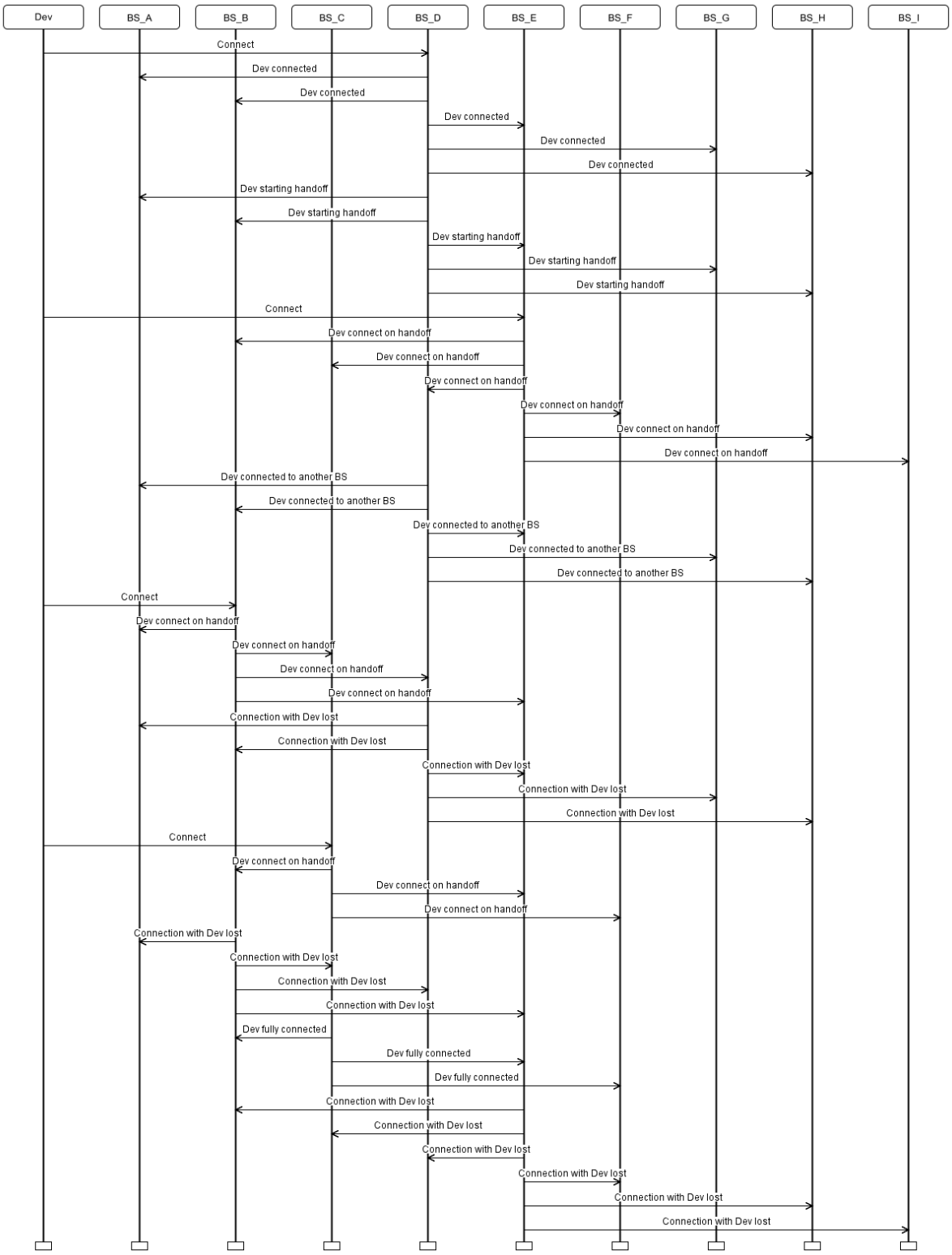


Figura 4.8: Diagrama de seqüencia para exemplo ilustrado na Figura 3.8.

BS_D, até o momento em que atravessa o ponto limiar para o disparo do *handoff*. Neste ponto, BS_D notifica suas vizinhas sobre a mudança de estado de *Dev*, evento representado no MSC pelas mensagens *Dev starting handoff*. Estas mensagens são geradas quando a transição *Start Handoff* é disparada no modelo. Como *Dev* se move em direção a BS_E, esta é a primeira EB a conseguir contato com *Dev*. Uma conexão entre *Dev* e BS_E é estabelecida, evento representado pela mensagem *Connect* no MSC. Após o estabelecimento desta conexão, BS_E envia uma notificação para todas as suas vizinhas, avisando do seu contato com *Dev* no estado SOB_HANDOFF (mensagens *Dev connected on handoff* no MSC). Neste momento, *Dev* está conectado ao mesmo tempo com BS_D e BS_E.

Quando BS_D recebe a notificação de BS_E ele a repassa às suas vizinhas, evento representado no MSC pelas mensagens *Dev connected to another BS*. Esta mensagem serve para que EBs que não são vizinhas de BS_E possam cancelar seus pedidos de *handoff* para *Dev*. Neste momento, apenas as EBs que são vizinhas tanto de BS_D quanto de BS_E, no caso BS_B e BS_H, continuam com pedidos de *handoff* ativos para *Dev*. No modelo, todas essas mensagens são geradas junto com o disparo da transição *Connect2* do modelo.

Neste momento, *Dev* começa a mover-se pela área de sombra de BS_E. Em sua trajetória, *Dev* entra na intersecção entre as áreas de cobertura de BS_B e BS_E, estabelecendo contato com BS_B. Este evento, e a notificação de BS_B sobre seu contato com *Dev*, são indicados pelas mensagens *Connect* e *Dev connected on handoff* no MSC. Adicionalmente, o contato entre *Dev* e BS_D é perdido. Quando isso acontece, BS_D notifica suas vizinhas sobre a perda de contato com *Dev*, evento representado pelas mensagens *Connection with Dev lost* no MSC. Estes eventos são gerados pelo modelo CPN quando a transição *KeepWalking* é disparada. Esta transição será disparada mais uma vez, indicando que *Dev* se moveu da intersecção das áreas de cobertura de BS_B e BS_E para a intersecção das áreas de cobertura de BS_C e BS_E.

Finalmente, o cenário de simulação encerra-se com o dispositivo totalmente conectado à EB BS_C. Primeiro, baseado na aproximação de *Dev*, BS_C notifica seus vizinhos sobre a mudança de estado de *Dev* para CONECTADO enviando as mensagens *Dev fully connected* no MSC. Em seguida, BS_E notifica seus vizinhos sobre a perda de contato com *Dev*. O cenário acaba com o disparo da transição *FinishHO*.

Como dito anteriormente, o MSC mostrado na Figura 4.8 foi gerado a partir do modelo

CPN ilustrado na Figura 4.7. A geração de MSCs acontece de forma automática junto com a simulação do modelo usando a biblioteca apropriada do CPN Tools. As mensagens são automaticamente geradas baseado nos valores das fichas nos lugares de entrada das transições. Foram gerados diversos MSCs, com diferentes tamanhos e disposições de EBs, e em todos os casos o comportamento registrado nos MSCs estava de acordo com o comportamento esperado pela definição do protocolo.

Além da geração e análise dos MSCs, também foram feitas verificações formais sobre o modelo em busca de certas propriedades. Mais especificamente, foram adicionadas algumas transições adicionais para que dispositivos, após perderem contato com as EBs do sistema, pudessem se conectar novamente. Após essa alteração, verificou-se que o modelo não possui marcações mortas e que é sempre possível retornar à marcação inicial. Em outras palavras, um dispositivo pode sempre conectar-se e desconectar-se das EBs do domínio de mobilidade sem interrupções em suas conexões com as EBs devido problemas decorrentes do protocolo de monitoração de DMs. Obviamente, problemas de descontinuidade nas conexões de DMs podem acontecer em conseqüência de falhas no módulo para detecção de mudanças de estado dos DMs ou ainda devido à falta de recursos nas EBs de destino.

Capítulo 5

Considerações Finais

Neste capítulo são feitas as considerações finais sobre o trabalho apresentado. São apresentadas discussões sobre os resultados obtidos, assim como uma discussão relativa a possíveis temas futuros de pesquisa, para dar continuidade ao trabalho apresentado ao longo desse documento.

5.1 Conclusões

Neste trabalho foi apresentado um protocolo para gerência de *handoffs* em WPANs Bluetooth no contexto de aplicações de tempo-real. A solução introduzida contempla tanto a alocação de responsabilidades quanto configurações relacionados à transição direta de DMs entre EBs. Adicionalmente, também foi definido o protocolo pelo qual EBs trocam informações para manter o DM conectado à medida que este se move através de suas áreas de cobertura.

O protocolo apresentado é voltado para o uso com aplicações que demandam transferências de dados em tempo-real, sendo sua adequação para esse tipo de aplicação demonstrada através de experimentos com uma aplicação de *streaming* de áudio. Os resultados descritos no Capítulo 4 demonstram a viabilidade e encorajam o desenvolvimento de um número de aplicações sobre o protocolo introduzido, tais quais as descritas no Apêndice B. Considerando que as taxas de transmissão dos *streams* usados para os testes naquele capítulo são razoavelmente altas, em torno de 320 Kbps, resultados ainda melhores devem ser obtidos quando *streams* com taxas de transmissão mais realistas sejam usados. Por exemplo, conteúdo de áudio é comumente codificado como MP3 numa taxa de 128 Kbps. Em

aplicações de telefonia pela Internet, por exemplo, CODECs modernos, como G.729A [73], conseguem codificar dados para transmissões de até 8 Kbps mantendo-se a mesma qualidade de CODECs com requisitos de banda maiores [73]. Larguras de banda entre 64 e 384 Kbps são comuns e suficientes para *streaming* de vídeo para dispositivos móveis usando MPEG4 [76].

O protocolo apresentado foi projetado focando-se as limitações dos potenciais dispositivos clientes, ou seja, pequenos dispositivos móveis com pouco poder de processamento, pouca memória, e largura de banda restrita. Dessa forma, todas as atividades relativas à monitoração de DMs e gerência de *handoffs* foram transferidas para as EBs. O protocolo apresentado descarta ainda a necessidade de coordenação, sinalizações ou quaisquer outras trocas de mensagens entre DMs e EBs durante os *handoffs*. O único requisito sobre a participação do DM em todo o processo é que este mantenha-se "conectável" enquanto move-se pelo domínio de mobilidade. Opcionalmente, DMs podem ainda ajustar os parâmetros *page scan window* e *page scan interval*. Com ajustes adequados desses valores reduz-se o número de *slots* da EB de destino necessários para a conexão com o DM sob *handoff*, conseguindo portanto um contato mais rápido com o DM. Com um gasto menor de slots reduz-se ainda o impacto do *paging* sobre outros DMs conectados na EB de destino. Por utilizar apenas operações padronizadas do Bluetooth, viabiliza-se o uso do protocolo proposto junto com qualquer dispositivo programável equipado com interface Bluetooth de acordo com a especificação, sendo portanto dispensada a necessidade de, por exemplo, alterações em camadas inferiores de *software*, como a pilha Bluetooth.

Aplicações de tempo-real, particularmente aquelas que envolvem transmissão de dados via rede, possuem fortes requisitos quanto à transmissão/recepção de suas informações. Gerenciar *handoffs* levando-se em conta esse tipo de aplicação significa buscar os meios pelos quais conexões podem ser estabelecidas e desfeitas com o mínimo de impacto sobre a disponibilidade do canal de comunicação. A solução aqui introduzida descarta a operação de *inquiry* durante os *handoffs* e explora a capacidade de Bluetooth de múltiplas conexões simultâneas para gerenciar *soft handoffs*, evitando assim a perda total de conectividade do DM enquanto este se move através das áreas de coberturas de diferentes EBs.

O trabalho aqui apresentado discute ainda outras facetas do relacionamento entre aplicações de tempo-real e gerenciamento de *handoffs* em Bluetooth. Por exemplo,

compreendeu-se o relacionamento entre aplicações de tempo-real e os constantes chaveamentos da interface para escutar por tentativas de *pagings* e *inquiries*, assim como os procedimentos de disparo e resposta dessas operações. Vimos que o simples ajuste dos parâmetros do *paging* é suficiente para reduzir o tempo de conexão para algo em torno de 10 ms (16 *slots*). Entretanto, temos que na prática dificilmente esse desempenho será alcançado levando-se em conta o tráfego de dados em tempo-real pelas interfaces Bluetooth devido ao impacto dessa configuração na disponibilidade do canal.

Os resultados obtidos com a interface CSR avaliada mostram que interfaces de menor qualidade podem não ser adequadas para aplicações de tempo-real mais exigentes, como telefonia. Entretanto, essas interfaces podem ser utilizadas em aplicações similares às tradicionais aplicações de *streaming*, onde *buffers* podem ser empregados pois um atraso de alguns segundos na reprodução do *stream*, ou eventuais paradas para recarregar o *buffer*, não comprometem a aplicação. Neste caso, *buffers* ocultariam das aplicações finais a queda na largura de banda disponível à aplicação durante o *handoff*.

Infelizmente, nenhum dos trabalhos relacionados conhecidos até o momento da escrita desse trabalho apresentam dados suficientes sobre sua adequação ou não para o funcionamento com aplicações de tempo-real [13; 30; 20; 11; 35; 10; 31; 3; 70]. Por isso, não é possível fazer uma análise comparativa entre os resultados aqui obtidos e os resultados de outros trabalhos.

Por fim, a modelagem e análise serviu para validar o protocolo através de simulações, mostrando que o protocolo se comporta de acordo com o comportamento especificado na Seção 3.3 para vários cenários. Através da verificação formal do modelo criado provaram-se propriedades importantes, a principal delas é que o protocolo cumpre seu papel de manter DMs que se movem pelo domínio de mobilidade conectados a pelo menos uma EB.

A solução apresentada delega para o Link Monitor a tarefa de implementar as heurísticas necessárias para a localização de dispositivos. Devido essa delegação de responsabilidade, a solução de gerenciamento de *handoffs* apresentada pode ser afetada por más implementações do Link Monitor. Ou seja, informações pouco precisas, ou tardias, sobre a localização de dispositivos podem causar desde disparos de procedimentos de *handoff* não necessários, à total perda de conectividade de DMs em movimento devido ao não envio de mudanças no seu estado por parte de sua EB de origem.

5.2 Trabalhos Futuros

Como visto ao longo deste documento, a questão da gerência de *handoffs* é um problema amplo. Para a condução deste trabalho uma série de suposições precisaram ser feitas, assim como uma cuidadosa delimitação do problema a ser resolvido. Nesse contexto alguns tópicos para trabalhos futuros são discutidos a seguir.

5.2.1 Definição de uma Solução Escalável

Como brevemente discutido na Seção 3.4, e mais detalhadamente no Apêndice C, escalabilidade é uma questão natural num sistema com recursos limitados e onde o número de usuários pode crescer rapidamente. Assim, faz-se necessário o desenvolvimento do conhecimento sobre como aumentar a disponibilidade de recursos disponíveis no sistema, viabilizando um maior número de clientes atendidos de forma simultânea. Ao mesmo tempo, aumenta-se também a disponibilidade de interfaces para atender um número maior de *handoffs* simultâneos.

Outra necessidade intrinsicamente ligada à questão de escalabilidade consiste no desenvolvimento de mecanismos para controle de admissão nas EBs. Através da redução do número de *handoffs* simultâneos que precisam ser gerenciados por uma EB, soluções apropriadas para controle de admissão podem evitar tanto o desperdício de recursos das EBs quanto oscilações na qualidade do serviço já em uso por outros DMs. Por exemplo, caso um DM em *handoff* para uma determinada EB consuma mais recursos do que os atualmente disponíveis na interface onde ele será conectado na EB de destino, seu pedido de *handoff* além de consumir tempo para ser processado, prejudicando outros DMs também em *handoff* para aquela EB, tende a congestionar o enlace sem fio da EB após a admissão do DM, prejudicando o tráfego de dados de todos os demais clientes ligados à mesma interface. A idéia então é que dispositivos movendo-se em direção a uma EB que não tem recursos suficientes para recebê-lo devem ter seus pedidos de *handoff* ignorados, em benefício de outros DMs sob *handoff* para aquela mesma EB.

Nesse contexto, experiências, sucessos, e problemas já conhecidos no trabalho descrito no Apêndice C, podem ser usados como base para a evolução da atual proposta rumo a um sistema escalável.

5.2.2 Integração com Mecanismos de Mobilidade de Aplicações

O trabalho aqui apresentado contemplou o processo para a troca de conexão física entre EBs, que é apenas o começo de um problema maior e que envolve várias camadas de *software*. Por exemplo, ao mudar de EB, DMs podem ganhar novo endereço IP, exigindo uma atualização nas rotas entre o DM e outros computadores com conexões abertas com aquele dispositivo. No caso de transferência de dados em tempo-real, por exemplo, em algum momento o tráfego precisa ser redirecionado de uma EB para outra, assim como dados em trânsito (dados que foram enviados por algum servidor antes deste ter sido notificado sobre a mudança de estado do DM) precisam ser, de alguma forma, roteados corretamente para o DM após sua mudança de ponto de acesso. Tudo isso precisa acontecer de forma rápida, de forma a não prejudicar o desempenho das aplicações em curso. De outra maneira, o *handoff* torna-se perceptível para o usuário, comprometendo o funcionamento de aplicações (por exemplo, aplicações com fortes requisitos temporais, como áudio e vídeo) e protocolos (TCP por exemplo, devido o uso de *backoffs* exponenciais para lidar com congestionamentos na rede).

A pesquisa em torno de mobilidade de aplicações já têm longa data e diversos resultados consolidados, como IP Móvel [48] ou Celular IP [8]. Nesse contexto, um ponto a ser investigado seria a expansão da solução de *handoff* já existente em direção a camadas mais altas da pilha de *software*. Pode-se investigar a integração com soluções já existentes para mobilidade de aplicações, como as duas citadas anteriormente, ou soluções próprias, voltadas para aplicações e cenários específicos, podem ser desenvolvidas e acopladas ao trabalho atual.

5.2.3 Localização e Monitoração dos Estados de Dispositivos

Outro ponto de trabalho futuro consiste em extender o componente para localização de dispositivos implementado para os testes tendo em vista soluções já existentes para localização e monitoração de dispositivos Bluetooth. O objetivo é conhecer as vantagens e problemas de cada alternativa, buscando aquela que melhor se encaixe nas necessidades das aplicações/ambientes alvo. A exemplo do trabalho apresentado por Shantidev Mohanty et al. em [42], a solução adotada deve levar em consideração aspectos com velocidade média dos DMs, atrasos envolvidos na detecção e sinalização de *handoffs*, e outros parâmetros do sistema, conforme discutido na Seção 5.2.4.

5.2.4 Métodos e Modelos para Calibração do Sistema

Por fim, o último tópico sugerido para investigação futura consiste em definir procedimentos e modelos para calibração do sistema. Ou seja, existem parâmetros que podem melhorar o funcionamento do sistema apresentado anteriormente, como reduzir o tempo dos *pagings* ou a probabilidade de *handoffs* mau sucedidos. Como visto na Seção 2.5.3, o tempo da ope-ração de *paging* pode ser reduzido ou aumentado conforme ajustes nos parâmetros *page scan window* e *page scan interval*. Nesse contexto, pode-se buscar modelos que relacionem o comportamento da interface Bluetooth, junto com os parâmetros do *paging*, mais as características da aplicação, de forma a sugerir valores ótimos, ou próximos disso, para a operação de *paging*, sem prejudicar o desempenho do fluxo de dados.

Além do ajuste dos parâmetros do *paging*, outro atributo para calibração do sistema consiste na busca de heurísticas e/ou metodologias para definir um posicionamento ótimo das EBs, assim como do ponto limiar para disparo do *handoff*. Se, por um lado, menos EBs são necessárias para cobrir uma mesma área contínua desde que estas estejam suficientemente distantes umas das outras, por outro esse posicionamento reduz a "área de sombra" entre as EBs, o que aumenta os requisitos sobre o módulo de localização e monitoração de DMs, que agora tem menos tempo de atraso tolerado para reportar a mudança de estado do DM. O caso inverso também possui perdas e ganhos. Ao posicionar EBs muito próximas umas às outras aumenta-se a área de sombra entre as EBs, o que diminui as exigências sobre o módulo de monitoração e localização de DMs mas, em compensação, aumenta-se a quantidade de *handoffs* que as EBs precisam atender, além de diminuir a quantidade de recursos no sistema, pois serão comuns as situações onde DMs não necessariamente estão se movendo entre EBs, mas estão conectados a mais de uma ao mesmo tempo, desperdiçando recursos.

Bibliografia

[1] Vivek Arora, Larry H. Chang, Seyhan Civanlar, Vikram R. Saksena, and Agnes C. Tow. Method and apparatus for dynamically forming multimedia emulated local area networks. http://www.google.com/patents/pdf/Method_and_apparatus_for_dynamically_for.pdf, 2007. Acessado em Setembro 2007.

[2] IEEE Standards Association. IEEE 802.15 WPAN Task Group 1 (TG1). http://www.ieee802.org/15/pub/TG1.html, 2007. Acessado em Setembro 2007.

[3] S. Baatz, M. Frank, R. Gopffarth, D. Kassatkine, P. Martini, M. Schetelig, and A. Vilavaara. Handoff Support for Mobility with IP over Bluetooth. In *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks (LCN'00)*, páginas 143–154. IEEE Computer Society, 2000.

[4] N. Banerjee, W. Wu, K. Basu, and S. Das. Analysis of SIP-based Mobility Management in 4G Wireless Networks. *Computer Communications*, 27(8):697–707, 2004.

[5] Paolo Bellavista, Antonio Corradi, and Carlo Giannelli. Adaptive Buffering-Based on Handoff Prediction for Wireless Internet Continuous Services. In *Proceedings of the First International Conference on High Performance Computing and Communications (HPCC 2005)*, páginas 1021–1032, 2005.

[6] Timothy M. Bielawa. Position Location of Remote Bluetooth Devices. Dissertação de Mestrado, Virginia Polytechnic Institute and State University, Blacksburg, Estados Unidos, 2005.

[7] Jennifer Bray and Charles F. Sturman. *Bluetooth: Connect Without Cables*. Prentice Hall, 1 edição, 2000.

[8] A. Campbell, J. Gomez, C-Y. Wan, and S. Kim. Cellular IP. http://comet.columbia.edu/cellularip/pub/draft-ietf-mobileip-cellularip-00.txt, 1999. Acessado em Setembro 2007.

[9] Javier García Castaño. Algorithms and Protocols Enhancing Mobility Support for Wireless Sensor Networks Based on Bluetooth and Zigbee. Dissertação de Mestrado, Department of Computer Science and Electronics, Västeras, Suécia, 2006.

[10] Wah-Chun Chan, Jiann-Liang Chen, Po-Tsang Lin, and Ka-Chin Yen. Quality-of-Service in IP Services over Bluetooth Ad-Hoc Networks. *Mobile Networks and Applications*, 8(6):699–709, 2003.

[11] Ming-Chiao Chen, Jiann-Liang Chen, and Pei-Chun Yao. Efficient Handoff Algorithm for Bluetooth Networks. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics Systems: Networking and Services*, páginas 3884–3889. IEEE Computer Society, 2005.

[12] S. Chetan, J. Al-Muhtadi, R. Campbell, and M. D. Mickunas. Mobile Gaia: A Middleware for Ad-Hoc Pervasive Computing. In *Proceedings of the 2005 Consumer Communications and Networking Conference (CCNC 2005)*, páginas 223–228. IEEE Computer Society, 2005.

[13] S. Chung, H. Yoon, and J. Cho. A Fast Handoff Scheme For IP over Bluetooth. In *Proceedings of the 31th International Conference on Parallel Processing Workshops (ICPP 2002)*, páginas 51–55. IEEE Computer Society, 2002.

[14] Intel Corporation. Ultra-Wideband (UWB) Technology. http://www.intel.com/technology/comms/uwb/index.htm, 2007. Acessado em Setembro 2007.

[15] Antonio DeSimone, Joseph Golan, Ashok K. Kuthyar, Bryant Richard Parent, Ram S. Ramamurthy, and David Hilton Shur. Method for managing multicast addresses for transmitting and receiving multimedia conferencing. http://www.google.com/patents/pdf/Method_for_managing_multicast_addresses_.pdf, 2007. Acessado em Setembro 2007.

[16] Anind Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[17] Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, and Zighuo Lue. An Indoor Bluetooth-based Positioning System: Concept, Implementation and Experimental Evaluation. In *Proceedings of the International Conference on Wireless Networks (ICWN 2003)*, páginas 109–113. CSREA Press, 2003.

[18] F. Forno, G. Malnati, and G. Portelli. Design and Implementation of a Bluetooth Ad-Hoc Network for Indoor Positioning. *IEE PROCEEDINGS SOFTWARE*, 152(5):223–228, 2005.

[19] Alessandro Genco, Salvatore Sorce, and Giuseppe Scelfo. Bluetooth Base Station Minimal Deployment for High Definition Positioning. In *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2005)*, páginas 454–460. IEEE Computer Society, 2005.

[20] M. L. George, L. J. Kallidukil, and J. M. Chung. Bluetooth handover control for roaming system applications. In *Proceedings of the 45th Midwest Symposium on Circuits and Systems (MWSCAS 2002)*, páginas I404–I407. IEEE Computer Society, 2002.

[21] GStreamer. GStreamer - Open Source Multimedia Framework. http://gstreamer.freedesktop.org/, 2007. Acessado em Setembro 2007.

[22] Timo Halonen, Javier Romero, and Juan Melero. *GSM, GPRS and EDGE Performance: Evolution Towards 3G/UMTS*. John Wiley & Sons, 1 edição, 2003.

[23] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, Internet Engineering Task Force, 1998.

[24] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The Gator Tech Smart House: A Programmable Pervasive Space. *Computer*, 38(3):50–60, 2005.

[25] Sumi Helal. Programming Pervasive Spaces. *IEEE Pervasive Computing*, 4(1):84–87, 2005.

[26] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla. Personal Area Networks: Bluetooth or IEEE 802.11? *International Journal of Wireless Information Networks*, 9(2):89–103, 2002.

[27] P. Johansson, M. Kazantzidis, R. Kapoor, and M. Gerla. Bluetooth: An Enabler for Personal Area Networking. *IEEE Network*, 15(5):28–37, 2001.

[28] Val Jones, Richard Bults, Dimitri Konstantas, and Pieter AM Vierhout. Healthcare pans: Personal area networks for trauma care and home care. In *Proceedings of the 4th International Symposium on Wireless Personal Multimedia Software Composition (WPMC 2001)*, páginas 1369–1374, 2001.

[29] E. Jovanov, D. Raskovic, J. Price, A. Moore, J. Chapman, and A. Krishnamurthy. Patient monitoring using personal area networks of wireless intelligent sensors. In *Proceedings of the 38th Annual Rocky Mountain Bioengineering Symposium*, páginas 373–378, 2001.

[30] Aman Kansal. A Handoff Protocol for Mobility in Bluetooth Public Access. Dissertação de Mestrado, Department of Electrical Engineering, Mumbai, Índia, 2002.

[31] Aman Kansal and UB Desai. Handoff Protocol for Bluetooth Public Access. In *Proceedings of the 2002 IEEE International Conference on Personal Wireless Communicatons (ICPWC'02)*, páginas 159–162. IEEE Computer Society, 2002.

[32] E. Kirda, C. Kerer, M. Jazayeri, and C. Kruegel. Supporting Multi-Device Enabled Web Services: Challenges and Open Problems. In *Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2001)*, páginas 49–54. IEEE Computer Society, 2001.

[33] Chul-Seung Lee, Dong-Jun Cho, Young-Hwan You, and Hyoung-Kyu Song. A Solution to Improvement of DS-UWB System in the Wireless Home Entertainment Network. *IEEE Transactions on Consumer Electronics*, 51(2):529–533, 2005.

[34] Jack Y. B. Lee. *Scalable Continuous Media Streaming Systems: Architecture, Design, Analysis and Implementation*. John Wiley & Sons, Ltd, 1 edição, 2005.

[35] Won Hee Lee, Yang-Ick Joo, Kyun Hyon Tchah, Yongsuk Kim, and DooSeop Eom. Handoff Provisioning in Bluetooth Wireless Personal Area Networks. *IEEE Transactions on Consumer Electronics*, 49(4):1004–1012, 2003.

[36] M. Li, Y. Fei, V. Leung, and T. Randhawa. A New Method to Support UMTS/WLAN Vertical Handover Using SCTP. *IEEE Wireless Communications*, 11(4):44–51, 2004.

[37] Yi-Bing Lin and Ai-Chun Pang. Comparing Soft and Hard Handoffs. *IEEE Transactions on Vehicular Technology*, 49(3):792–798, 2000.

[38] Jane W.C. Liu. *Real-Time Systems*. Prentice Hall, 1 edição, 2000.

[39] Michael R. Macedonia and Donald P. Brutzman. Mbone provides audio and video across the internet. *Computer*, 27(4):30–36, 1994.

[40] J. McNair and Z. Fang. Vertical Handoffs in Fourth Generation Multinetwork Environments. *IEEE Wireless Communications*, 11(3):8–15, 2004.

[41] Live Media. Live555 Streaming Media. http://www.live555.com/liveMedia/, 2007. Acessado em Setembro 2007.

[42] Shantidev Mohanty and Ian F. Akyildiz. A Cross-Layer (Layer 2 + 3) Handoff Management Protocol for Next-Generation Wireless Systems. *IEEE Transactions on Mobile Computing*, 5(10):1347–1360, 2006.

[43] Loreno Oliveira, Emerson Loureiro, Hyggo Almeida, and Angelo Perkusich. *Encyclopedia of Mobile Computing and Commerce*, volume 2, chapter Bridging together Mobile and Service-Oriented Computing, páginas 1–7. Idea Group Inc., 2006.

[44] Loreno Oliveira, Leandro Sales, Emerson Loureiro, Hyggo Almeida, and Angelo Perkusich. Filling the Gap Between Mobile and Service-oriented Computing: Issues for Evolving Mobile Computing Towards Wired Infrastructures and Vice Versa. *International Journal on Web and Grid Services*, 2(4):355–378, 2006.

[45] MIT Project Oxygen. MIT Project Oxygen - Pervasive, Human-centered Computing. http://oxygen.csail.mit.edu/, 2007. Acessado em Setembro 2007.

[46] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *ACM SIGCOMM Computer Communication Review*, 28(4):303–314, 1998.

[47] C. Perkins. Mobile Networking in the Internet. *Mobile Networks and Applications*, 3(4):319–334, 1998.

[48] C. Perkins. IP Mobility Support. http://www.ietf.org/rfc/rfc2002.txt, 2002. Acessado em Setembro 2007.

[49] Ramjee Prasad and Liljana Gavrilovska. Research Challenges for Wireless Personal Area Networks. In *Proceedings of the 3rd International Conference on Information, Communications and Signal Processing (ICICS 2001)*, 2001.

[50] Ramjee Prasad and Luis Munoz. *WLANs and WPANs towards 4G Wireless*. Artech House Publishers, 1 edição, 2003.

[51] BlueZ Project. BlueZ - Official Linux Bluetooth Protocol Stack. http://www.bluez.org/, 2007. Acessado em Setembro 2007.

[52] Ishwar Ramani and Stefan Savage. SyncScan: Practical Fast Handof for 802.11 Infrastructure Networks. In *Proceedings of the 24th Anual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2005)*, páginas 675–684. IEEE Computer Society, 2005.

[53] F. Ramparany, O. Boissier, and H. Brouchoud. Cooperating Autonomous Smart Devices. In *Proceedings of the Smart Objects Conference (sOc'03)*, páginas 182–185, 2003.

[54] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. RFC 3261 - SIP: Session Initiation Protocol. http://www.faqs.org/rfcs/rfc3261.html, 2002. Acessado em Setembro 2007.

[55] McCann S., Groting W., Pandolfi A., and Hepworth E. Next Generation Multimode Terminals. http://www.roke.co.uk/download/papers/next_generation_multimode_terminals.pdf, 2007. Acessado em Setembro 2007.

[56] D. Saha, A. Mukherjee, I. S. Misra, M. Chakraborty, and N. Subhash. Mobility support in IP: a survey of related protocols. *IEEE Network*, 18(6):34–40, 2004.

[57] Debashis Saha and Amitava Mukherjee. Pervasive Computing: A Paradigm for the 21st Century. *Computer*, 36(3):25–31, 2003.

[58] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4):10–17, 2001.

[59] Albrecht Schmidt, Martin Strohbach, Kristof van Laerhoven, Adrian Friday, and Hans-Werner Gellersen. Context Acquisition Based on Load Sensing. In *Proceedings of the Fourth International Conference on Ubiquitous Computing (UbiComp'02)*, páginas 333–350. Springer, 2002.

[60] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. http://rfc.dotsrc.org/rfc/rfc1889.html, 1996. Acessado em Setembro 2007.

[61] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). ftp://ftp.isi.edu/in-notes/rfc2326.txt, 1998. Acessado em Setembro 2007.

[62] Henning Schulzrinne and Elin Wedlund. Application-layer mobility using SIP. *Mobile Computing and Communications Review*, 1(2):47–57, 2000.

[63] F. Siddiqui and S. Zeadally. Mobility Management across Hybrid Wireless Networks: Trends and Challenges. *Computer Communications*, 29(9):1363–1385, 2005.

[64] Bluetooth SIG. Bluetooth Specification Version 2.0 + EDR. http://www.bluetooth.com/NR/rdonlyres/1F6469BA-6AE7-42B6-B5A1-65148B9DB238/840/Core_v210_EDR.zip, 2007. Acessado em Setembro 2007.

[65] Bluetooth SIG. Compare Bluetooth with Other Technologies. http://www.bluetooth.com/Bluetooth/Learn/Technology/Compare/, 2007. Acessado em Setembro 2007.

[66] Alex C. Snoeren and Hari Balakrishnan. An End-to-End Approach to Host Mobility. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, páginas 155–166. ACM Press, 2000.

[67] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. http://www.ietf.org/rfc/rfc2960.txt, 2000. Acessado em Setembro 2007.

[68] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley Professional, 3 edição, 2000.

[69] Jean Tourriles. Bluetooth Roaming Proposals. Relatório Técnico, HP Laboratories, 2000.

[70] Jean Tourrilhes. On-Demand Bluetooth: Experience Integrating Bluetooth in Connection Diversity. Relatório Técnico HPL-2003-178, HP Laboratories Palo Alto, 2003.

[71] Kenneth Virgile. Network bridge with multicast forwarding table. http://www.google.com/patents/pdf/Network_bridge_with_multicast_forwarding.pdf, 2007. Acessado em Setembro 2007.

[72] Bernhard H. Walke, Stefan Mangold, and Lars Berlemann. *IEEE 802 Wireless Systems: Protocols, Multi-hop Mesh/Relaying, Performance and Spectrum Coexistence*. John Wiley & Sons, Ltd, 1 edição, 2007.

[73] Ted Wallingford. *Switching to VoIP*. O´Reilly Media, 1 edição, 2005.

[74] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):66–75, 1991.

[75] WiBree. WiBree. http://www.wibree.com/, 2007. Acessado em Setembro 2007.

[76] Stefan Winkler and Frédéric Dufaux. Video quality evaluation for mobile applications. In *Proceedings of the 2003 SPIE Visual Communications and Image Processing Conference*, páginas 593–603, 2003.

[77] WirelessUSB. Certified Wireless USB from the USB-IF. http://www.usb.org/developers/wusb/, 2007. Acessado em Setembro 2007.

[78] Qing-An Zeng and Dharma P. Agrawal. *Handbook of wireless networks and mobile computing*, chapter Handoff in wireless mobile networks, páginas 1–25. John Wiley & Sons, Inc., 2002.

[79] ZigBee. ZigBee: Wireless Control that Simply Works. http://www.zigbee.org, 2005. Acessado em Setembro 2007.

Apêndice A

Escolhendo Valores para os Parâmetros do *Paging*

De acordo com a especificação do *Bluetooth* [64], os parâmetros *page scan window* e *page scan interval* podem variar entre 17 e 4096 *slots*, e entre 18 e 4096 *slots*, respectivamente. Ainda de acordo com a especificação, estes valores podem ser definidos livremente dentro dessas faixas, contanto que o *page scan window* seja menor ou igual ao *page scan interval*. No escopo dos testes executados o objetivo não era encontrar valores ótimos para esses parâmetros. Como dito anteriormente na Seção 3.2.2, esta é uma busca mais complexa, que envolve diversas variáveis e que ainda pode ser influenciada por um componente aleatório, no caso, as condições do canal sem fio e as escolhas de pacotes de camada física feitas pela interface mediante essas condições. Ao invés disso, foi buscada apenas uma combinação de *page scan window* e *page scan interval* que reduzisse o tempo de *paging*, reduzindo portanto o número de *slots* necessários pela EB para estabelecer conexão com o DM. Além disso, o impacto sobre um fluxo de dados em tempo-real, causado por ajustes dos parâmetros do *paging*, pode ser ilustrado através dos testes feitos para a escolha dos valores desses parâmetros.

Dados os objetivos descritos anteriormente, e para reduzir a quantidade de valores testados para cada parâmetro, as combinações de valores avaliados para os parâmetros *page scan window* e *page scan interval* foram escolhidos de acordo com as regras por trás de seu funcionamento (Seção 2.5.3). Conforme descrito por Jennifer Bray e Charles Sturman em [7], é recomendado que o *page scan window* seja longo o suficiente para que sejam escutadas,

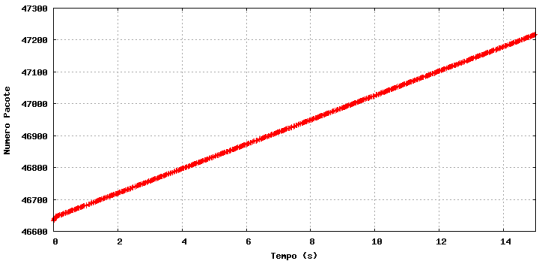
pelo menos, metade das 32 freqüências disponíveis. Assim, a estratégia usada para variar seu valor foi escolher valores múltiplos de 18. A estratégia para escolher os valores para o *page scan interval* consistiu apenas em, partindo do seu valor padrão, reduzir seu valor pela metade.

Nesse contexto, os valores do *page scan window* escolhidos para avaliação foram 18, 36, e 54. Estes valores foram combinados com os valores 1024, 512, 256, e 128, escolhidos para o *page scan interval*. Para os testes a seguir, assim como na Seção 4.1.2, foi utilizado um *stream* de um arquivo MP3 codificado a 320 Kbps. Interfaces Broadcom foram usadas tanto na EB quanto no DM. Durante esses testes não houve *handoff*. Em cada teste era alterada apenas a configuração dos parâmetros do *paging*. Ainda, a qualidade do *stream* recebido em cada teste foi verificada via avaliação subjetiva usando o *gstreamer*, da mesma maneira descrita na Seção 4.1.1.

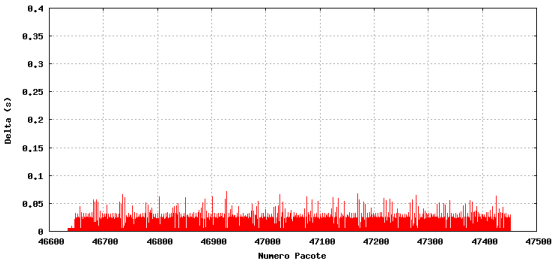
Na Figura A.1 são mostrados os gráficos da recepção de pacotes e intervalos na chegada entre pacotes para um *page scan window* de 18 *slots*. Nos testes com essa configuração de *page scan window*, apenas o experimento com *page scan interval* de 128 *slots* (Figuras A.1(g) e A.1(h)) falhou na avaliação subjetiva. Todos os demais apresentaram reproduções perfeitas. Mesmo assim, as imperfeições na reprodução do *stream* desse teste, reveladas pela avaliação subjetiva, foram muito pequenas e esporádicas.

Na Figura A.2 são mostrados os gráficos para os testes com ajuste do *page scan window* para 36 *slots*. Nos testes para esse novo valor, os experimentos com *page scan interval* de 1024 (Figuras A.2(a) e A.2(b)) mostraram uma reprodução suave do audio, sem falhas perceptíveis. Em compensação, os experimentos com *page scan interval* de 512 *slots* (Figuras A.2(c) e A.2(d)), 256 (Figuras A.2(e) e A.2(f)) e 128 *slots* (Figuras A.2(g) e A.2(h)) falharam. Enquanto as imperfeições ouvidas nos experimentos para os ajustes de 512 e 256 *slots* eram, assim como o experimento das Figuras A.1(g) e A.1(h), poucas e esporádicas, a avaliação subjetiva do *stream* com *page scan interval* de 128 *slots* revelou uma reprodução com diversos ruídos e curtas interrupções.

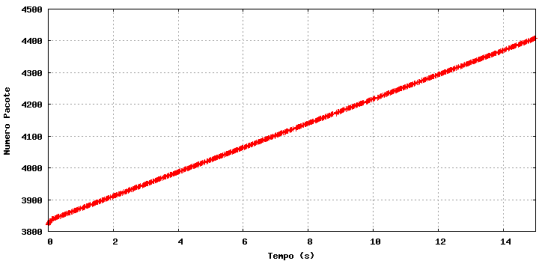
Por fim, na Figura A.3 são mostrados os gráficos obtidos a partir do ajuste do *page scan window* para 54 *slots*. Todos os experimentos para esse ajuste de *page scan window* mostraram falhas na reprodução do audio recebido. Enquanto o experimento com ajuste do *page scan interval* para 1024 (Figuras A.3(a) e A.3(b)) *slots* apresentou apenas pequenas



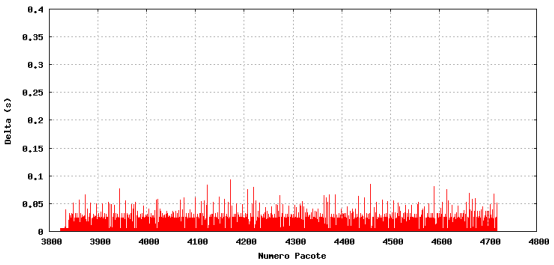
(a) Chegada de pacotes RTP para *page scan interval* = 1024 *slots*



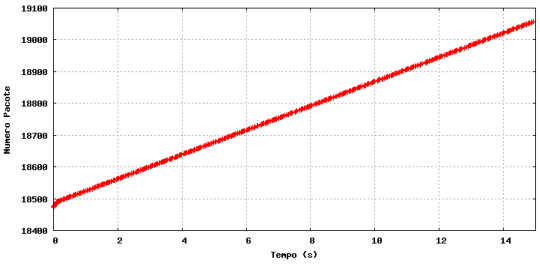
(b) Intervalos entre as chegadas *page scan interval* = 1024 *slots*



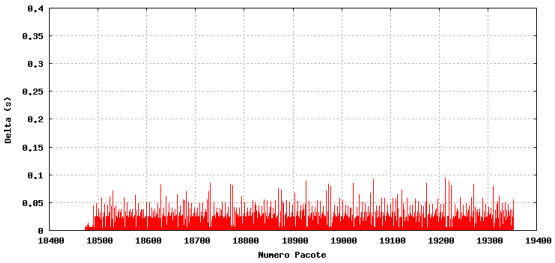
(c) Chegada de pacotes RTP para *page scan interval* = 512 *slots*



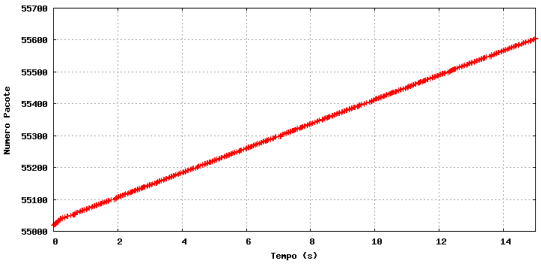
(d) Intervalos entre as chegadas para *page scan interval* = 512 *slots*



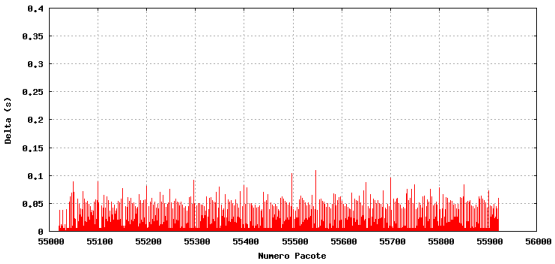
(e) Chegada de pacotes RTP para *page scan interval* = 256 *slots*



(f) Intervalos entre as chegadas para *page scan interval* = 256 *slots*

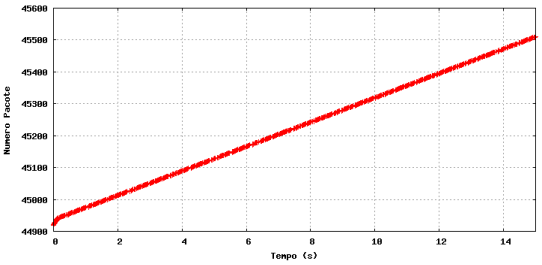


(g) Chegada de pacotes RTP para *page scan interval* = 128 *slots*

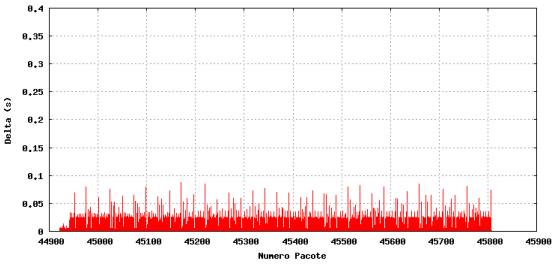


(h) Intervalos entre as chegadas para *page scan interval* = 128 *slots*

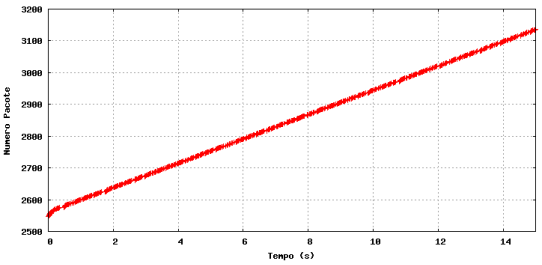
Figura A.1: Avaliações para *page scan window* = 18 *slots*



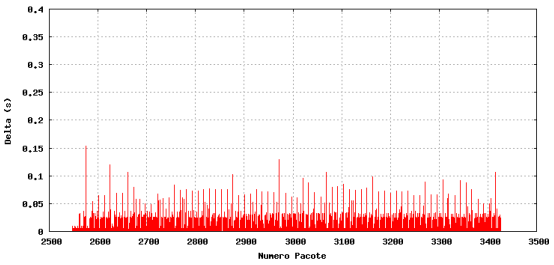
(a) Chegada de pacotes RTP para *page scan interval* = 1024 *slots*



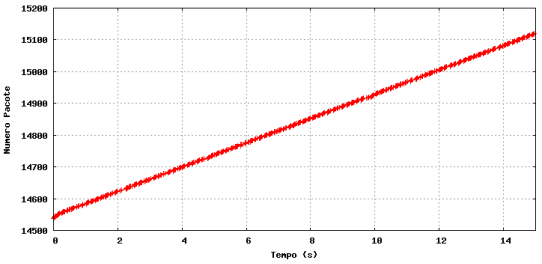
(b) Intervalos entre as chegadas para *page scan interval* = 1024 *slots*



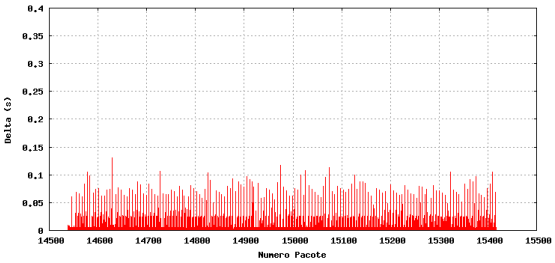
(c) Chegada de pacotes RTP para *page scan interval* = 512 *slots*



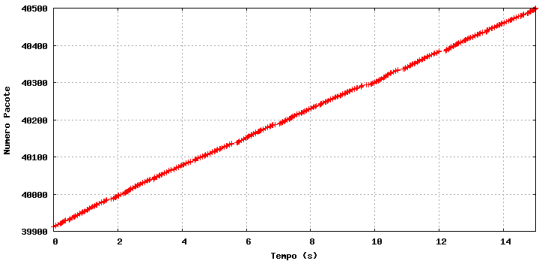
(d) Intervalos entre as chegadas para *page scan interval* = 512 *slots*



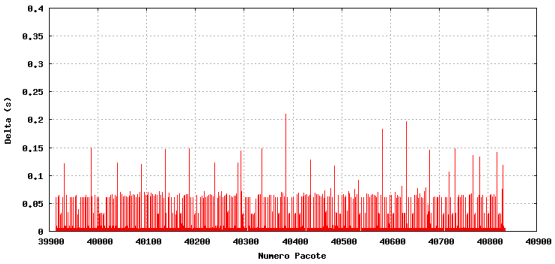
(e) Chegada de pacotes RTP para *page scan interval* = 256 *slots*



(f) Intervalos entre as chegadas para *page scan interval* = 256 *slots*

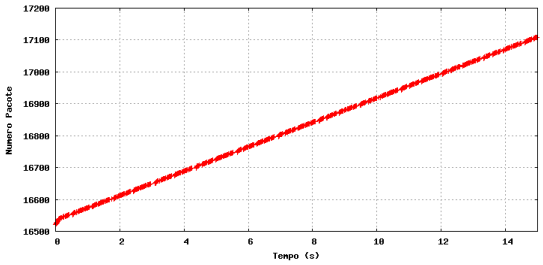


(g) Chegada de pacotes RTP para *page scan interval* = 128 *slots*

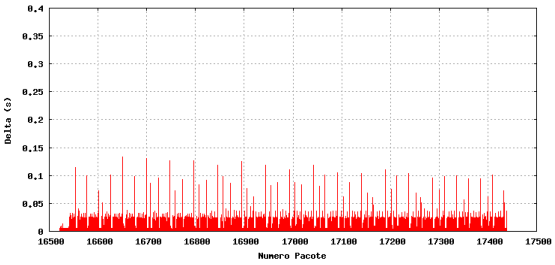


(h) Intervalos entre as chegadas para *page scan interval* = 128 *slots*

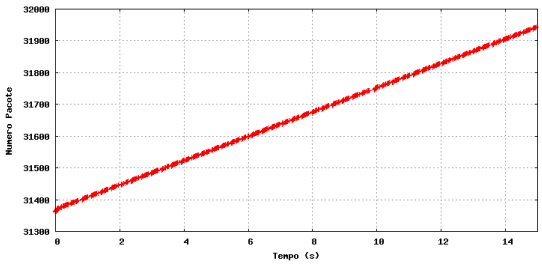
Figura A.2: Avaliações de valores para *page scan window* = 36 *slots*



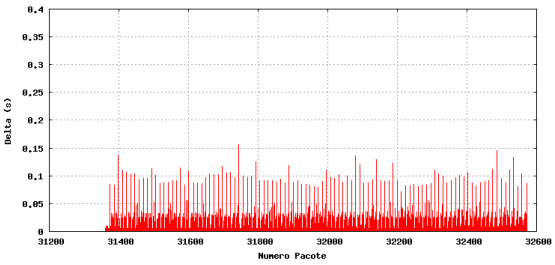
(a) Chegada de pacotes RTP para *page scan interval* = 1024 *slots*



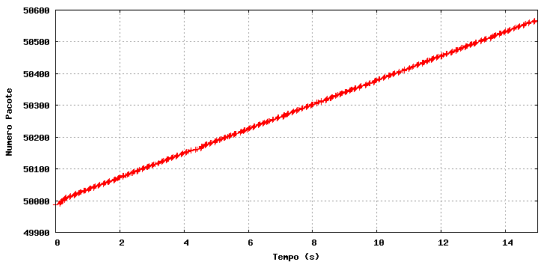
(b) Intervalos entre as chegadas para *page scan interval* = 1024 *slots*



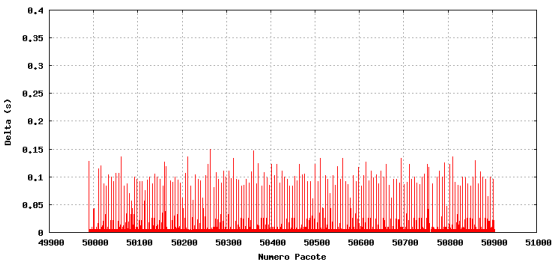
(c) Chegada de pacotes RTP para *page scan interval* = 512 *slots*



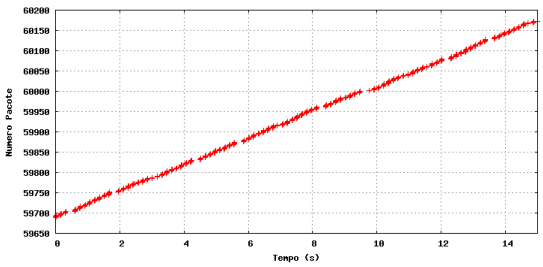
(d) Intervalos entre as chegadas para *page scan interval* = 512 *slots*



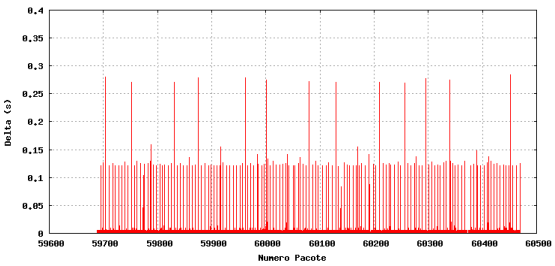
(e) Chegada de pacotes RTP para *page scan interval* = 256 *slots*



(f) Intervalos entre as chegadas para *page scan interval* = 256 *slots*



(g) Chegada de pacotes RTP para *page scan interval* = 128 *slots*



(h) Intervalos entre as chegadas para *page scan interval* = 128 *slots*

Figura A.3: Avaliações de valores para *page scan window* = 54 *slots*

falhas, os experimentos para os ajustes de 512 (Figuras A.3(c) e A.3(d)), 256 (Figuras A.3(e) e A.3(f)), e principalmente 128 *slots* (Figuras A.3(g) e A.3(h)), apresentaram muitos ruídos e interrupções durante a reprodução do *stream*.

Na Tabela A.1 são enumerados os resultados dos testes com os diferentes configurações avaliadas para os parâmetros do *paging*. Levando-se em conta o impacto das combinações de valores para *page scan window* e *page scan interval* avaliados sobre o *stream* usado, os valores de 18 e 256 *slots*, para *page scan window* e *page scan interval* respectivamente, foram escolhidos para os testes da Seção 4.1.2. Usando essa configuração, a EB conseguiu conexão com o DM em média a 258,142 ms (em torno de 413 *slots*).

page scan window	***page scan interval***	**Tempo *Paging* (ms)**	**Número Aprox. *Slots***	**Res. Aval. Subjetiva**
18	1024	721,565	1154	Perfeito
18	512	419,988	672	Perfeito
18	256	258,142	413	Perfeito
18	128	200,419	321	Poucas Falhas
36	1024	736,239	1221	Perfeito
36	512	436,807	699	Poucas Falhas
36	256	282,414	452	Poucas Falhas
36	128	211,516	338	Muitas Falhas
54	1024	733,559	1173	Poucas Falhas
54	512	440,288	704	Muitas Falhas
54	256	285,061	456	Muitas Falhas
54	128	207,727	332	Muitas Falhas

Tabela A.1: Resumo dos resultados dos testes com diferentes parâmetros para o *paging*.

Apêndice B

Aplicações

O enfoque do trabalho apresentado foi a definição de um protocolo de *handoff* para transição de conexões físicas entre pontos de acesso *Bluetooth*, que viabilizasse o uso de aplicações de tempo-real em dispositivos móveis mesmo quando estes se deslocam através das áreas de cobertura de diferentes pontos de acesso. Particularmente, as aplicações de tempo-real vislumbradas durante o desenvolvimento da solução são aquelas que precisam transmitir e/ou receber dados pela rede. Provavelmente a principal classe de aplicações a se encaixar nesse requisito são as aplicações de *streaming*.

Nesse contexto, nas próximas seções são enumeradas algumas possíveis aplicações futuras do trabalho proposto. Um requisito comum a todos os casos de uso a seguir é a necessidade de se manter constante a transmissão dos *streams* mesmo quando o usuário se move através das áreas de cobertura de diferentes pontos de acesso. Ou seja, é necessário fazer a adequada gerência de *handoff*. Idealmente, estas transições entre pontos de acesso devem ser feitas de forma a não interromper o trânsito dos *streams* e nem degradar sua qualidade.

B.1 Reprodução de Áudio e/ou Vídeo em Movimento

Provavelmente esta seja a aplicação mais direta de uma infra-estrutura de *streaming* para dispositivos móveis. O cenário geral é aquele onde aplicações rodando em dispositivos móveis consomem *streams* disponibilizados por servidores numa rede cabeada, independente se são *streams* pré-gravados ou transmissões ao vivo.

Uma vez conectado a algum ponto de acesso, o usuário informa à sua aplicação cliente

de *streaming* preferida o identificador do *stream* que deseja assistir. Em geral, esse identificador será uma URI [1] com diversas informações sobre o *stream* a ser consumido, como protocolo usado, endereço e porta onde contactar a aplicação servidora de *streams*, e identificadores extras para indicar qual dos *streams* previamente armazenados no servidor deverá ser transmitido (e.g. *rtsp://200.222.12.43:7743/VideosF1/silverstone2007*).

Aplicações efetivas para essa abordagem incluem, por exemplo, assistir/ouvir cameras/microfones de segurança instalados em algum local remoto, ou apenas assistir TV ou ouvir rádio via *webcast* [2], por exemplo.

B.2 Visitas Guiadas

Visitas guiadas são exemplos particularmente interessantes de como informações sobre o contexto dos usuários e aplicações de *streaming* podem se integrar. Por informações sobre contexto, neste caso de uso, nos referimos a dados atuais sobre as localizações dos usuários e as direções para onde eles se movem.

Como exemplo de aplicação de visita guiada, pode-se imaginar um visitante conhecendo as instalações de alguma empresa. Ao entrar na empresa, o visitante pode usar seu *smartphone* para conectar-se com o sistema de visita guiada daquele local. Enquanto ele ainda se encontra na sala de entrada da empresa, ele começa a receber em seu *smartphone* áudio e vídeo pré gravados, dando detalhes sobre o que é aquela empresa e o que se faz lá. O uso de uma tecnologia sem fio de curto alcance, como *Bluetooth*, possibilita que os *streams* recebidos pelo visitante vão se adequando à medida que este caminha pela empresa. Ou seja, ao caminhar um pouco e entrar num corredor, será redirecionado para o dispositivo do visitante um *stream* explicando o que acontece em cada uma das salas que aquele corredor dá acesso. Ao entrar em alguma sala, o *stream* mais uma vez pode ser alterado, dessa vez para dar mais detalhes sobre os projetos e atividades que são desempenhadas naquela sala. O sistema pode ainda incluir no *stream* informações sobre os atuais ocupantes da sala, dizendo quem são e quais seus cargos/papéis.

A idéia pode ser mapeada para diferentes negócios, como museus, galerias, ou qualquer

[1] *Universal Resource Identifier*

[2] Transmissão de áudio e vídeo pela Internet

outro ambiente de visitação pública. Fazendo-se uma mesclagem entre tecnologias de curto e médio alcance, é possível ainda criar visitas guiadas para ambientes bem maiores, como pontos turísticos que ocupem grandes áreas, bairros, ou mesmo cidades inteiras.

B.3 VoIP

Nos últimos anos a tecnologia de telefonia via Internet VoIP vêm se firmando como uma alternativa aos serviços de telefonia tradicionais, principalmente por causa do seu baixo custo. Numa visão de longo prazo, quando redes 4G [50] forem predominantes, usuários de *smartphones* no meio de uma ligação e/ou seção de dados usando uma rede de longo alcance tarifada poderão transferir automaticamente suas sessões de voz e dados para uma infra-estrutura de rede local mais barata ou eventualmente gratuita, como *Wi-Fi*. A idéia é tirar proveito das capacidades de dispositivos multimodais [55] e do acesso local a redes sem fio de baixo custo, e de menor alcance, para economizar dinheiro.

Enquanto este cenário não se torna realidade, uma opção mais simples e imediata para o mesmo tipo de economia consiste em usar VoIP, ao invés ligações via GSM tarifadas, por exemplo, sempre que este serviço estiver disponível. O cenário imaginado é aquele onde empresas e residências, com acesso dedicado à Internet, disponibilizem pontos de acesso sem fio. Usuários poderiam optar, no momento da ligação, usar essas infra-estruturas para fazer ligações de baixo custo. Áreas grandes, obviamente, demandariam mais de um ponto de acesso para serem totalmente cobertas. Se por um lado tecnologias como *Wi-Fi*, com alcance em torno de 100m, reduzem a probabilidade/necessidade de gerenciamento de *handoff*, por outro lado elas esgotam mais rapidamente a autonomia dos dispositivos clientes. Tecnologias como *Bluetooth*, com menor alcance, possuem uma probabilidade/necessidade maior de gerenciamento de *handoff*. Em compensação, demandam apenas 20% do consumo de energia dos seus concorrentes de maior alcance [65].

B.4 *Push to Talk*

Em aplicações de *Push to Talk* (PTT), dispositivos de telefonia, como celulares e *smartphones*, passam a atuar de forma semelhante aos seus antecessores, os *walk talks*. Em PTT

a comunicação passa a acontecer entre grupos de dispositivos, ao invés de apenas dois como naturalmente acontece em telefonia. Atualmente, o trafego de chamadas via PTT acontece via conexões de longa distância usando a infra-estrutura da operadora, o que implica tarifação pelo uso do serviço e custo para o cliente.

Assim como discutido na Seção B.3, usuários poderiam se beneficiar da presença de uma infra-estrutura local sem fios também para a comunicação via PTT. A idéia seria estabelecer conexões com pontos de acesso locais, em residências e empresas, para a comunicação via PTT. Apesar de ser uma conexão com uma infra-estrutura local, os participantes de um grupo PTT podem, assim como o PTT oferecido pelas operadoras, estar geograficamente dispersos. Basta que as infra-estruturas cabeadas, às quais cada usuário do grupo está ligado via algum ponto de acesso, estejam ligadas a uma mesma rede em comum, como a Internet, para que os tráfegos sejam adequadamente roteados.

Apêndice C

Escalabilidade e Qualidade de Serviço

A preocupação com questões de escalabilidade é natural quando se pensa, por exemplo, em aplicações como as discutidas no Apêndice B. Mesmo o melhor dos protocolos de *handoff* pode ser inútil se sistemas baseados em tal solução saturam com apenas poucos usuários simultâneos. Além disso, o fato de se lidar com aplicações com fortes requisitos para a transmissão constante e pontual de dados significa que, mais uma vez, eventuais decisões de projeto para um sistema escalável devem ser conciliadas com as aplicações fins do sistema. Ou seja, a solução final deve ser escalável ao mesmo tempo que não prejudica a qualidade de serviço de usuários já conectados ao sistema.

Durante o desenvolvimento deste trabalho, questões de escalabilidade e qualidade de serviço foram consideradas em diversos momentos. Entretanto, por uma questão de foco do trabalho sobre o procedimento de *handoff* em si, algumas questões marginais mas importantes, como escalabilidade, tolerância a falhas, e definição de mecanismos para localização de dispositivos, tiveram prioridade reduzida. Entretanto, neste Apêndice são discutidas possíveis soluções relacionados à escalabilidade e à qualidade de serviço tomando como base o protocolo proposto no Capítulo 3, assim como o arcabouço de uma infra-estrutura para a evolução do protocolo original.

C.1 Os Gargalos de Escalabilidade

Analisando o protocolo descrito no Capítulo 3, são identificados dois pontos onde o aumento de demanda causará gargalos de escalabilidade: o número de clientes e a quantidade de

pedidos de *handoff* que podem ser atendidos ao mesmo tempo em cada EB. Como o número de clientes simultâneos que cada EB consegue comportar é conseqüência da capacidade de sua interface Bluetooth de estar conectada a mais de um dispositivo ao mesmo tempo, temos que sete é o número máximo de clientes servidos por cada EB ao mesmo tempo. Atender *handoffs* também tende a causar problemas de desempenho. Pelo protocolo atual, para cada pedido de *handoff* que a EB precisa atender, tentativas de conexão com o DM em questão são executadas até que o dispositivo seja contactado ou até que alguma outra EB informe contato com aquele DM (Seção 3.2). O problema surge quando a EB precisa gerenciar vários pedidos de *handoff* ao mesmo tempo. Nesse caso, cada pedido precisa ser atendido de forma seqüencial, o que pode implicar perda de contato com algum DM cujo pedido de *handoff* demorou muito tempo para ser executado pela EB.

Uma outra dimensão para o problema de escalabilidade é revelada quando são também considerados os recursos limitados do sistema. Ou seja, apesar de cada interface da EB poder comportar até sete clientes, sua capacidade de banda é também limitada [1] e ainda precisa ser dividida entre cada DM atualmente conectado. Isto significa que a capacidade de uma EB pode, por exemplo, saturar com apenas um DM. De forma similar, DMs em *handoff* para alguma EB podem não encontrar nela uma disponibilidade de banda suficiente para suas aplicações em curso. O resultado desta mudança, considerando que o *handoff* seja completado, é que o novo DM tende a congestionar o canal sem fio da nova EB, tendo prejudicada a qualidade de suas aplicações e a qualidade das aplicações em curso em outros DMs também conectados àquela EB.

Tomando como base o protocolo já existente, a solução direta para a questão de escalabilidade consiste em aumentar a quantidade de interfaces Bluetooth em cada EB do sistema. Para ter algum controle sobre a qualidade do serviço oferecido pelas EBs, é necessário ainda definir os mecanismos pelos quais recursos são alocados entre DMs e garantidos enquanto estes estejam conectados ao sistema. Nesse contexto, as organizações lógica e física da EBs e DMs, apresentadas no Capítulo 3, são estendidas a seguir para acomodar as novas necessidades de escalabilidade e qualidade de serviço. As especificações dos novos elementos lógicos e físicos são apresentados a partir da próxima seção. Por questão de simplicidade, serão focadas aqui essencialmente as diferenças em relação ao que já foi colocado no Capí-

[1] Largura de banda bruta de 1 Mbps para interfaces 1.1 e 1.2, e 3 Mbps brutos para interfaces 2.0 e 2.1.

tulo 3.

C.2 Definindo um Sistema Escalável

A visão geral do novo sistema é a mesma apresentada anteriormente na Figura 3.1. Assim como no protocolo original, as entidades físicas do sistema continuam sendo EBs e DMs. Os estados que DMs assumem no sistema também são os mesmos definidos na Seção 3.1.1, mas o registro de DMs em cada EB, descrito na Seção 3.1.2, precisa agora guardar informações sobre as trocas de dados atualmente ativas para cada DM (Figura C.1), necessárias para o controle de admissão descrito na Seção C.2.4.

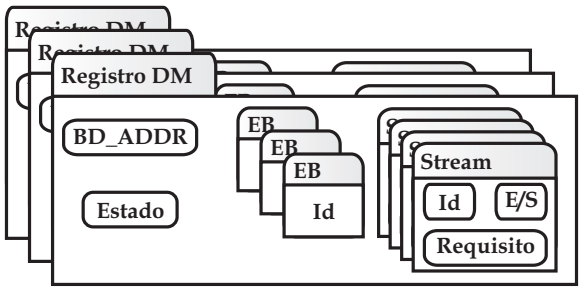


Figura C.1: Registro de EBs com informações sobre *streams* em curso.

Arquiteturalmente, as principais mudanças, em relação ao protocolo apresentado, aconteceram nas EBs, onde agora supõe-se que existam diversas interfaces Bluetooth ligadas a elas ao mesmo tempo. Esta suposição pode ser dita realística dado o atual baixo custo desse tipo de interface. Para implementar um controle de admissão nas EBs, e assim ter algum controle sobre a qualidade do serviço oferecido aos DMs, a camada de *software* responsável pelo gerenciamento dos *handoffs* cresceu e acoplou-se às aplicações dos usuários. O arcabouço da solução agora apresentada, baseia-se na própria abordagem usada pelo Bluetooth SIG [2] para o desenvolvimento de aplicações especializadas usando o Bluetooth: a criação de perfis. Perfis encapsulam o comportamento comum a toda uma classe de aplicações, servindo então como base para o desenvolvimento de aplicações finais especializadas. Na prática, perfis realizam-se como camadas de *software* que facilitam o desenvolvimento de novas aplicações, oferecendo aos desenvolvedores uma interface mais alto nível, especializada em algum tipo

[2] *Bluetooth Special Interest Group* - Grupo responsável pelo desenvolvimento do Bluetooth.

de tarefa, e que abstrai as dificuldades do manuseio direto da interface HCI, ou ainda do uso de serviços oferecidos por outros perfis. Assim, e focando num primeiro momento apenas aplicações de *streaming*, foi definido um perfil, HoSP (sigla para *Handing-off Streams Profile*), independente de aplicação. A implementação deste perfil encapsula a lógica por trás do gerenciamento de *handoff*, controle de admissão, comunicação inter-EBs, e a troca de dados no enlace entre DMs e EB. Parte da responsabilidade da EB, que é dependente de aplicação, é delegada para uma aplicação externa, responsável por representar a aplicação final do usuário, que roda no DM, junto à rede cabeada. Uma visão geral dessa nova organização é ilustrada na Figura C.2. A seguir são descritas as novas entidades físicas e lógicas do sistema e suas funções.

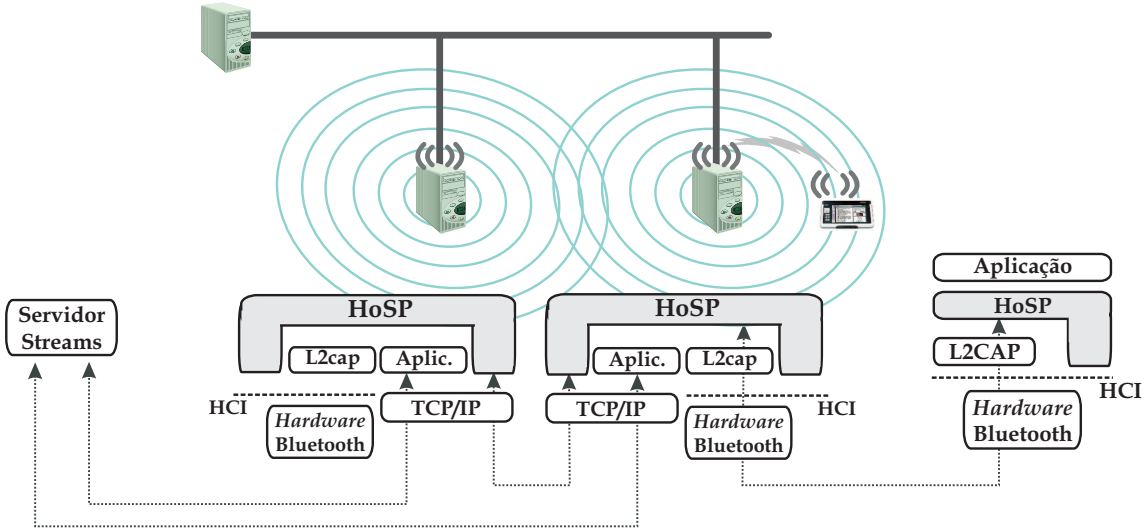


Figura C.2: Nova organização lógica do sistema.

Do lado da EB, a camada HoSP implementa a lógica para monitorar DMs, requisitar *streams*, conceder a transmissão de *streams*, controle de admissão de DMs, escalonamento de pedidos de *handoff*, e comunicação entre EBs. No lado do cliente, implementa as regras para entrada no sistema, interação com a EB para requisição de recepção/transmissão de *streams*, monitoração de conexões estabelecidas e perdidas, e gerenciamento de múltiplas conexões. Para ter acesso às informações e à conectividade necessária por seus módulos, a camada HoSP têm acesso direto à camada HCI e à rede local TCP/IP. No enlace com DMs a troca de dados é feita via L2CAP.

C.2.1 As Estações Base

Em sua nova organização, EBs possuem várias interfaces Bluetooth, onde distinguem-se dois tipos de interfaces (Figura C.3): interfaces de provisão de serviço (IPSs); e interfaces de ponto de entrada (IPEs). Cada EB possui, pelo menos, duas interfaces Bluetooth disponíveis. Neste cenário mínimo, cada interface irá assumir um dos papéis anteriores. Cada tipo de interface possui uma configuração própria, definida de acordo com seu papel no sistema.

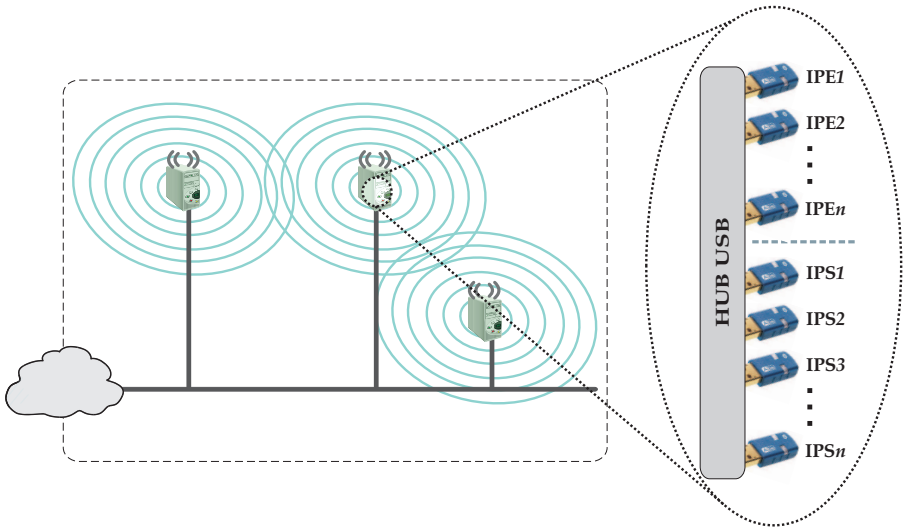


Figura C.3: Interfaces de Provisão de Serviço (IPSs) e Interfaces de Ponto de Entrada (IPEs).

Desde que não estejam ocupadas atendendo algum pedido de *handoff*, IPEs são configuradas para responder tanto *inquiries* quanto *pagings* e possuem duas funções: servir como ponto de entrada no sistema para novos dispositivos; e tentar contato com DMs durante *handoffs*. Dispositivos externos ao sistema localizam e conectam-se às EBs através das IPEs (Seção C.3.1). Nenhum tráfego útil é realmente transferido por estas interfaces. O segundo tipo de interface, IPS, é responsável pelo efetivo tráfego de dados entre DMs e servidores. DMs só podem receber/transmitir dados quando conectados a uma IPS. IPEs podem ter suas configurações alteradas em tempo de execução para não responder nem *pagings* nem *inquiries* como forma de aumentar a disponibilidade da interface durante *handoffs* (Seção C.3).

Diferente das IPEs, as IPSs são configuradas para não responderem nem *inquiries* nem *pagings* de outros dispositivos. Assim, isola-se o tráfego de dados em tempo-real de grande parte das perturbações que eles poderiam sofrer por parte de outros dispositivos Bluetooth

próximos. Como IPSs não respondem *inquiries*/*pagings* de outros dispositivos, as conexões entre IPSs e DMs são iniciadas pela EB.

Cada EB pode dispor de quantas IPSs e IPEs quantas forem possíveis/necessárias, desde que existam, pelo menos, uma interface de cada tipo. A forma direta de escalar o sistema, com relação à capacidade de clientes servidos simultaneamente, é aumentar a quantidade de IPSs nas EBs, aumentando assim o número de clientes suportados em cada EB e a quantidade de recursos disponíveis para eles. De maneira similar, IPEs ocupadas com pedidos de *handoff* tentem a tornar-se um gargalo de desempenho durante esse processo. Assim, a forma de escalar o sistema com relação à capacidade de *handoffs* simultâneos numa mesma EB é aumentar a quantidade de IPEs da mesma. Como visto na Seção 2.5.4, o tráfego de dados em tempo-real sobre Bluetooth é consideravelmente afetado pelas operações de *paging* e *inquiry*. Esta é a razão por trás da diferenciação entre IPEs e IPSs.

C.2.2 Os Dispositivos Clientes

Nos DMs, aplicações finais são desenvolvidas sobre a camada HoSP. Assim como já descrito na Seção 3.2, toda a monitoração e procedimentos para trocas de EBs são tarefas das EBs. Continua cabendo aos DMs apenas "perceber" os estabelecimentos e perdas de conexões com as EBs e reagir de forma adequada. Considerando ainda o nicho inicial para o qual o HoSP foi pensado (aplicações de *streaming*) e os procedimentos para controle de admissão (vide Seção C.2.4), temos que o HoSP pode encapsular todo o comportamento comum à gerência de conexões, trocas de dados no enlace com as EBs, e interação com o controle de admissão das EBs, deixando para as aplicações finais apenas o tratamento comum ao tráfego de *streaming* via redes de melhor esforço (e.g. ordenação de pacotes fora de ordem e descarte de pacotes muito atrasados ou redundantes) e/ou a geração de tráfego para encaminhamento na rede cabeada.

Assim, na camada HoSP dos DMs é anunciada uma interface, ilustrada abaixo, pela qual aplicações especializadas em *streaming*, e que demandem suporte transparente a *handoffs*, podem ser desenvolvidas. Ao construir aplicações sobre essa interface, aplicações automaticamente se beneficiam de facilidades codificadas no HoSP quanto à troca de dados com diferentes EBs, e da gerência de *handoffs* implementada na rede cabeada.

```
buscarEBs()
abrirConexaoEB( idEstacaoBase )
solicitaRecepcaoStream( idStream, funcaoCallback )
solicitaTransmissaoStream( destinatario, larguraBanda )
envia( idConexao, dados )
```

Aplicações clientes usam a função *buscarEBs()* para que a camada HoSP faça um *inquiry* e filtre os resultados, retornando apenas uma lista com os dispositivos próximos que são estações base do sistema. A lista retornada por essa função pode ser iterada pela aplicação, que usa a função *abrirConexaoEB(idEstacaoBase)* para tentar conexão com a EB escolhida. Uma vez conectados a alguma IPS, DMs podem solicitar *streams* anunciados por outros nodos da rede através da função *solicitaRecepcaoStream(idStream, funcaoCallback)*. Quando invocada, essa função passa para a camada HoSP da EB o identificador do *stream* desejado. Esta, por sua vez, executa os procedimentos necessários para iniciar ou rejeitar o *stream* solicitado pelo DM (vide Seção C.3.2). Caso o *stream* seja aceito, os dados do *stream* são repassados para a camada HoSP do cliente que, para cada pacote recebido, invoca a função de *callback* informada quando o *stream* foi solicitado. Para a transmissão de *streams* a partir do DM, as funções *solicitaTransmissaoStream(destinatario, larguraBanda)* e *envia(idConexao, dados)* são usadas. A aplicação cliente primeiro informaria à EB que quer transmitir algo, informando o consumo estimado de banda. A EB, mais uma vez, verifica sua disponibilidade de recursos e autorizaria ou não a nova transmissão.

C.2.3 As Aplicações Finais

As aplicações finais do sistema são aplicações de *streaming*. Estas aplicações rodam nos DM e podem tanto transmitir dados para a rede cabeada quanto receber dados de lá. Na organização proposta, aplicações clientes são divididas em duas. Uma parte roda no próprio DM, gerando *streams* e renderizando dados de *streams* recebidos, enquanto outra porção roda junto à EB. Esta segunda porção da aplicação trabalharia em conjunto com a EB para efetivamente interagir com os servidores de *streams*, e repassar para o HoSP dados recebidos da rede cabeada, para que este os encaminhe para o respectivo DM. De forma inversa, *streams* gerados pelos DMs seriam encaminhados para o HoSP da EB, e esta os repassaria

para a aplicação, que seria responsável por encaminhar os dados recebidos na rede cabeada.

Como requisito para a implementação do controle de admissão, a porção da aplicação final que roda na EB deve implementar dois serviços, que são usados pelo HoSP: um para consultar a descrição de *streams*; e outro para efetivamente iniciar o *stream* (vide Seção C.3).

C.2.4 Controle de Admissão

Como forma de ter algum controle sobre a qualidade do serviço oferecido aos DMs e reduzir a probabilidade de *handoffs* mau sucedidos devido atraso em seu atendimento, foi especificado um mecanismo de controle de admissão nas EBs. O controle de admissão decide se novos DMs podem ou não conectar-se a uma EB, seja este um DM anteriormente externo ao sistema ou um DM sob *handoff*.

Quando usada para atender *handoffs*, cada IPE é configurada para não responder nem *pagings* nem *inquiries* de dispositivos próximos. O objetivo é priorizar os recursos de cada IPE para os *handoffs*, mesmo que isso atrase a entrada de novos DMs no sistema. A principal função do controle de admissão é rejeitar pedidos de *handoff* caso não existam recursos suficientes para acomodar o novo dispositivo na EB de destino. Tentar contato com DMs que não podem ter seus *handoffs* completos, devido à falta de recursos (vagas nas *piconets* das IPSs e/ou largura de banda nas IPSs disponíveis pelo critério anterior) na EB de destino, significa aumentar, em vão, o tempo de espera de outros pedidos de *handoff* na fila. Este tempo extra de espera pode comprometer o sucesso dos *handoffs* de outros dispositivos, que eventualmente demandam menos recursos e que poderiam ser alocados na nova EB. Dessa forma, conforme ilustrado no Algoritmo C.1, cada pedido de *handoff* escalonado têm seus requisitos de uso de banda avaliados antes de serem efetivamente executados, assim como também é verificada a disponibilidade de *slots* nas IPSs disponíveis. Apenas caso, no momento do escalonamento do pedido, existam recursos suficientes em uma das IPSs da EB, a requisição será atendida. Caso contrário, a requisição é devolvida para a fila de requisições e uma nova escolha é feita. Na seleção de IPS, ilustrada no Algoritmo C.2, é escolhida a interface que possua *slots* livres em sua *piconet* e que tenha a menor largura de banda disponível que seja maior ou igual ao requisito de banda do DM sob *handoff*.

Algoritmo C.1: Escalonando requisições de *handoff*

```
1  ENQUANTO VERDADEIRO FACA
2    // escolhe uma requisição da fila, ou bloqueia caso a fila esteja
         vazia
3    reqHandoff ← escalonaRequisicaoHandoff()
4    // vide Algoritmo C.2
5    IPS_ComMenorLarguraDeBanda ← selecionaIPS(reqHandoff.dadosEB)
6    SE(IPS_ComMenorLarguraDeBanda = NULO) ENTAO
7        devolveParaFila(reqHandoff)
8        CONTINUE
9    FIM SE
10   ipe ← escalonaIPE(reqHandoff)
11   resultadoTentativa ← tentaConexao(reqHandoff.dadosEB, ipe)
12   SE(resultadoTentativa = VERDADEIRO) ENTAO
13       IPS_ComMenorLarguraDeBanda ← selecionaIPS(reqHandoff.dadosEB)
14       SE(IPS_ComMenorLarguraDeBanda = NULO) ENTAO
15           // disconecta DM da IPE
16           rejeitaDMPorFaltaDeRecurso(reqHandoff)
17       SENAO
18           // desconecta DM da IPE
19           realocaDM(IPS_ComMenorLarguraDeBanda, reqHandoff.dadosEB)
20           disparaMonitorDM(reqHandoff.dadosEB)
21           registroLocal.atualizaRegistro(reqHandoff.dadosEB)
22           notificaEBsVizinhas(reqHandoff.dadosEB)
23           restabeleceStreamsEntrada(reqHandoff.dadosEB.streamsEntrada)
24           restabeleceStreamsSaida(reqHandoff.dadosEB.streamsEntrada)
25       FIM SE
26   SENAO
27       // também libera a IPE. Caso seja o último pedido de handoff
             associado à IPE, ela é reconfigurada para voltar a aceitar
             pagins e inquiries
28       devolveParaFila(reqHandoff)
29   FIM SE
30 FIM ENQUANTO
```

Como não é feita a reserva de recursos na IPS antes de iniciar o *handoff* para um determinado DM, caso seja conseguido contato com o DM o escalonador checa novamente a

disponibilidade de IPSs. Apesar de existirem recursos para acolher aquele DM no momento em que seu pedido de *handoff* foi escalonado, eventualmente parte desses recursos podem ter sido alocados para outro DM que conseguiu finalizar primeiro seu *handoff* através de outra IPE da EB. Assim, o *handoff* para o um DM ainda pode ser rejeitado mesmo após o primeiro contato com a EB. EBs vizinhas apenas são notificadas sobre contato de uma outra EB com um DM sob *handoff* caso aquela EB consiga re-alocar o DM para uma de suas IPSs.

Algoritmo C.2: Escolhendo IPS durante *handoff*.

```
1  menorLarguraDeBandaDisponivel ← ∞
2  IPS_ComMenorLarguraDeBanda ← NULO
3  listaDeIPSs ← listaDeIPSs()
4  PARA TODOS IPS EM listaDeIPSs FACA
5     larguraDeBandaDisponivel ← larguraDeBandaDisponivel(IPS)
6     // dados sobre o DM recebidos como parâmetro
7     SE(larguraDeBandaDisponivel ≤ menorLarguraDeBandaDisponivel) E (
          larguraDeBandaDisponivel ≥ dadosDM.larguraDeBandaUsada) E (
          possuiSlotLivre(IPS)) ENTAO
8          menorLarguraDeBandaDisponivel ← larguraDeBandaDisponivel
9          IPS_ComMenorLarguraDeBanda ← IPS
10    FIM SE
11 FIM FOR
12 RETORNA IPS_ComMenorLarguraDeBanda
```

C.3 Ilustrando o Funcionamento do Novo Sistema

A seguir, o funcionamento do novo sistema é ilustrado através de alguns cenários de uso. São descritos os processos de entrada de novos DMs no sistema, a requisição de *streams* anunciados na rede cabeada, a requisição para transmitir *streams* em direção à rede cabeada, e o processo de *handoff*. O objetivo é proporcionar uma visão prática dos mecanismos pensados para melhorar a característica de escalabilidade do protocolo original.

C.3.1 Entrando no Sistema

O fluxo de atividades para entrada no sistema é ilustrado na Figura C.4.

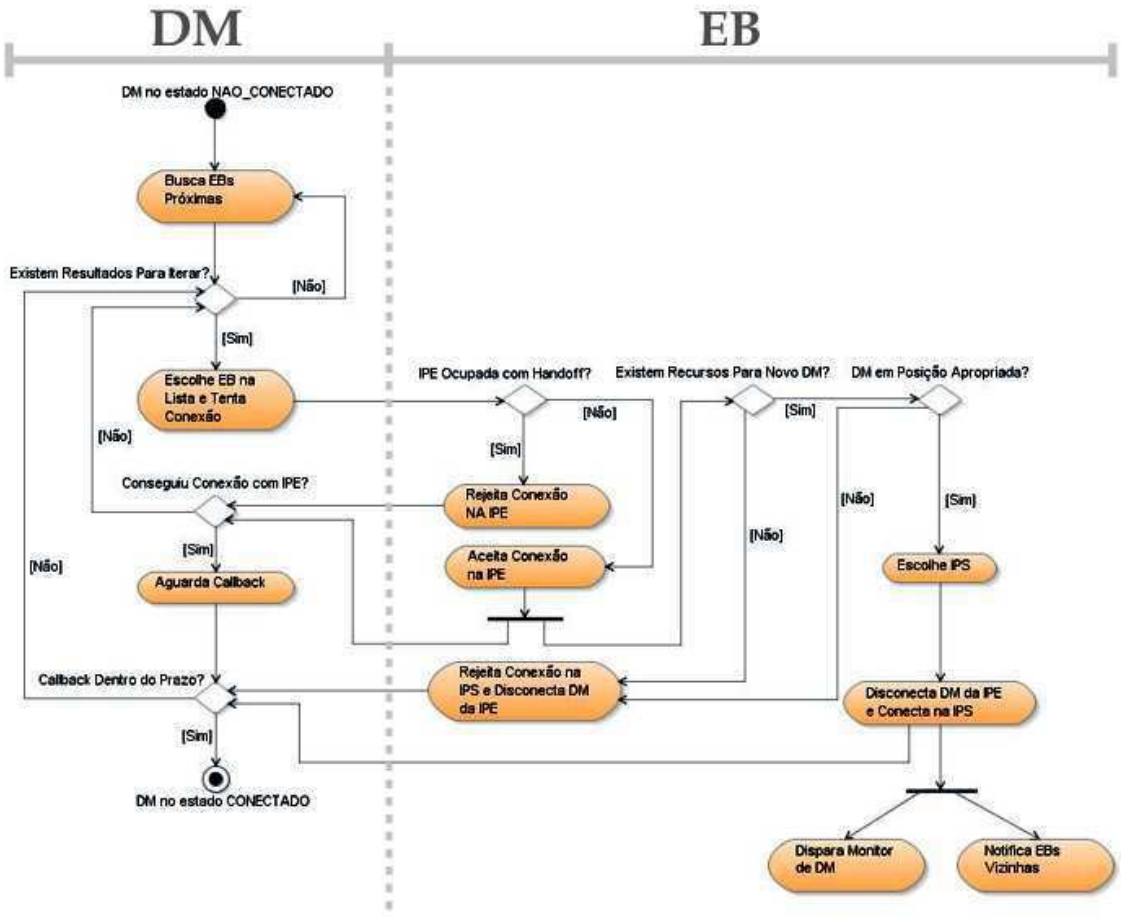


Figura C.4: Fluxograma com as etapas para entrada de DMs no sistema.

Como primeiro passo para entrada no sistema, aplicações finais, construídas sobre a camada HoSP, solicitam a busca por EBs próximas. A implementação dessa busca na camada HoSP filtra os resultados, retornando apenas os dispositivos Bluetooth que são EBs. Em seguida, é feita uma tentativa de conexão com uma das EBs retornadas, seja a partir de de uma decisão automática da aplicação, ou por escolha do usuário.

O processo de conexão entre DM e EB acontece em duas fases. Como as IPSs são configuradas para não responderem nem *pagings* nem *inquiries*, então as únicas interfaces que respondem à busca do DM são as IPEs, caso não estejam ocupadas com o *handoff* de algum DM. Após se conectar a alguma das IPEs da EB escolhida, inicia-se a segunda parte do processo de entrada, onde o DM aguarda uma chamada de *callback*, vinda de alguma das IPSs daquela EB. A chamada de *callback* na verdade é uma nova conexão *baseband* entre DM e EB, mas desta vez iniciada pela EB a partir de uma de suas IPSs.

Do lado da EB, o primeiro passo ao receber uma conexão de um novo cliente numa de

suas IPEs, é verificar se existem recursos disponíveis para acomodar mais um cliente. Como não se sabe qual a largura de banda será usada pelo DM, a decisão por aceitar ou não o novo DM consiste, neste momento, na disponibilidade de *slots* em alguma de suas IPSs. Caso exista pelo menos uma vaga na *piconet* em uma das IPSs da EB, é checada então a distância estimada entre esse novo dispositivo e a EB. Caso o cliente esteja numa posição além de onde o processo de *handoff* é disparado, então sua conexão é rejeitada. Caso o usuário encontre-se antes do ponto a partir do qual o processo de *handoff* é disparado, a EB prossegue com a conexão e usa uma estratégia gulosa para decidir à qual de suas IPSs o novo DM será conectado. Neste segundo momento de checagem de recursos, a EB itera sobre todas as IPSs com vagas em suas *piconets*, e escolhe aquela que têm maior largura de banda disponível. A largura de banda usada por cada IPS é estimada baseado nas descrições dos *streams* atualmente transferidos por aquela EB. Como a EB não sabe, a priori, qual será o consumo de banda do novo DM, a estratégia gulosa consiste em alocar o novo DM na IPS com maior largura de banda disponível.

Após decidir para qual IPS o DM será alocado, a EB guarda temporariamente o endereço BD_ADDR do DM (disponível à EB assim que foi aceita a conexão *baseband* entre DM e IPE). Estas informações são usadas para, após desconectar o DM da IPE, estabelecer uma nova conexão com o dispositivo cliente a partir da IPS escolhida (*callback* aguardado pelo DM).

Ao finalizar a chamada de *callback*, o DM é registrado na EB como estando no estado CONECTADO e sua distância em relação à EB passa a ser monitorada. Neste estado, o DM está pronto para requisitar ou transmitir *streams*. Após cadastrar o DM em sua tabela própria de DMs, a EB atual também envia para todas as suas EBs vizinhas uma mensagem, informando que aquele dispositivo está ligado a ela e seu estado é CONECTADO. Cada EB vizinha, ao receber essa mensagem, inclui uma referência para aquele DM em seu registro de DMs.

C.3.2 Iniciando Transmissões

Uma vez no estado CONECTADO, DMs podem solicitar às EBs a recepção de *streams* anunciados na rede. O processo de requisição de *streams* é ilustrado na Figura C.5. Primeiro, a aplicação final no DM passa para a EB o identificador do *stream* desejado. O identificador,

que pode ser, por exemplo, uma URL RTSP ou SIP, é passado para a aplicação do lado da EB para que esta interprete seu conteúdo e interaja com os servidores daquele recurso.

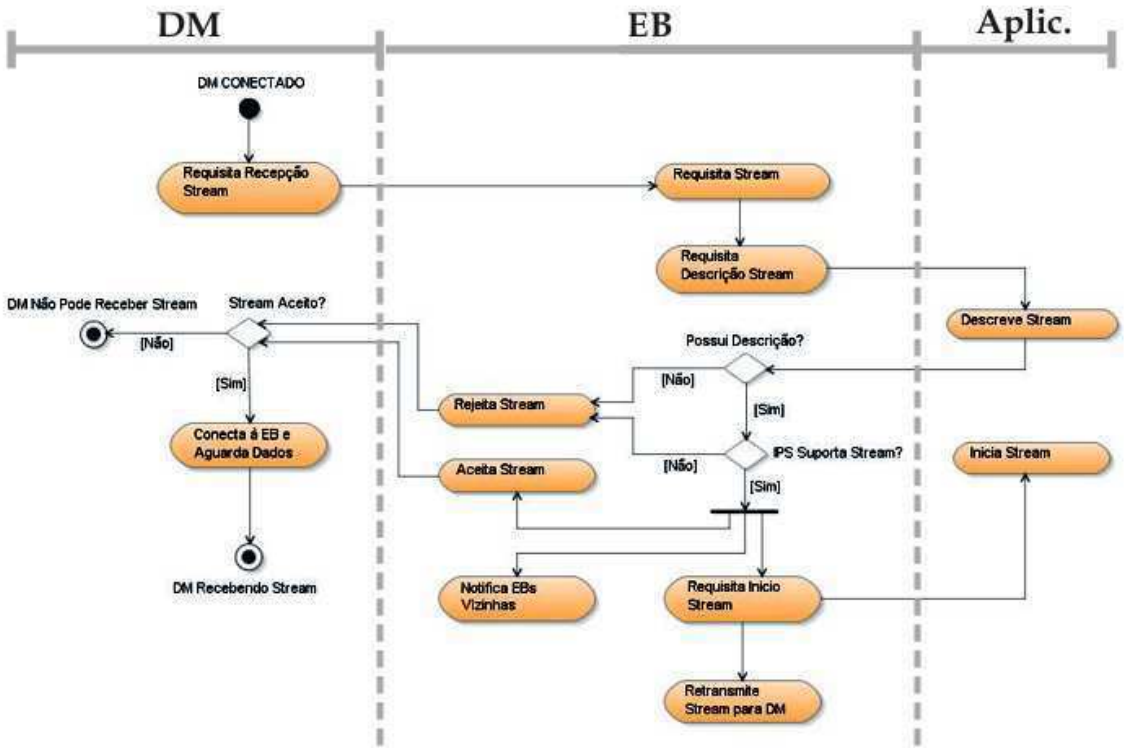


Figura C.5: Etapas para iniciar a recepção de *streams*.

Na EB, o processo de requisição de *streams* acontece em duas fases, podendo ser rejeitado. Primeiro, ao receber uma requisição de *stream* de algum DM ao qual serve, a EB delega para a respectiva aplicação auxiliar a tarefa de consultar a descrição do *stream* de interesse, em particular, a estimativa de largura de banda usada por aquele *stream*. Caso não seja possível descobrir de antemão o consumo médio de banda da um determinado *stream*, então sua requisição é rejeitada. Caso contrário, a EB verifica se a IPS à qual o DM está conectado possui recursos livres suficientes para acomodar aquele *stream*. Quando existe banda suficiente, a aplicação auxiliar é mais uma vez solicitada, desta vez para que entre em contato com o servidor do *stream* e inicie sua transmissão. Para cada novo *stream* requisitado à aplicação, é assumido que seus pacotes serão retransmitidos dentro do domínio de mobilidade como tráfego *multicast* e que o endereço *multicast* usado será retornado para a EB ao final do comando da aplicação para o inicio do *stream*. Os detalhes para a realização desta suposição são abstraídos aqui, entretanto, possíveis direções para a implementação incluem o MBone [39] e diversas patentes públicas [1; 15;

71]. O endereço *multicast* do *stream*, seu tipo (neste caso, *stream* de entrada), e sua estimativa de banda, são notificados para todas as EBs vizinhas. Com a confirmação do aceite do seu pedido, o DM estabelece uma conexão com a EB para a transferência de dados e aguarda sua chegada. Do lado da EB, assim que a aplicação auxiliar começa a receber os pacotes do *stream*, estes são passados para o HoSP, que os encaminha para o DM.

A segunda hipótese, onde o DM inicia uma transmissão, é ilustrada na Figura C.6. Antes de iniciar uma transmissão o DM precisa solicitar o recurso em sua EB. Para a solicitação o DM informa a estimativa de consumo do *stream* que pretende transmitir. Assim como no caso anterior, a EB verifica se a IPS na qual o DM está conectado possui recursos suficientes para acomodar o novo *stream*. Em caso afirmativo, o pedido do DM é aceito, as EBs vizinhas são notificadas do tráfego do novo *stream*, e os dados do *stream* vindos do DM são passados para a aplicação, que se encarrega de encaminhar os dados na rede cabeada. Sempre que um *stream* é aceito pela EB, essa informação também é incluída no registro local sobre o DM antes de ser propagada para as EBs vizinhas.

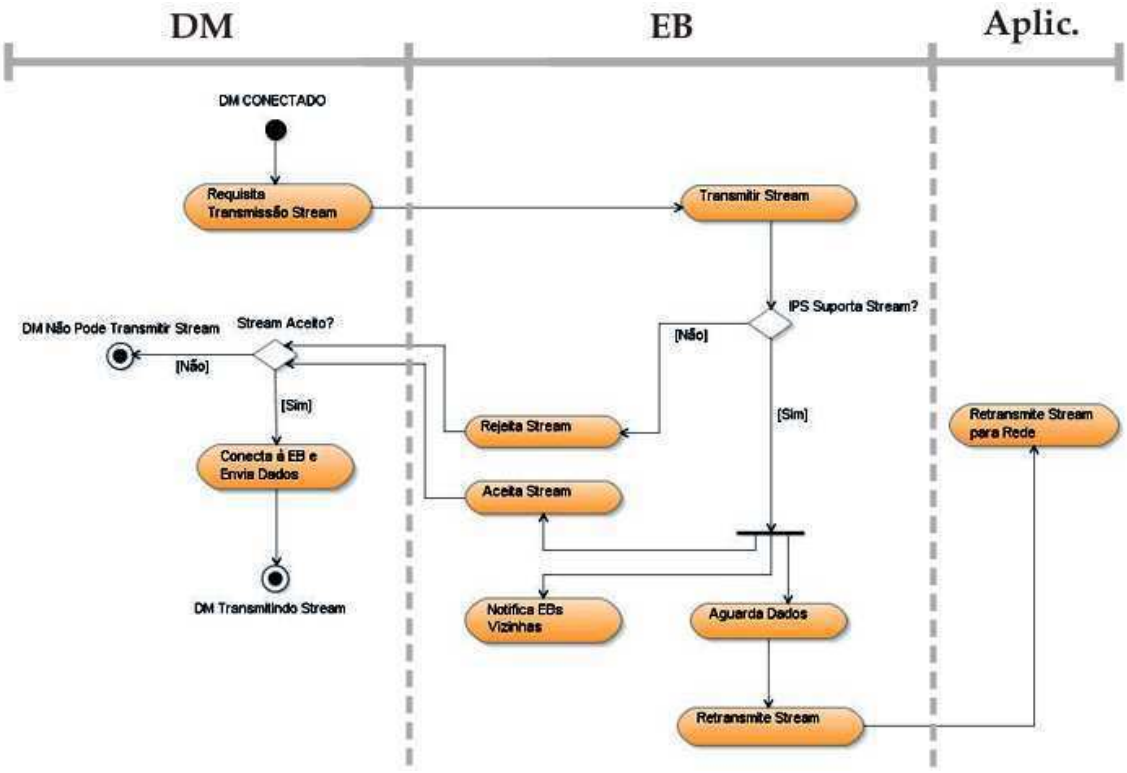


Figura C.6: Etapas para iniciar a transmissão de *streams*.

C.3.3 *Handoff* no Nível da Aplicação

Durante o processo de *handoff* o DM mantém-se conectado a mais de uma EB ao mesmo tempo. Como não existe informação precisa sobre a direção na qual o usuário se move, fechar conexão com uma das EBs antes do final do *handoff* pode causar perda total de conexão caso a conexão fechada seja justamente a da EB para a qual o DM se movia. Assim, durante os *handoffs*, os *streams* de entrada são repassados para o DM, ao mesmo tempo, por todas as EBs às quais o DM está conectado, assim como dados de *streams* de saída são encaminhados por todas as conexões disponíveis. A responsabilidade de se lidar com dados replicados fica por conta das aplicações finais, que naturalmente já implementam esse tratamento, assim como também implementam tratamento para outros problemas relacionados ao tráfego de *streaming* em redes de melhor esforço.

Sob a aplicação cliente, a camada HoSP monitora o estabelecimento e perda de conexões *baseband* entre DM e EBs. Cada nova conexão *baseband* implica no estabelecimento de conexões fim-a-fim entre DM e EB para cada *stream* em curso, enquanto a perda de conexões *baseband* causa o fechamento destas conexões. No tráfego de dados da rede em direção ao DM, o dispositivo atua como servidor na conexão com a EB. Ou seja, ao ser estabelecida uma conexão fim-a-fim l2cap entre DM e EB para o encaminhamento de um *stream*, a aplicação fornece à camada inferior uma função para *callback*. Essa função é chamada pela camada sempre que uma carga útil encaminhada pela EB, normalmente um pacote RTP lido da rede cabeada, chega por completo. Quando múltiplas conexões estão abertas entre o DM e EBs (durante o *handoff*), a camada HoSP inicia diferentes receptores de dados, um para cada conexão, e repassa para a aplicação as cargas úteis recebidas. No tráfego de dados do DM para a rede, o DM atua como cliente na sua conexão com a EB. As cargas úteis a serem transmitidas são passadas da aplicação para a camada HoSP, que por sua vez envia uma cópia da carga útil recebida para cada conexão ativa com diferentes EBs. Cada EB ao receber essa carga a repassa para a aplicação, que faz seu encaminhamento via rede cabeada.

C.4 Discussões Adicionais

Neste Apêndice foram discutidas algumas idéias preliminares para a expansão do protocolo original deste trabalho em direção a uma infra-estrutura mais madura do ponto de vista de

escalabilidade e qualidade de serviço. As soluções aqui discutidas são um arcabouço do que pode ser o próximo passo na evolução do protocolo descrito no Capítulo 3. Essencialmente, temos que o sistema é tão escalável quanto seja possível implantar mais interfaces Bluetooth em cada EB. Ao adicionar mais IPSs aumenta-se também a capacidade das EBs de comportarem mais clientes ao mesmo tempo e a quantidade total de recursos disponíveis naquela EB. De forma análoga, aumentando-se o número de IPEs aumenta-se a quantidade de pedidos de *handoff* que podem ser atendidos ao mesmo tempo.

Para diminuir a probabilidade de perda de conexões com DMs, devido atrasos no atendimento de seu pedido de *handoff*, foi também descrita uma especificação de um mecanismo de controle de admissão. Este mecanismo usa informação sobre as aplicações atualmente em curso para um DM sob *handoff* para decidir se vale ou não à pena tentar contato com aquele DM. Caso a EB perceba que não possui vaga em nenhuma de suas IPSs para o novo DM, ou que, mesmo havendo essa vaga, não existem recursos suficientes para acomodar os *streams* em uso pelo DM, então a EB não executa o pedido de *handoff* para aquele dispositivo. Assim, reduz-se o número de pedidos na fila por uma IPE e, conseqüentemente, reduz-se a probabilidade de que um pedido que realmente pode ser executado seja mau sucedido devido atraso para sua execução. No controle de admissão também são definidos procedimentos para acomodar a entrada de novos dispositivos, antes externos ao sistema.