UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO DE UM AMBIENTE DE DESENVOLVIMENTO DE SERVIÇOS BASEADOS EM PRESENÇA UTILIZANDO ESPECIFICAÇÃO DO WIRELESS VILLAGE SOBRE PLATAFORMA IEEE 802.11 (W-LAN)

GUSTAVO HENRIQUE SOUSA FERREIRA
VITOR HUGO VIEIRA LOPES

ORIENTADOR: PAULO H. PORTELA DE CARVALHO

PROJETO FINAL DE GRADUAÇÃO

BRASÍLIA / DF: DEZEMBRO/2003

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO DE UM AMBIENTE DE DESENVOLVIMENTO DE SERVIÇOS BASEADOS EM PRESENÇA UTILIZANDO ESPECIFICAÇÃO DO WIRELESS VILLAGE SOBRE PLATAFORMA IEEE 802.11 (W-LAN)

GUSTAVO HENRIQUE SOUSA FERREIRA
VITOR HUGO VIEIRA LOPES

PROJETO FINAL DE GRADUAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO.

APROVADA POR:

PAULO HENRIQUE PORTELA DE CARVALHO, Doutor, UnB
(ORIENTADOR)

LEONARDO R. A. X. MENEZES, Ph. D., UnB
(EXAMINADOR)

ADSON FERREIRA DA ROCHA, Ph. D., UnB
(EXAMINADOR)

DATA: BRASÍLIA/DF, DEZEMBRO DE 2003

FICHA CATALOGRÁFICA

FERREIRA, GUSTAVO HENRIQUE SOUSA
LOPES, VITOR HUGO VIEIRA
Implementação de um Ambiente de Desenvolvimento de Serviços Baseados em Presença utilizando Especificação do Wireless Village sobre Plataforma IEEE 802.11 (W-LAN) [Distrito Federal] 2003.
xvi, 95p., 297 mm (ENE/FT/UnB, Bacharel, Engenharia Elétrica, 2003).

Projeto Final de Graduação – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. IEEE 802.11 2. Wireless Village
3. Location Based Services (LBS)

I. ENE/FT/UnB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

FERREIRA, GUSTAVO HENRIQUE SOUSA e LOPES, VITOR HUGO VIEIRA (2003). Implementação de um Ambiente de Desenvolvimento de Serviços Baseados em Presença utilizando Especificação do Wireless Village sobre Plataforma IEEE 802.11 (W-LAN). (Projeto Final de Graduação), Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 95p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Gustavo Henrique Sousa Ferreira e Vitor Hugo Vieira Lopes
TÍTULO DA DISSERTAÇÃO: Implementação de um Ambiente de Desenvolvimento de Serviços Baseados em Presença utilizando Especificação do Wireless Village sobre Plataforma IEEE 802.11 (W-LAN).
GRAU/ANO: Bacharel/2003.

Gustavo Henrique Sousa Ferreira
QE 19 conjunto J casa 35 Guará 2
CEP 71050-103 – Brasília – DF - Brasil

Vitor Hugo Vieira Lopes
3ª Avenida Bloco 1780 casa 23
CEP 71720-023 – Brasília – DF - Brasil

AGRADECIMENTOS

Os autores agradecem a Deus, acima de tudo. Ao nosso orientador, professor Paulo Portela por todo o apoio, incentivo e motivação. Ao professor Leonardo Menezes, aos demais participantes do Projeto Nokia – UnB e em especial aos amigos Alex Helder, Maisa, Arildo e Andréa, pela grande ajuda na confecção deste trabalho.

Gustavo agradece aos seus pais e avós, e a todos de sua família, pelo apoio e compreensão nas horas difíceis. A todos os seus amigos, pelas farras que ajudaram a superar estes momentos. À galera do G13 (Bruno Guimarães, Bruno Miranda, Bruno Oliveira, Daniel França, Gustavo Wagner, Hanibal Gazzola, Marco Antônio, Paulo Garcia, Regis Machado, Saulo Martins, Vitor Hugo e Victor Godoy) e a todos que esqueceu, mas que de alguma forma contribuíram nesses longos e melhores anos da sua vida.

Vitor agradece aos seus pais pela formação exemplar recebida, primando sempre pelo estudo, responsabilidade e dignidade, e pelo apoio na superação de todos os obstáculos encontrados nessa jornada de conhecimentos. A sua namorada, Marina Borgatto, que sempre demonstrou compreensão com o tempo requerido para o desenvolvimento e implementação do projeto. Aos mesmos colegas de faculdade mencionados pelo Gustavo, com a inclusão deste, que por todo o curso se mantiveram unidos tanto nas ocasiões de diversão como naquelas de dedicação e ajuda mútua.

DEDICATÓRIA

Gustavo e Vitor dedicam este trabalho às suas famílias

RESUMO

O presente trabalho visa implementar um Ambiente de Desenvolvimento de Serviços Baseados em Presença, utilizando especificação do *Wireless Village*, sob plataforma IEEE 802.11 (W-LAN). A motivação por trás deste projeto é o constante crescimento da indústria de comunicações móveis e, conseqüentemente, da demanda por esse tipo de serviço. As tecnologias utilizadas na implementação do sistema, tais como J2ME, Servlets, XML e MySQL, representam a última geração para o desenvolvimento de aplicações *wireless*. Uma vez implementado, o ambiente é capaz de suprir as necessidades básicas de desenvolvedores de serviços IMPS, disponibilizando tanto a troca de mensagens instantâneas como a informação de presença em si.

ABSTRACT

The present work aims at implementing an Environment for the Development of Presence Based Services, as specified in the Wireless Village initiative and under the IEEE 802.11 Platform (W-LAN). The motivation that pushed us towards this project was the mobile communications industry constant growth, witch, in turn, drives up the demand for this kind of service. The technologies used while implementing this system, such as J2ME, Servlets, XML and MySQL, are among the newest ones for the development of wireless applications. Once operating, the Environment developed is capable of supplying the basic needs of IMPS developers, as it gives support to both Instant Messaging and Presence information.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIAÇÕES

Abreviação	Significado
AID	Association ID
ANSI	American National Standards Institute
AP	Access Point
API	Applications Programming Interface
ARP	Address Resolution Protocol
BSA	Basic Service Area
BSS	Basic Service Set
BTS	Base Transceiver Subsystem
CBS	Cell Broadcast Service
CDC	Connected Device Configuration
CDMA	Code Division Multiple Access
CIR	Communications Initiation Request
CLDC	Connected Limited Device Configuration
CLI	Command Line Interface
CLP	Command Line Protocol
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSP	Client-Server Protocol
DFC	Distributed Coordination Function
DFWMAC	Distributed Foundation Wireless MAC
DIFS	Distributed Coordination Function IFS
DOM	Document Object Model
DSSS	Direct-Sequence Spread-Spectrum
DTD	Document Type Definition
E-OTD	Enhanced Observed Time Difference
ESS	Extended Service Set
FCC	Federal Communications Commission
FHSS	Frequency-Hopping Spread-Spectrum
GCF	Generic Connection Framework
GPRS	General Packet Radio Service
GSM	Global System of Mobile Communications
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Security
IBSS	Independent Basic Service Set
IEEE	Institute of Electrical and Eletronic Engineering
IFS	Interframe Spaces
IM	Instant Messaging
IMPS	Instant Messaging and Presence Services
ISM	Industrial Scientific and Medical
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JAR	Java Archive
JAXB	Java API for XML Binding

JAXP	Java API for XML Parsing
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDBC	Java Database Connectivity
JNI	Java Native Interface
JSR	Java Specification Request
JVM	Java Virtual Machine
KVM	Kilo Virtual Machine
LAN	Local Area Network
LCS	Location Server
LLC	Logical Link Control
MAC	Medium Access Control
MCN	Mobile Core Network
MIB	Management Information Base
MID	Mobile Information Device
MIDP	Mobile Information Device Profile
MIN	Mobile Identification Number
MLME	MAC Layer Management Entity
MMS	Multimedia Messaging Service
OSI	Open System Interconnection
OTA	Over-The-Air
PCF	Point Coordination Function
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PIFS	Point Coordination Function IFS
PLCP	Physical Layer Convergence Procedure
PLME	Physical Layer Management Entity
PMD	Physical Medium Dependent
RMI	Remote Method Invocation
SAP	Service Access Point
SAX	Simple API for XML
SIFS	Short IFS
SMCNP	Server-Mobile Core Network Protocol
SME	System Management System
SMS	Short Messaging Service
SQL	Structured Query Language
SSP	Server-Server Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Universal Modeling Language
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WMA	Wireless Messaging API
WML	Wireless Markup Language
WSP	Wireless Session Protocol

WTLS	Wireless Transport Layer Security
WV	Wireless Village
WVS	Wireless Village Server
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

1. INTRODUÇÃO

Durante os últimos cinco anos, o estilo de vida da população mundial tem se tornado cada vez mais dinâmico. Como resultado, os modos tradicionais de se interconectar o mundo se provaram inadequados para satisfazer os desafios impostos pelo atual estilo de vida. Dentro dessa concepção tradicional, a mobilidade de um usuário conectado a uma rede é drasticamente reduzida na medida em que essa conexão é realizada através de cabos físicos. Uma conexão *wireless*, no entanto, não impõe tais restrições. Esta é a razão do enorme sucesso da telefonia *wireless*: ela permite às pessoas se conectarem umas às outras independentemente de sua localização.

O panorama das comunicações *wireless* vem atravessando uma rápida mudança na medida em que os telefones celulares e as redes *wireless* vêm sendo aprimorados, de forma a propiciar a criação de serviços que vão além dos simples serviços de voz.

O movimento em direção à computação móvel se dá a passos largos. A venda de dispositivos móveis e *Smart Cards* bate de longe a de seus concorrentes fixos e, com a adoção de tecnologias como o *Bluetooth* e *Wireless* LAN, as plataformas computacionais se tornarão cada vez mais móveis [1].

Até 2003/4, quase todos os telefones celulares vendidos já irão incorporar o acesso a dados via conexão *wireless.* O mercado de *Smart Cards* já alcança a marca de 500 milhões de unidades por ano, com uma expectativa de crescimento de 40% ao ano. A previsão de crescimento do mercado de *Wireless* LAN é de aproximadamente 30% ao ano para os próximos 4 anos [1].

A indústria *wireless* presencia agora a rápida expansão dos serviços de dados para dispositivos móveis. Tal expansão pode ser explicada por uma série de fatores tais como a convergência dos domínios da Internet e das redes *wireless;* e a crescente demanda por parte dos consumidores e profissionais por novas e cada vez mais complexas aplicações *wireless.*

Dentre os serviços mais requeridos, aqueles que vem apresentando a maior procura são os Serviços Baseados em Troca de Mensagens Instantâneas e Informações de Presença – IMPS (*Instant Messaging and Presence Services*) [2].

No entanto, o sucesso desse tipo de serviço estará diretamente ligado à capacidade do mercado *wireless* de produzir uma solução baseada num padrão que dê suporte à interoperabilidade entre os diversos fabricantes e ao *roaming*.

É nesse cenário que surge a Iniciativa *Wireless Village*, com o objetivo de assegurar a interoperabilidade de serviços IMPS na medida em que cria uma comunidade entre fabricantes, desenvolvedores e consumidores a fim de prover o mercado com serviços IMPS inovadores.

O presente trabalho se propõe a unir as qualidades de mobilidade, flexibilidade, facilidade de configuração, gerenciamento e operação presentes na plataforma IEEE 802.11 para redes *Wireless* (W-LAN) com a proposta de interoperabilidade, portabilidade e suporte ao desenvolvimento de Serviços Baseados em Troca de Mensagens Instantâneas e Informações de Presença sugerida e especificada pela iniciativa *Wireless Village*.

Para tanto, este texto visa explicar em detalhes toda a modelagem, estudo, implementação e teste de um Ambiente de Desenvolvimento de Serviços Baseados em Presença utilizando Especificação do *Wireless Village* sobre Plataforma IEEE 802.11 (W-LAN). Este ambiente utiliza tecnologias e sistemas *open source* para dar suporte à implementação de serviços IMPS, tais como Java, J2ME, JDBC, Servlets, MySQL e XML, todos sob sistema operacional Linux.

Em linhas gerais, o Ambiente consiste na implementação de parte da solução *Wireless Village*, rodando sobre um servidor de informações de presença desenvolvido com base na plataforma de Redes Locais Wireless – W-LAN (Wireless Local Area Network) – do padrão IEEE 802.11.

O texto está estruturado em seis capítulos. O Capítulo 1 consiste na presente Introdução, onde é apresentado o panorama atual das comunicações *wireless*, a motivação para o desenvolvimento do trabalho e uma breve descrição sobre a que o mesmo se destina. O Capítulo 2 apresenta uma descrição do padrão IEEE 802.11 para Redes *Wireless*, abrangendo suas principais características técnicas e funcionais de forma a propiciar ao leitor uma base para o entendimento do sistema de presença desenvolvido. No Capítulo 3 é apresentada a solução *Wireless Village*, descrevendo o conceito por traz dos IMPS, bem como a arquitetura do sistema e os serviços os quais a solução visa suportar. O Capítulo 4 discorre sobre as tecnologias utilizadas na confecção do Ambiente e situa o leitor no cenário tecnológico atual,

uma vez que as tecnologias utilizadas representam a vanguarda do desenvolvimento de aplicações *wireless*. São apresentadas as seguintes seções: Java 2 Platform, Micro Edition (J2ME); Servlets; Conceitos da Linguagem XML; e O Banco de Dados MySQL. No Capítulo 5 é apresentada a implementação do Ambiente de Desenvolvimento de Serviços Baseados em Presença. São explicadas em detalhe a arquitetura utilizada, o Servidor de Informação de Presença implementado e a interface com o usuário disponibilizada. Finalmente, no Capítulo 6 são descritas as Conclusões alcançadas bem como as Perspectivas para trabalhos e desenvolvimentos futuros.

2. O PADRÃO IEEE 802.11 PARA REDES WIRELESS (W-LAN)

Como um resultado da incapacidade das atuais redes em satisfazer os desafios impostos pela dinâmica de vida moderna, as tecnologias de rede *wireless* vêm se desenvolvendo rapidamente. Dentre essas tecnologias, pode-se destacar o padrão 802.11 como aquela que têm obtido maior aceitação e sucesso até o presente momento. Dessa forma, para compreender melhor esse mundo *wireless* se torna fundamental o estudo desse referido padrão.

Essas redes estão crescendo em um ritmo bastante acelerado e, recentemente, já têm grandes funcionalidades. Muitos comerciantes já as utilizam para autorização de cartão de crédito e outros processos ligados a pontos de venda, linhas aéreas estão instalando em aeroportos e salas de espera e grandes empresas as usam para oferecer uma maior dinamicidade aos funcionários [3].

Áreas com constante concentração de pessoas tais como aeroportos, shoppings, cinemas, cafés, centros de lazer em geral, tendem, ultimamente, a instalar os chamados *hot spots.* Tais *hot spots* são, simplesmente, pontos de acesso WLAN de alta velocidade disponíveis ao público, por um determinado preço ou mesmo de graça, para fins de *marketing*. Diante da presença de tais pontos, a conexão discada, ou mesmo uma conexão via celular, se torna desvantajosa para o cliente visto que o este dispõe de uma largura de banda maior nesses pontos.

Tendo em vista uma melhor explanação do tópico abordado, inicia-se o capítulo com uma introdução preliminar sobre redes *wireless* seguida de uma visão geral sobre redes 802.11. Na seqüência, apresenta-se a idéia central por trás do controle de acesso ao meio (MAC – Medium Access Control) e por fim, objetivando abordar o contexto de implementação do projeto, finaliza-se o capítulo enunciando os principais temas relacionados às operações internas de controle das redes *wireless*.

2.1. INTRODUÇÃO ÀS REDES WIRELESS

Mesmo desconhecendo a forma com que os protocolos são projetados ou, ainda, o tipo de dados que estas redes *wireless* carregam, sabe-se que elas apresentam diversas vantagens sobre as redes tradicionais.

A vantagem mais óbvia é a mobilidade, característica esta que permite o livre trânsito de usuários após a conexão a uma dada rede. Dessa forma, enquanto os usuários permanecerem dentro da área de cobertura de uma estação base, estes podem se utilizar dos benefícios da redes.

Tipicamente, redes *wireless* são caracterizadas por flexibilidade, ou seja, dada uma determinada base, a expansão da mesma se dá de forma fácil e versátil. Em suma, redes *wireless* usam várias estações para conectar usuários a uma rede existente. Entretanto, a infra-estrutura de uma rede desse porte é qualitativamente a mesma independente do número de usuários que estejam sendo conectados. Uma vez que a infra-estrutura está construída, acrescentar um usuário é, basicamente, uma questão de autorização.

Assim como todas as redes, redes *wireless* transmitem dados sobre um meio. O meio é uma forma de radiação eletromagnética. Para estar bem adaptado para uso em redes móveis, o meio deve ser capaz de cobrir uma ampla área para permitir aos usuários uma boa qualidade de conexão por toda área de cobertura. Vista a boa capacidade de penetração das ondas de rádio, esta tem sido definida como a principal camada física do mercado atual.

2.1.1. Espectro de Rádio: o Recurso Chave

Dispositivos *wireless* estão restritos a operar em uma determinada banda de freqüência. O uso do spectro de rádio é rigorosamente controlado por entidades reguladoras que, no caso dos Estados Unidos, é realizado pela Comissão Federal de Comunicações (FCC – Federal Communications Commission). Para evitar a sobreposição de ondas de rádio, a freqüência é alocada em bandas, que são, simplesmente, faixas de freqüências disponíveis para aplicações específicas. A Tabela 2.1 lista algumas das bandas utilizadas nos Estados Unidos.

Tabela 2.1 – Espectro de freqüências utilizadas em comunicações móveis

Band	Frequency range
UHF ISM	902–928 MHz
S-Band	2–4 GHz
S-Band ISM	2.4–2.5 GHz
C-Band	4–8 GHz
C-Band satellite downlink	3.7–4.2 GHz
C-Band Radar (weather)	5.25–5.925 GHz
C-Band ISM	5.725–5.875 GHz
C-Band satellite uplink	5.925–6.425 GHz
X-Band	8–12 GHz
X-Band Radar (police/weather)	8.5–10.55 GHz
Ku-Band	12–18 GHz
Ku-Band Radar (police)	13.4–14 GHz
	15.7–17.7 GHz

Na Tabela 2.1, existem três bandas denominadas ISM, que é uma abreviação para *Industrial* (área industrial), *Scientific* (área científica) e *Medical* (área médica). Essas bandas são atribuídas, de forma geral, para equipamentos relacionados a processos industriais, científicos ou médicos e, geralmente, têm licença livre. Tais bandas merecem atenção especial visto que os dispositivos 802.11 operam nelas. Pode-se verificar na Tabela 2.2 uma breve comparação dos padrões 802.11.

Tabela 2.2 – Comparação entre os diversos padrões IEEE 802.11

IEEE standard	Speed	Frequency band	Notes
802.11	1 Mbps 2 Mbps	2.4 GHz	First standard (1997). Featured both frequency-hopping and direct-sequence modulation techniques.
802.11a	up to 54 Mbps	5 GHz	Second standard (1999), but products not released until late 2000.
802.11b	5.5 Mbps 11 Mbps	2.4 GHz	Third standard, but second wave of products. The most common 802.11 equipment as this book was written.
802.11g	up to 54 Mbps	2.4 GHz	Not yet standardized.

2.1.2. Os Limites da Rede Wireless

De uma forma geral, redes *wireless* não substituem redes fixas. Apesar da vantagem da mobilidade dos usuários das redes, servidores e outros equipamentos centrais precisam, unicamente, acessar ou processar dados e a posição dos mesmos é irrelevante. Desta forma, equipamentos dessa ordem podem continuar ligados a fios em busca de recursos que as redes *wireless* não podem oferecer.

A velocidade das redes *wireless* está limitada à largura de banda disponível para o sistema. A menos que as autoridades reguladoras decidam tornar maiores as faixas de freqüência não licenciadas, sempre haverá um limite máximo de velocidade imposto às redes *wireless*. Além dessa restrição de banda, redes *wireless* tendem a ser mais devagar que redes com fio. Diferentemente do padrão Ethernet 10-GB, padrões de rede sem fio devem, cuidadosamente, validar quadros recebidos para se precaver quanto a perdas ligadas ao meio.

O uso de ondas de rádio impõe vários desafios. Primeiramente, ondas de rádio podem sofrer diversos problemas de propagação que podem interromper a conexão, tal como o multi-percurso.

Segurança em qualquer rede é um conceito primário. Em redes *wireless*, este conceito é, freqüentemente, crítico haja visto que as transmissões estão disponíveis para qualquer um que esteja nos limites do transmissor com uma antena adequada. Dessa forma, e considerando que os transmissores de rádio são projetados para serem processados por qualquer receptor, *sniffing*[1] é muito mais fácil em redes sem fio. Além disso, redes sem fio têm fronteiras não determinadas que, de uma certa forma, dificulta, ainda mais, o controle do acesso ao meio.

2.2. VISÃO GERAL SOBRE REDES 802.11

Uma visão geral é sempre necessária quando se estuda tópicos relacionados a redes porque o número de acrônimos pode se tornar impressionante.

Inicialmente, é importante compreender que 802.11 é superficialmente similar à Ethernet, motivo este que leva muitos a chamá-lo de *wireless Ethernet*. No entanto, existem diversas particularidades que 802.11 precisa para adaptar a tecnologia Ethernet tradicional para o mundo *wireless*.

[1] Termo que faz referência ao processo de captura, autorizada ou não, de todos os pacotes que circulam na rede em um determinado período de tempo.

2.2.1. Família da Tecnologia de Redes IEEE 802

802.11 é um membro da família IEEE 802, que é uma série de especificações para tecnologia de redes locais (LAN – *Local Area Network*). Figura 2.1 mostra a relação entre os vários componentes da família 802 e seus devidos lugares no modelo OSI.

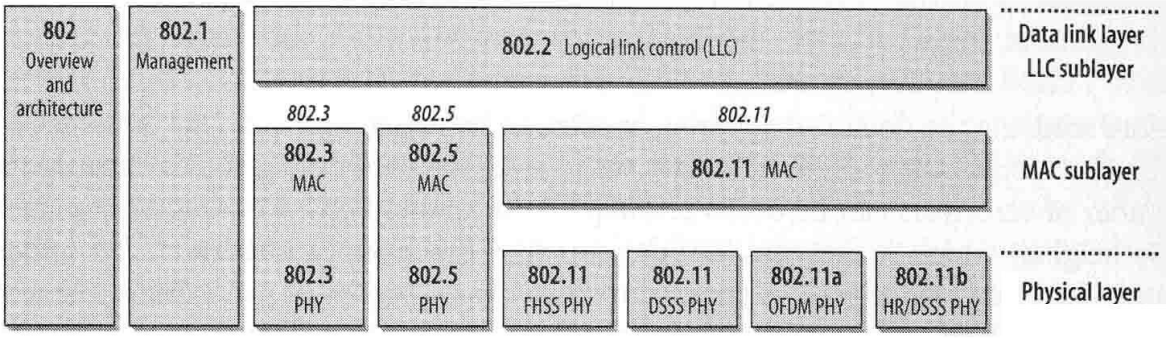

Figura 2.1 – A família 802.11 e sua correspondência ao Modelo OSI

As especificações 802 focam nas duas camadas inferiores do modelo OSI visto que elas incorporam tanto os componentes físicos como os de conexão de dados. Todas as redes 802 têm um componente MAC e um físico (PHY). O MAC é um conjunto de regras para determinar como acessar o meio e enviar dados, mas os detalhes da transmissão e recepção são deixados para o PHY.

As especificações individuais da série 802 são identificadas pelo segundo número. Por exemplo, 802.2 especifica uma camada de conexão comum, o Controle Lógico de Conexão (LLC – *Logical Link Control*), que pode ser usado por qualquer tecnologia LAN de camada inferior.

A especificação de redes *wireless*, 802.11, é apenas outra camada de conexão que pode usar a encapsulação LLC/802.2. A base da especificação 802.11 inclui o MAC 802.11 e duas camadas físicas: uma de espalhamento de espectro por salto de freqüência (FHSS - *Frequency-Hopping Spread-Spectrum*) e outra de espalhamento de espectro por seqüência direta (DSSS - *Direct-Sequence Spread-Spectrum*).

O uso de ondas de rádio como meio para a camada física requer uma PHY relativamente complexa. 802.11 divide a PHY em dois componentes genéricos: a Camada Física de Convergência de Procedimentos (PLCP – *Physical Layer Convergence Procedure*), para mapear os quadros MAC no meio, e um sistema de Meio Físico Dependente (PMD –

Physical Médium Dependent) para transmitir estes quadros. O PLCP ultrapassa a fronteira de MAC e camadas física, como pode ser visto na Figura 2.2.

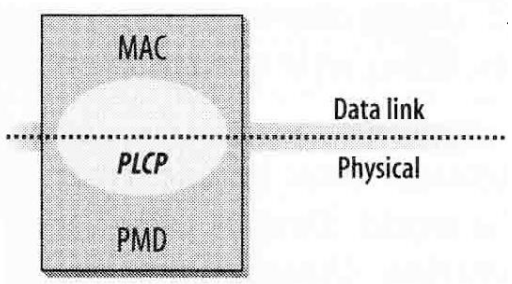

Figura 2.2 – Componentes da Camada PHY

2.2.2. Nomenclatura e Projeto

Redes 802.11 consistem de quatro componentes físicos principais que são visualizados na Figura 2.3.

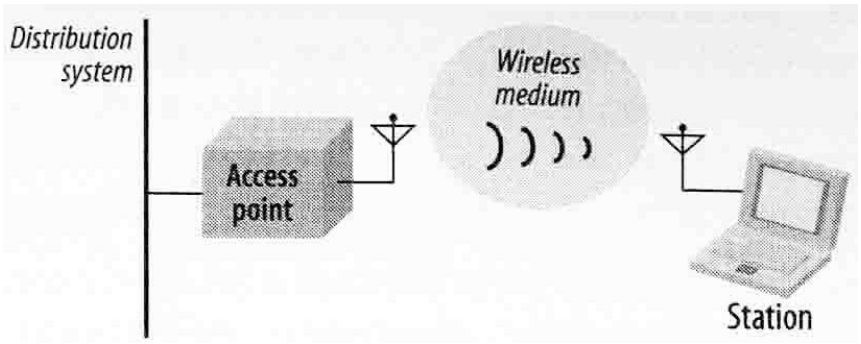

Figura 2.3 – Componentes de uma LAN 802.11

Como pode ser visto na Figura 2.3, esses componentes são:

- Sistema de Distribuição (*Distribution System*): quando diversos *access points* estão conectados para formar uma grande área de cobertura, eles precisam se comunicar entre si para coordenar a movimentação das estações móveis. O Sistema de Distribuição é o componente lógico do 802.11 usado para encaminhar os quadros para seu devido destino. Este sistema é normalmente conhecido como o *backbone* da rede. Em quase todos os produtos comerciais que obtiveram sucesso no mercado, Ethernet é usada como a tecnologia de *backbone* da rede [4].

- *Access Points*: os quadros em uma rede 802.11 precisam ser convertidos em outro tipo de formato para serem entregues à rede externa. Dispositivos chamados *access points* realizam a função de ponte entre a parte sem fio e a com fio.

- Meio sem fio (*Wireless Médium*): para mover quadros de uma estação para outra o padrão utiliza um meio sem fio. A arquitetura permite diferente camadas física para apoiar o MAC 802.11.

- Estações (*Stations*): estações são, simplesmente, dispositivos computacionais com interfaces de rede *wireless*.

2.2.3. Tipos de Redes

O bloco de construção básico em uma rede 802.11 é o Conjunto de Serviços Básicos (BSS – *Basic Service Set*), que é um grupo de estações que se comunicam entre si. Comunicações acontecem em uma área sem limites fixos, chamada de Área de Serviços Básicos (BSA – *Basic Service Area*), definida pelas características de propagação do meio *wireless*. Quando uma estação está na área básica de serviços, ela pode comunicar com outros membros do BSS. Esses BSS's aparecem em duas disposições principais, ilustradas na Figura 2.4.

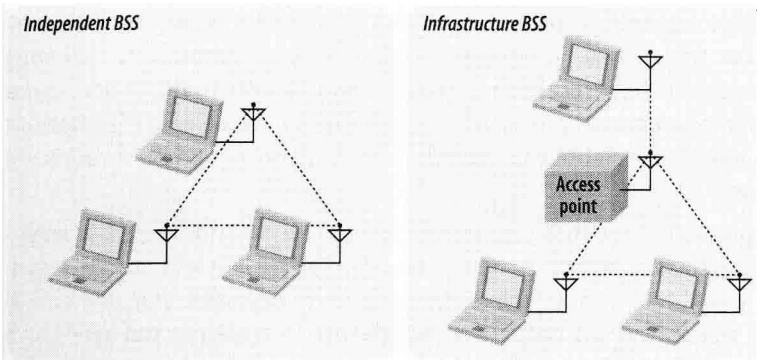

Figura 2.4 – BSS Independente e BSS em Infra-estrutura

2.2.3.1. Redes Independentes

A disposição visualizada à esquerda da Figura 2.4 é uma BSS independente (IBSS – *Independent Basic Service Set*). Estações em uma IBSS comunicam-se diretamente entre si e, portanto, precisam estar em um arranjo que permita visada direta entre as estações. Tipicamente, IBSS's são compostas de um número pequeno de estações colocadas para um propósito específico e por um curto período de tempo.

2.2.3.2. Redes de Infra-estrutura

A disposição visualizada à direita da Figura 2.4 é uma BSS de infra-estrutura. Redes de infra-estrutura se diferenciam pelo uso de *access points. Access points* são usados para

todas as comunicações em uma rede de infra-estrutura. Com todas as comunicações realizadas por meio de um *access point*, a área de serviços básicos para uma BSS de infra-estrutura é definida pelos pontos nos quais transmissões podem ser recebidas do *access point*. Embora as transmissões, no caso das redes de infra-estrutura, precisem de vários passos, no mínimo dois, e conseqüentemente utilizem mais capacidade que um quadro encaminhado direto do emissor para o receptor, a rede de infra-estrutura tem duas vantagens principais:

- Uma BSS de infra-estrutura é definida pela distância do *access point.* Todas as estações móveis devem estar na área de cobertura do *access point*, no entanto, não existem restrições quanto à distância entre as estações propriamente ditas.

- *Access points* em uma rede de infra-estrutura estão em uma posição para monitorar as estações de forma a economizar potência. *Access points* podem perceber quando uma estação entra no modo de economia de energia e *bufferizar* os quadros para ela.

2.2.3.3. Áreas de Serviço Estendidas

A especificação 802.11 permite que redes sem fio de tamanho arbitrário sejam criadas ligando BSS's em um Conjunto de Serviços Estendidos (ESS – *Extended Service Set*). Um ESS é criado encadeando-se BSS's em um mesmo *backbone*. Pode-se verificar na Figura 2.5 um ESS composto da união de quatro BSS's.

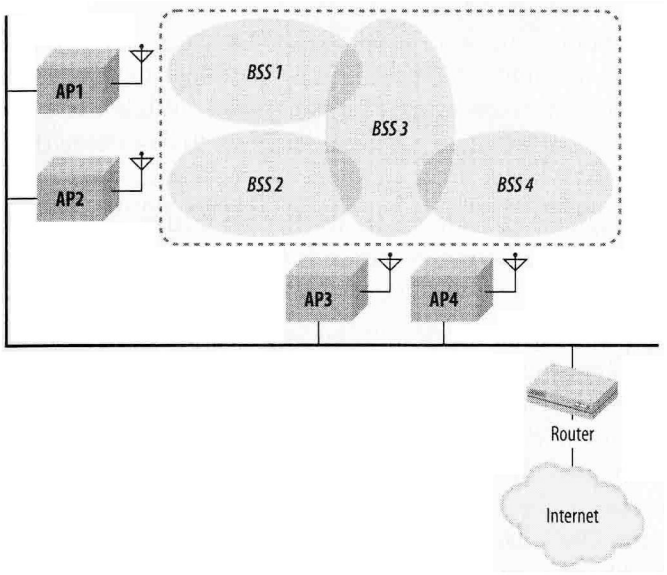

Figura 2.5 – Conjunto de Serviços Estendidos (ESS – Extended Service Set)

Estações no mesmo ESS podem comunicar entre si, mesmo que estas estejam em diferentes áreas de serviços básicos (BSA's) e se movendo entre estas. *Access Points* em uma ESS operam de forma a permitir que o mundo externo use apenas um único endereço MAC para conversar com uma estação localizada dentro do ESS. Na Figura 2-5, o roteador usa um único endereço MAC para entregar os quadros para as estações móveis; o *Access Point* no qual a estação móvel está associada entrega o quadro. O roteador permanece ignorante a respeito da localização da estação móvel.

2.2.4. O Sistema de Distribuição

O sistema de distribuição é responsável por rastrear onde uma estação está fisicamente localizada e entregar os quadros apropriadamente. Obviamente, parte do mecanismo de entrega é realizada pelo *backbone* Ethernet, no entanto, o *backbone* não pode ser o sistema de distribuição por inteiro pois não haveria formas de escolher entre *access points.*

A maioria dos *access points*, hoje em dia, atuam como *bridges.* Eles têm, pelo menos, uma interface de rede *wireless* e, pelo menos, uma interface de rede *Ethernet*. O lado *Ethernet* pode estar conectado a uma rede existente e lado *wireless* se torna uma extensão dessa rede. O trânsito de quadros entre as duas redes é controlado por um sistema de ponte.

A Figura 2.6 ilustra a relação entre o *access point*, *backbone* de rede e o sistema de distribuição. Quaisquer quadros enviados pela porta *wireless* da *bridge* são transmitidos a todas as estações associadas. Cada estação associada pode transmitir os quadros para o *access point*. Finalmente, a porta de acesso ao *backbone* na *bridge* pode interagir diretamente com o *backbone* da rede.

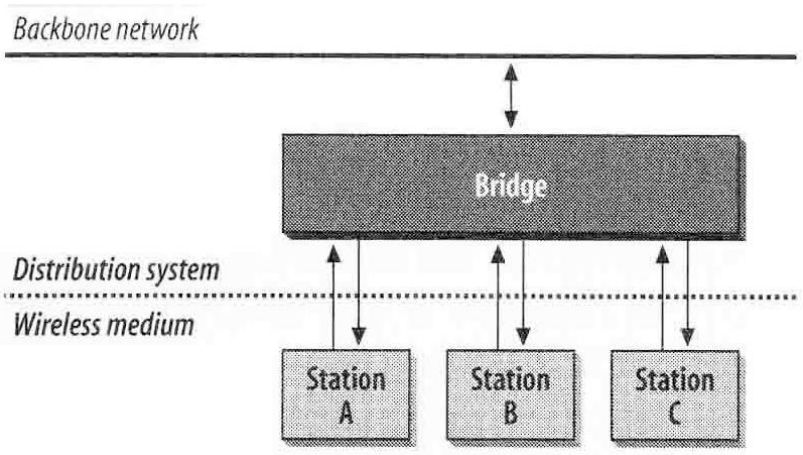

Figura 2.6 – Sistemas de Distribuição comuns às implementações dos AP's 802.11

2.2.5. Fronteiras de Redes

Como dito anteriormente, devido à natureza do meio sem fio, redes 802.11 têm fronteiras esdrúxulas. Desta forma, assim como nas redes telefônicas, permitindo que áreas de serviços básicos se interceptem aumenta a probabilidade de sucesso nas transições entre BSA's e oferece um nível de cobertura maior para a rede. Um exemplo de intersecção de BSS's pode ser visualizado na Figura 2.7.

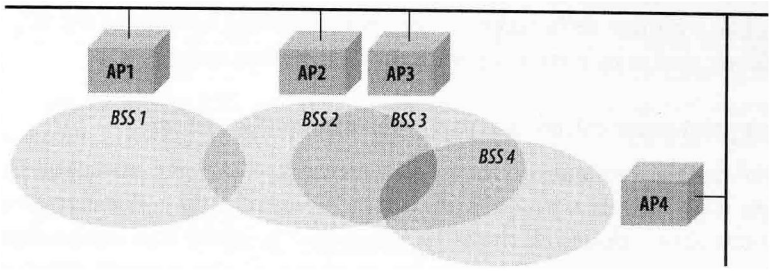

Figura 2.7 – Intersecção de BSS's em uma ESS

2.3. CONTROLE DE ACESSO AO MEIO (MAC) EM 802.11

O 802.11 considerou dois tipos de algoritmos MAC: protocolo de acesso distribuído, o qual, tal como o CSMA/CD, distribui a decisão de transmitir utilizando um mecanismo de escuta ao meio; e protocolos de acesso centralizado, no qual a decisão de transmissão é regulada por um tomador de decisão central.

O resultado escolhido foi um algoritmo MAC chamado DFWMAC(*Distributed Foundation Wireless MAC*) o qual prevê um mecanismo de controle de acesso distribuído com um controle central opcional superior, a Figura 2.8 ilustra a arquitetura.

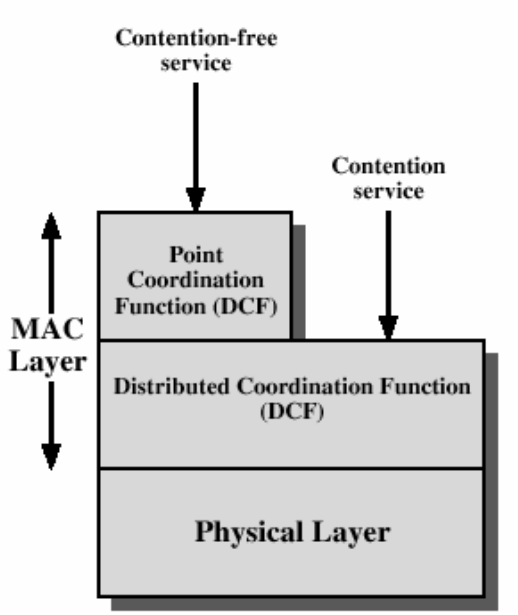

Figura 2.8 - Arquitetura do protocolo IEEE 802.11

A camada inferior da camada MAC é a Função de Coordenação Distribuída (*Distributed Coordination Function* – DCF), a qual utiliza um algoritmo de condensação para garantir acesso para todo o tráfico. A Função de Coordenação Pontual (*Point Coordination Function* – PCF) é utilizada para fornecer serviços sem contenção.

2.3.1. Função de Coordenação Distribuída (DCF)

A subcamada DCF utiliza um simples algoritmo CSMA: se a estação tem um frame MAC para transmitir, ela escuta o meio. Se o meio está livre a estação transmite. Caso contrário, a estação deve esperar até que a transmissão corrente se complete antes de transmitir. O DCF não inclui o sistema de detecção de colisão (isto é, CSMA/CD) pois isto não é praticável em redes Wireless.

A fim de segurar o funcionamento correto e justo deste algoritmo, o DCF inclui um conjunto de delays relativos a um esquema de prioridade. Na verdade existem três diferentes valores de espaço interframes (*Interframe Spaces* – IFS) no entanto consideraremos inicialmente somente um deles. Assim, as regras para o CSMA são:

1. A estação com o frame a transmitir escuta o meio. Se meio estiver livre ela continua na escuta por IFS para verificar se o meio continua livre. Se continuar a estação transmite.

2. Se em qualquer dos casos o meio estiver ocupado a estação defere a transmissão e continua na escuta até que o meio esteja livre.

3. Quando a transmissão atual terminar, a estação espera outro IFS. Se o meio continuar livre a estação se retira utilizando um esquema de retirada exponencial binária e a seguir escuta o meio. Se este ainda estiver livre a estação transmite.

Como forma de fornecer um acesso com prioridade o DCF dispões de três IFS:

- SIFS (*Short IFS*): usado para todas as ações de resposta imediata, tais como ACK e Clear to Send (CTS). É o mais curto.

- PIFS (*Point Coordination Function IFS*): usado pelo controlador central no esquema PCF. Tempo intermediário.

- DIFS (*Distributed Coordination Function IFS*): usado como delay mínimo para frames assíncronos. É o mais longo.

A Figura 2.9 ilustra o uso desses valores de tempo.

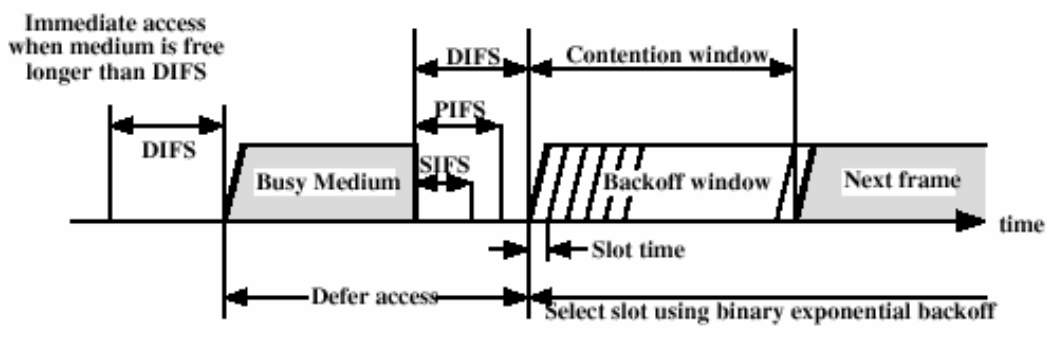

Figura 2.9 – Método de Acesso Básico

2.3.2. Função de Coordenação Pontual (PCF)

PCF é um método de acesso alternativo implementado no topo do DCF. Sua operação consiste na realização de eleições através do coordenador central, o qual utiliza PIFS para bloquear o tráfego assíncrono.

Uma vez que o PCF é utilizado para controle de estações com tráfego sensível ao tempo, pode correr que o tráfego assíncrono seja deliberadamente retardado por sucessivas eleições. Para evitar isso utiliza-se o conceito de *Superframe*, no qual o PCF atua durante a primeira metade e aguarda na metade seguinte.

A Figura 2.10 ilustra o uso do *Superframe.*

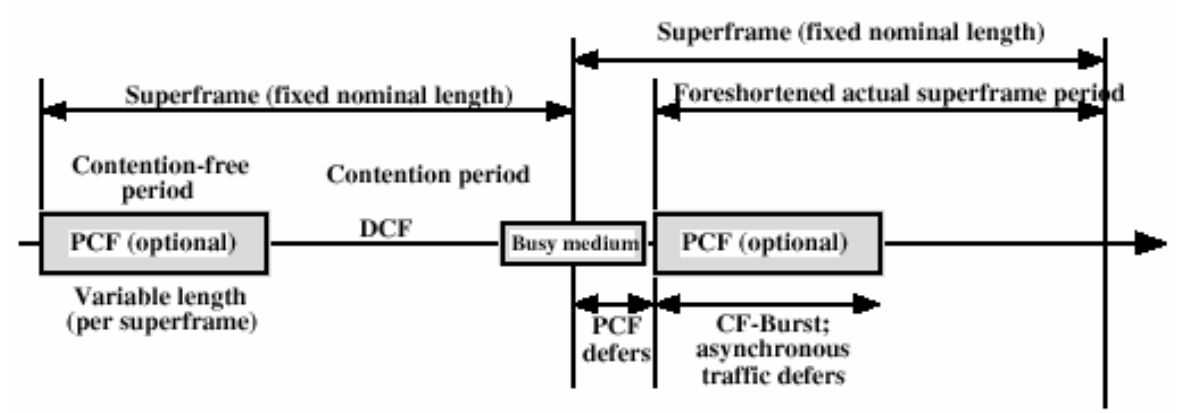

Figura 2.10 – Construção do Superframe PCF

2.4. OPERAÇÕES DE GERÊNCIA

As redes sem fio, de uma certa forma, herdam problemas das antigas redes e ainda os ampliam: meio é não confiável e de difícil monitoração, usuários não autorizados podem, mais facilmente, tirar vantagens da falta de fronteiras físicas definidas. Visando reduzir os efeitos destes problemas, os mecanismos de gestão do protocolo 802.11 foram melhor desenvolvidos.

2.4.1. Arquitetura de Gerência

Conceitualmente, a gestão da arquitetura 802.11 é composta de três componentes: a Entidade de Gerenciamento da Camada MAC (MLME – *MAC Layer Management Entity*), uma Entidade de Gerenciamento da Camada Física (PLME – *Physical Layer Management Entity*), e uma Entidade de Gerenciamento do Sistema (SME – *System Management System*). A relação entre essas diferentes entidades de gerenciamento, assim como as partes relacionadas ao protocolo 802.11 são visualizadas na Figura 2.11.

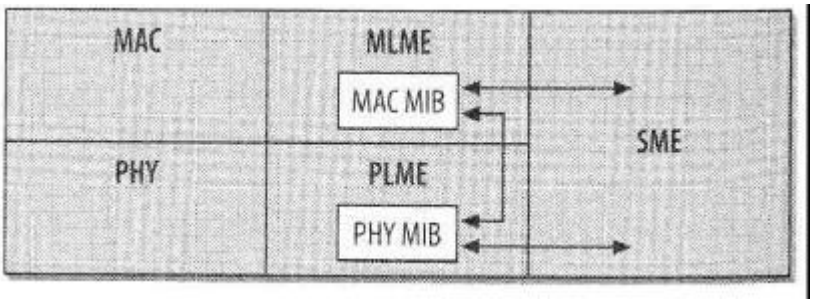

Figura 2.11 – Relação entre as entidades de gerência e os componentes do padrão 802.11

O protocolo 802.11 não especifica formalmente do SME. Resumidamente, o SME é o método pelo qual os usuários e dispositivos gerenciais interagem com a interface 802.11 e obtêm informações de seus *status*. Ambas as camadas MAC e PHY têm acesso a um Banco de Informações de Gerencimento (MIB – *Management Information Base*). O MIB tem objetos que podem ser instanciados para obter informações de *status*, assim como objetos que podem gerar ações específicas em um dado intervalo de tempo.

Existem três interfaces definidas entre os componentes de gerenciamento. A entidade de gerenciamento da estação pode alterar tanto a MIB da camada MAC como da camada PHY através das interfaces de serviços MLME e PLME. Adicionalmente, mudanças na MAC requerem mudanças correspondentes na PHY, assim, uma interface adicional entre o MLME e a PLME permite que a MAC faça alterações na PHY.

2.4.2. Scanning

Antes de usar qualquer rede é necessário, primeiramente, encontrá-la. No mundo *wireless*, as estações precisam identificar uma rede compatível antes de entrar. O processo de identificação de redes existentes é chamado de *scanning*. Para realizar tal processo diversos parâmetros são utilizados no procedimento, dentre eles *BSSType*, *BSSID*, *SSID*, *ScanType*, *ChannelList*, *ProbeDelay*, *MinChannelTime* e *MaxChannelTime.*

2.4.2.1. Scanning Passivo

Scanning Passivo economiza bateria visto que este não requer transmissão. No *scanning* passivo, a estação move de canal em canal, na lista de canais, e espera por quadros, *Beacon Frames*. Tais *Beacon Frames*, quadros de *broadcast* utilizados para gerência, são necessários para que a estação descubra tudo que precisa sobre os parâmetros ligados à BSS e inicie a comunicação. Na Figura 2.12, a estação móvel utiliza *scanning* passivo para achar BSSs na região em que se encontra.

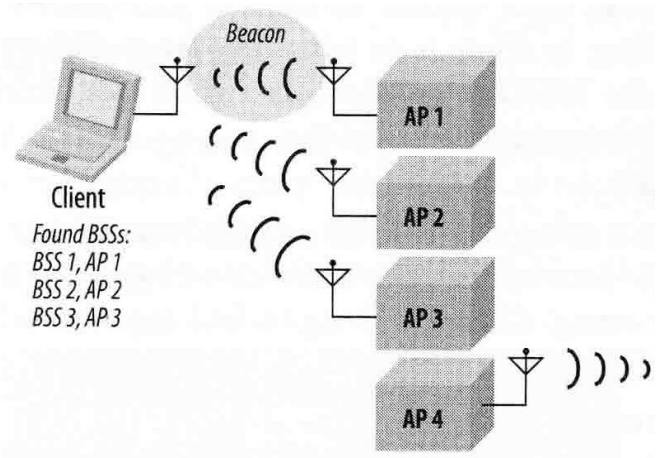

Figura 2.12 – Scanning Passivo

2.4.2.2. Scanning Ativo

No *scanning* ativo, a estação toma um papel mais decisivo. Em cada canal, quadros de *Probe Request*, requisições para teste, são utilizados para solicitar respostas de uma rede com um dado nome. Ao invés de escutar a rede para se anunciar, um *scanning* ativo objetiva encontrar a rede. Estações utilizando *scanning* ativo empregam o seguinte procedimento para cada canal na lista de canais:

1. Move para um dado canal e aguarda por tanto uma indicação de um quadro se aproximando como pela expiração do *ProbeDelay*, parâmetro da BSS que indica o atraso máximo antes de se iniciar o processo de sondagem do canal. Se a presença de um quadro se aproximando é detectada, o canal está em uso e pode ser, assim, sondado.

2. Ganha acesso ao meio usando o procedimento de acesso DCF básico, modo similar à Ethernet o qual primeiro checa se a conexão de rádio está livre antes de transmitir, e envia um quadro de *Probe Request*.

3. Espera que o tempo de canal, MinChannelTime — parâmetro da BSS que especifica o tempo mínimo para sondar o canal, esgote.

    a. Se em nenhum momento o meio esteve ocupado, não existia rede no canal. Mova para o próximo canal.

    b. Se o meio estava ocupado durante o intervalo do MinChannelTime, espera até o tempo máximo, MaxChannelTime, e processa os quadros de *Probe Response*, resposta de sondagem.

Na Figura 2.13 se verifica a relação entre a transmissão de quadros de *Probe* e os vários intervalos de tempo ligados ao procedimento de *scanning*.

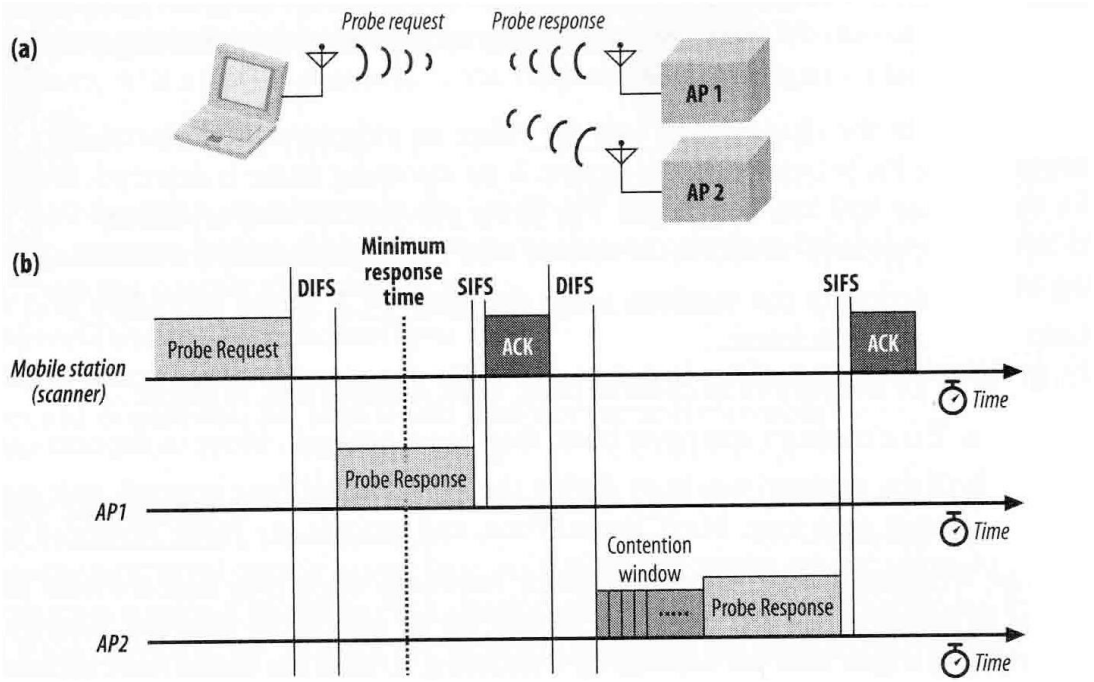

Figura 2.13 – Procedimento para scanning passivo e acesso ao meio

Na Figura 2.13(a), uma estação móvel transmite um *Probe Request* o qual dois *access points* respondem. Na Figura 2.13(b) é mostrada a atividade no meio. Ambos os *access points* respondem com um *Probe Response* que relata os parâmetros de suas respectivas redes. Vale ressaltar que o segundo *Probe Response* está sujeito às regras da Função de Coordenação Distribuída (DCF – *Distributed Coodination Function*) e precisa esperar por uma janela de congestão para transmitir. Como a primeira resposta é transmitida antes do tempo mínimo esgotar, a estação espera até que o tempo máximo se esgote para coletar todos os resultados e, assim, processá-los.

2.4.2.3. Relatório do *Scanning*

Um relatório do processo de *scanning* é gerado ao fim do mesmo. O relatório lista todas as BSSs que o *scan* descobriu assim com seus respectivos parâmetros. A lista completa dos parâmetros permite que a estação, a qual está realizando o *scanning*, se ligue a qualquer uma das redes descobertas.

2.4.2.4. Ligamento

Após compilar os resultados do *scanning*, uma estação pode eleger uma das BSSs para se ligar. Essa etapa de *joining,* ligamento, é precursora à associação e corresponde à escolha de um AP para se conectar. Ela não permite acesso à rede. BSSs que são parte de uma mesma ESS podem tomar decisões da forma que acharem mais conveniente, critérios como nível de potência e força do sinal são comumente utilizados. Observadores externos não podem dizer quando uma estação se ligou a uma rede porque este processo é interno ao nodo.

2.4.3. Autenticação

Em uma rede com fio, autenticação é implicitamente obtida com o acesso físico ao meio através da conexão de um cabo. Redes *wireless* são atrativas, em grande parte, porque o acesso físico não é requerido para usar os recursos da rede. Assim, um componente maior de segurança é garantir que as estações que estão tentando se associar com a rede têm essa permissão. Duas principais formas de autenticação são especificadas pelo 802.11: *open-system authentication*, autenticação de sistema aberto, e *shared-key authentication*, autenticação de chave compartilhada.

2.4.3.1. Autenticação de Sistema Aberto

Autenticação de sistema aberto é o único método requerido pelo padrão 802.11. Neste sistema, o *access point* aceita a estação de imediato, sem verificar a identidade. Para a efetivação desse processo, há a necessidade da troca de dois quadros, de acordo com a Figura 2.14.

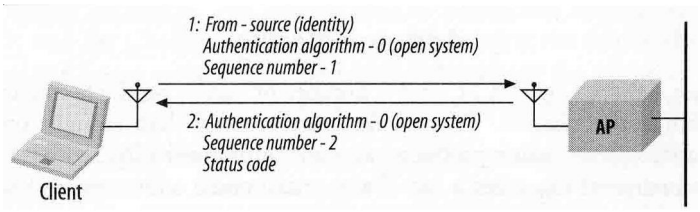

Figura 2.14 – Autenticação de sistema aberto

2.4.3.2. Autenticação de Chave Compartilhada

Autenticação de chave compartilhada faz uso do WEP, *Wired Equivalent Privacy*, e, portanto, só pode ser utilizado em produtos que implementam WEP. Além do mais, 802.11

requer que qualquer estação implementando WEP também implemente autenticação de chave compartilhada. Tal autenticação requer que a chave compartilhada seja distribuída entre as estações antes de se iniciar o processo de autenticação propriamente dito. Para a efetivação do processo de autenticação é necessário a troca de quatro quadros de acordo com a Figura 2.15.

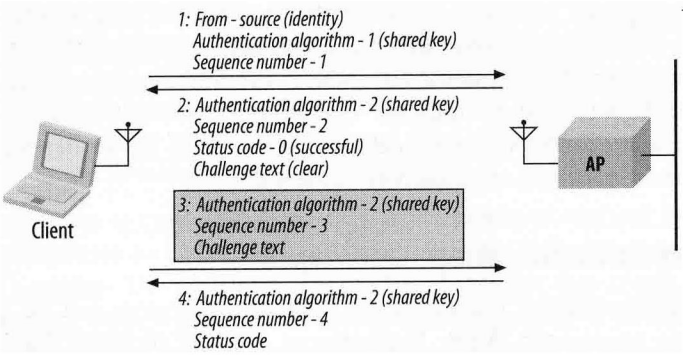

Figura 2.15 – Autenticação de chave compartilhada

2.4.3.3. Pré-Autenticação

As estações precisam se autenticar com um *access point* antes de associar a ele, mas nada no 802.11 requer que a autenticação aconteça imediatamente antes da associação. Estações podem se autenticar com diversos *access points* durante o processo de *scanning* de forma que quando a associação for necessária, a estação já está autenticada. Este processo é chamado de pré-autenticação. Como resultado da pré-autenticação, as estações podem reassociar com *access points* à medida que se movem pela área de cobertura, ao invés de esperar pela troca de autenticação.

Em ambas as partes da Figura 2.16 existe um conjunto de serviços estendidos, ESS, composto de dois *access points.* Assumindo que a estação inicia associada com o AP1, visto que foi ligada na área de cobertura do AP1, esta deve eventualmente se associar com AP2 à medida que se afasta da área de cobertura do AP1.

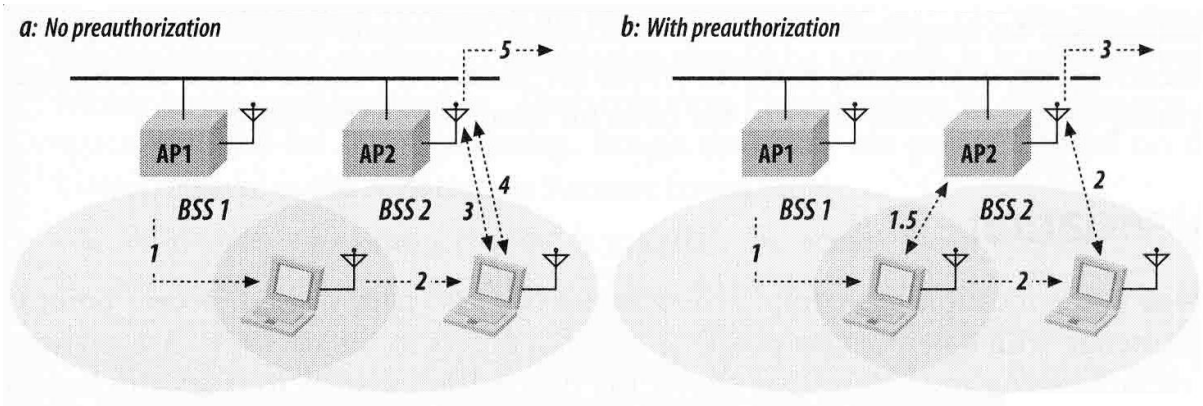

Figura 2.16 – Economia de tempo conseguida com a Pré-Autenticação

Como pode ser visualizado na Figura 2.16(a), a pré-autenticação não é utilizada na interpretação literal do 802.11. À medida que a estação se desloca para a direita, o sinal do AP1 enfraquece. A estação continua monitorando *Beacon Frames* relativos ao ESS que se encontra e, eventualmente, irá perceber a existência do AP2. Em algum ponto, a estação optará em se desassociar do AP1 e se autenticar e reassociar com o AP2. Estes passos são identificados na Figura 2.16 e detalhados na Tabela 2.3.

Tabela 2.3 – Cronologia da Figura 2.16

Passo	**Ação sem Pré-Autenticação (Figura 2-16a)**	**Ação com Pré-Autenticação (Figura 2-16b)**
0	Estação está associada com AP1	Estação está associada com AP1
1	Estação se move para a direita até a fronteira entre BSS1 e BSS2	Estação se move para a direita até a fronteira entre BSS1 e BSS2 e detecta a presença do AP2
1.5		Estação se pré-autentica com AP2
2	O sinal do AP2 é mais forte, assim, a estação decide se associar com AP2	O sinal do AP2 é mais forte, assim, a estação decide se associar com AP2
3	Estação se autentica com AP2	Estação começa a utilizar a rede
4	Estação reassocia com AP2	
5	Estação começa a utilizar a rede	

A Figura 2.16(b) mostra o que acontece quando a estação utiliza o recurso de pré-autenticação. Com esta pequena modificação de software, a estação pode se autenticar com AP2 assim que o detectar. À medida que a estação vai saindo da área de cobertura do AP1, esta fica autenticada em ambos AP's. Quando a estação deixa a área de cobertura do AP1 esta pode imediatamente se reassociar com AP2. Pré-autenticação torna o *roaming* uma operação mais suave. Todos os passos envolvidos na Figura 2.16(b) são detalhados na Tabela 2.3.

2.4.4. Associação

Uma vez que a autenticação se completou, estações podem se associar com um *access point* (ou se reassociar com um novo *access point*) para obter acesso completo à rede. Associação é um procedimento de memória que permite que o sistema de distribuição rastreie a localização de cada estação móvel, assim, quadros destinados para a estação móvel podem ser encaminhados para o *access point* correto. Um método de registrar é enviar um ARP, *Address Resolution Protocol*, gratuito para o endereço MAC das estações associadas com a porta de chaveamento do *access point*. Quando completado este procedimento, uma estação *wireless* pode usar o sistema de distribuição para alcançar toda a rede e a rede pode responder

através do sistema de distribuição. O padrão 802.11 proíbe explicitamente a associação com mais de um *access point*.

2.4.4.1. Procedimento de Associação

O procedimento básico de associação é mostrado na Figura 2.17.

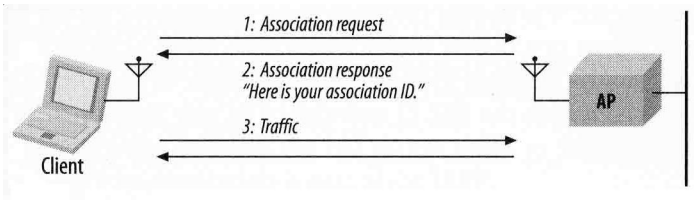

Figura 2.17 – Procedimento básico de associação

Assim como na autenticação, associação é iniciada pela estação móvel. Não são necessários números de seqüenciamento porque o processo de associação consiste em uma troca de três passos. Cada um desses passos pode ser detalhado da seguinte forma:

1. Uma vez autenticada, a estação pode enviar um quadro de requisição de associação, *Association Request Frame.*

2. O *access point* processa a requisição. 802.11 não especifica como determinar se uma associação deve ocorrer. Uma consideração comum é a quantidade de espaço requerido para *bufferizar* quadros. Assim, dada uma requisição, duas possíveis soluções são:

    a. Quando o acesso é garantido, o *access point* responde com um código de *status* 0 (sucesso) e o ID de Associação (AID – *Association ID*).

    b. Associações sem sucesso requerem apenas um código de *status* e o procedimento termina.

3. O *access point* começa a processar os quadros da estação móvel. Quando um *access point* recebe um quadro destinado para uma estação móvel associada, este quadro passa através de uma ponte Ethernet-Wireless ou pode ser armazenado se a estação estiver em modo de economia de energia.

2.4.4.2. Procedimento de Reassociação

Reassociação é o procedimento de mover a associação de um antigo *access point* para um novo. Pela interface aérea, este procedimento é quase o mesmo que uma associação; no *backbone* da rede, no entanto, *access points* precisam interagir entre si para encaminhar os quadros. Quando uma estação muda da área de cobertura de um *access point* para outro, esta utiliza o processo de reassociação para informar à rede 802.11 de sua nova localização. Este procedimento pode ser visualizado na Figura 2.18.

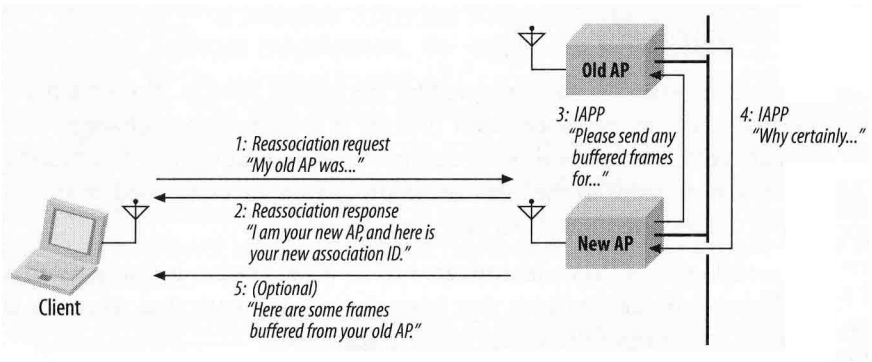

Figura 2.18 – Procedimento básico de reassociação

A estação começa o procedimento associada com um determinado *access point*. A estação monitora a qualidade do sinal que ela recebe deste *access point*, assim como a qualidade do sinal dos outros *access points* na mesma ESS. Quando a estação móvel detecta que outro *access point* seria uma escolha melhor, ela inicia o procedimento de reassociação. As etapas para esse procedimento podem ser vistas na Figura 2.18 e são detalhadas da seguinte forma:

1. A estação lança um *Reassociation Request*, requisição de reassociação, para o novo *access point*. *Reassociation Request* tem conteúdo similar ao do *Association Request*. A única diferença é que o *Reassociation Request* contem um campo com o endereço do antigo *access point*. O novo *access point* deve comunicar com o antigo para verificar que uma antiga associação realmente existia. Caso não, o novo *access point* responde com quadro de desautenticação e finaliza o processo.

2. O *access point* processa o *Reassociation Request*. A resposta é semelhante àquela resultante do *Association Request*.

3. O novo *access point* contata o antigo para finalizar o procedimento de reassociação.

4. O antigo *access point* envia quaisquer quadros *bufferizados*, da estação móvel em questão, para o novo *access point*.

5. O novo *access point* começa a processar os quadros da nova estação normalmente.

Reassociação também é utilizada para se religar a uma rede nos casos onde a estação deixa a área de cobertura e depois retorna ao mesmo *access point*. A Figura 2.19 ilustra essa situação.

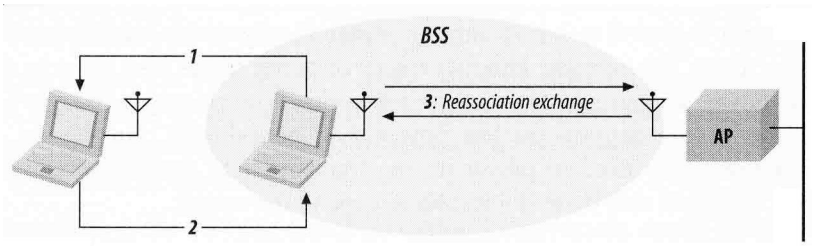

Figura 2.19 – Reassociação com o mesmo AP

3. A SOLUÇÃO WIRELESS VILLAGE

O Wireless Village é uma iniciativa da Nokia, Motorola e Ericsson que visa definir e promover especificações e padrões de forma a criar um protocolo universal para serviços IMPS entre workstations, servidores de aplicações, PDA´s e outros dispositivos móveis, que permita a interoperabilidade entre os diversos sistemas e fabricantes. A seguir são apresentados o conceito de IMPS e a arquitetura da solução Wireless Village, destacando os serviços de Presença e Mensagens Instantâneas.

3.1. IMPS (INSTANT MESSAGING AND PRESENCE SERVICES)

O conceito dos Serviços Baseados em Troca de Mensagens Instantâneas e Informações de Presença não é recente. Há alguns anos as trocas de mensagens instantâneas entre usuários de workstations, celulares e PDA´s já é utilizada. No entanto, os serviços de mensagens instantâneas atuais oferecem apenas a troca de mensagens de texto e multimídia, não estando disponíveis a troca de informações de presença.

O IMPS é composto de quatro serviços, independentes mas complementares:

- Presença;
- Troca de Mensagens Instantâneas;
- Grupos;
- Conteúdo Compartilhado.

O **Serviço de Presença** fornece informações acerca do status de um usuário, permitindo saber se o mesmo está on-line bem como informar a disponibilidade do mesmo para receber mensagens. Além disso, o serviço de presença oferece funcionalidade que permitem determinar a real localização de um usuário, além da vontade do mesmo em compartilhar essa informação.

O **Serviço de Troca de Mensagens Instantâneas** é o responsável pelo envio e recebimento de mensagens entre os usuários do sistema. O sistema permite o envio de diferentes tipos de conteúdo, tais como texto, vídeo, som e imagem.

O **Serviço de Grupos** permite que sejam criadas comunidades virtuais, públicas ou privadas, cuja gerência fica a cargo da operadora ou dos próprios usuários.

O **Serviço de Conteúdo Compartilhado** permite a troca de conteúdo/informação tais como imagens e documentos entre os usuários do sistema WV. Estas informações podem ser compartilhadas durante a troca de mensagens entre grupos ou por simples armazenamento compartilhado.

3.2. ARQUITETURA DO SISTEMA

A Figura 3.1 ilustra a Arquitetura do Sistema *Wireless Village*. A base do sistema é uma arquitetura cliente-servidor, onde o cliente pode ser um aparelho celular, um PDA ou um computador. A seguir são apresentados os principais componentes do sistema, descrevendo suas características e funcionalidades.

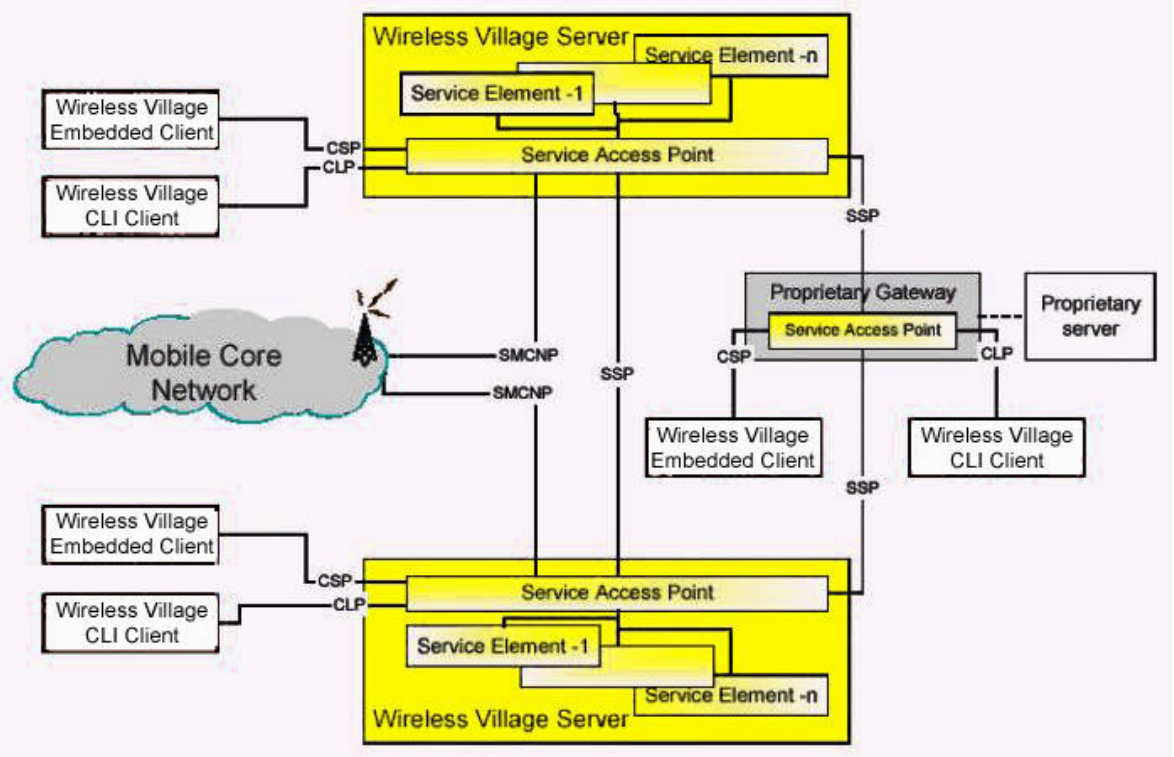

Figura 3.1 - Arquitetura do sistema Wireless Village

3.2.1. Wireless Village Server

O *Wireless Village Server* – WVS – é o ponto principal no projeto, é o ponto no qual se congregam todos os elementos que formam o sistema. A figura 3.2 apresenta um esquema do WVS, destacando seus elemento funcionais.

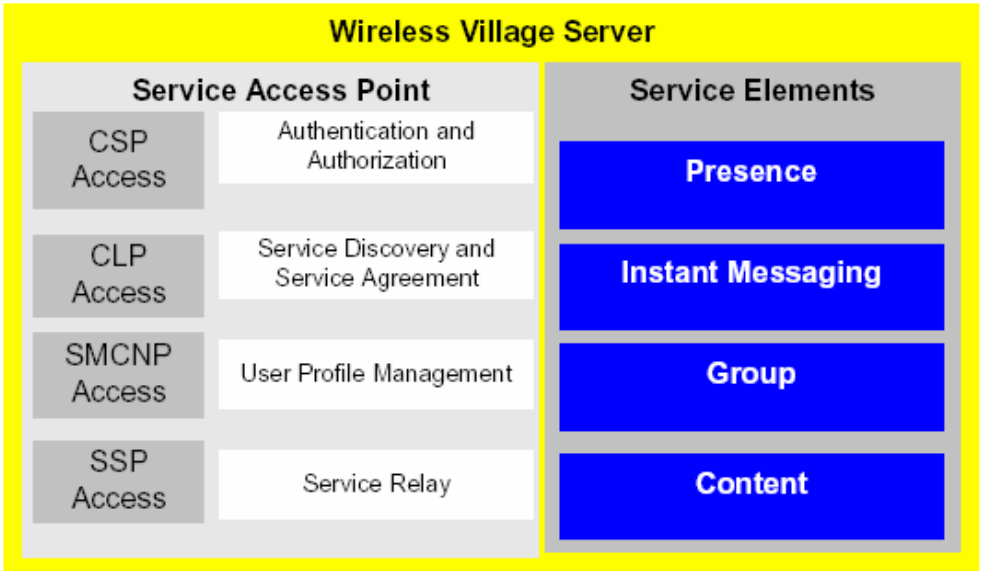

Figura 3.2 - Elementos funcionais do Wireless Village Server

O WV Server é composto de quatro módulos de serviço:

- ***Presence Service Element***: responsável por definir a localização do usuário em uma área urbana, assim como, armazenar toda a lista de contato deles, seus humores (*moods*) e estados, ou seja, guardar suas definições de presença. Estas informações podem ser manipuladas explicitamente pelo usuário ou implicitamente pelo sistema;

- ***Instant Messaging Service Element***: responsável pela troca de mensagens em formato SMS ou XML/HTTP entre os diferentes usuários do sistema. As mensagens podem ser endereçadas tanto para um usuário como para um grupo de usuários. Além disso, diferentes tipos de mensagens podem ser enviados tais como texto, imagens e som;

- ***Group Service Element***: provê habilidades para a manipulação de grupo de usuários que podem ser privados ou públicos. No fundo, um grupo nada mais é do que uma sala de chat;

- ***Shared Content Service Element***: permite a troca de conteúdo/informação entre os usuários do WV. O conteúdo compartilhado permite a troca de conteúdo entre usuários trocando mensagens ou em um grupo qualquer.

Para se implementar um WVS não é necessário a implementação dos quatro elementos de rede e uma implementação parcial já seria razoável. A Figura 3.3 mostra todos os serviços que cada módulo fornece.

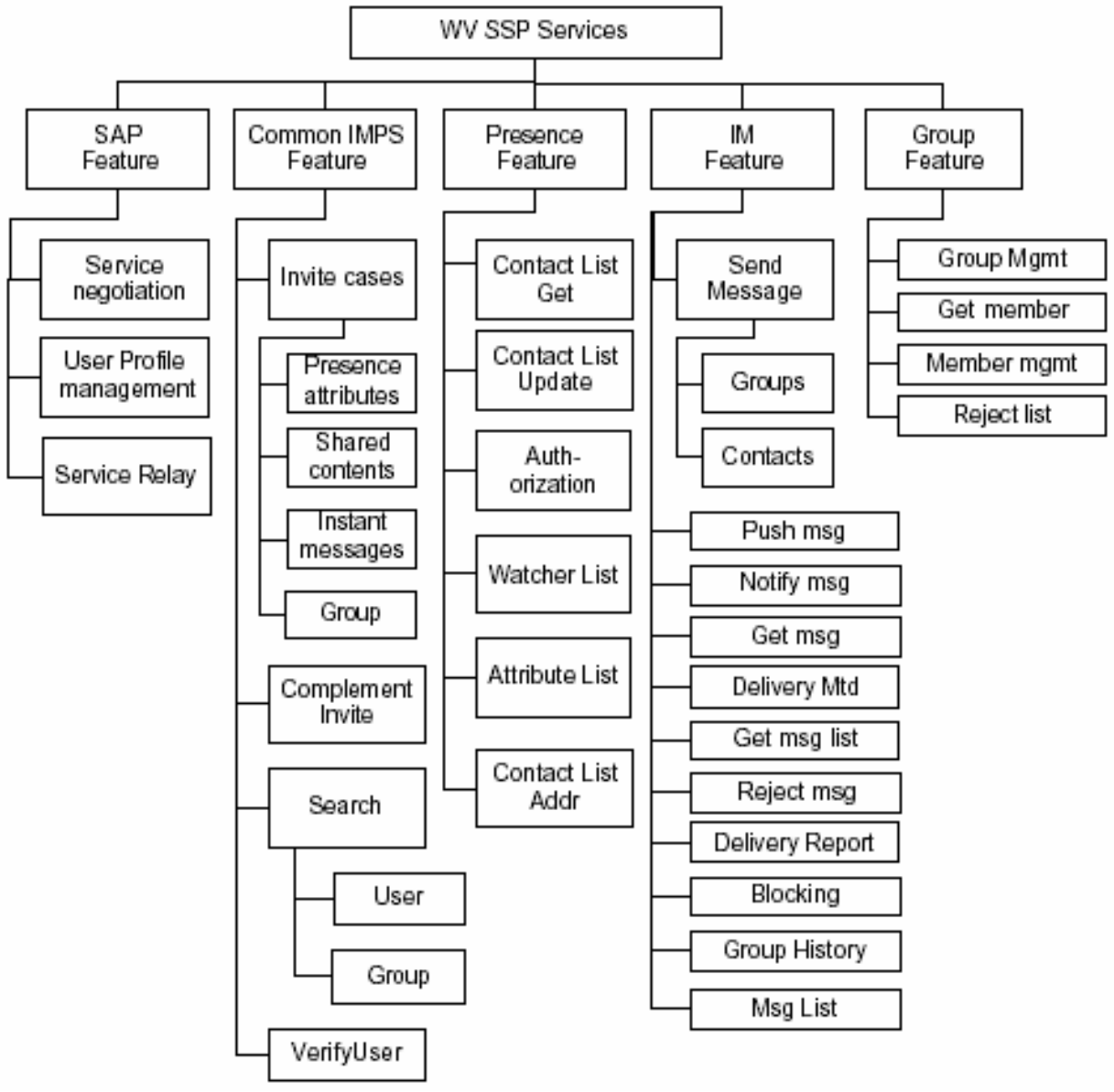

Figura 3.3 – Serviços prestados pelo Wireless Village Server

Além dos *Service Elements* acima, também é parte funcional do WVS o *Service Access Point* (SAP). O papel do SAP no sistema é fazer a interligação entre os quatro elementos listados e o restante da rede pertencente a operadora. Ele possui interface para todos os elementos da rede: o WV *Client*, outros WVS, a MCN e *Gateways* proprietários não-pertencentes ao WVS. As funções exercidas pelo SAP são:

- **Autenticação e Autorização**: o SAP faz a autenticação do usuário, garantindo que o usuário é quem diz ser. A autorização consiste em determinar o que um usuário já autenticado pode ou não fazer. Para tal é utilizado o CSP (*Client-Server Protocol*);

- **Descoberta de Serviços**: a descoberta de serviços informa à aplicação que tipo de serviços estarão disponíveis para seu uso Para tal é utilizado o CLP (*Command Line Protocol*);

- **Gerenciamento de perfis de usuários**: um ou mais perfis de usuários descrevem como o usuário deseja gerenciar e interagir com seus serviços. Para tal é utilizado o SMCNP (*Server-Mobile Core Network Protocol*);

- **Serviço de Relay**: um SAP deve ser capaz de rotear todo tipo de requisição e resposta de serviços entre os vários servidores. Para tal é utilizado o SSP (*Server-Server Protocol*).

3.2.2. WV Embedded Client e WV CLI Client

Um cliente WV é uma aplicação que está embutida em um dispositivo móvel. Clientes de diferentes fabricantes provavelmente terão um visual e funcionalidades diferentes. É possível, por exemplo, que as funcionalidades oferecidas pelo *Wireless Village* sejam integradas para funcionar junto com a agenda telefônica de um aparelho.

Outro tipo de cliente é o *CLI Client* (*Command Line Interface Client*). Esse cliente se comunicará com o servidor através do CLP (*Command Line Protocol*) com mensagens de texto. Um exemplo é um dispositivo que utiliza SMS para se comunicar com um servidor. Este tipo de cliente terá funcionalidade reduzida.

3.2.3. Interfaces e Protocolos

- ***Client Server Protocol* (CSP)**: Protocolo utilizado para comunicação entre o cliente e o servidor;

- ***Server Server Protocol* (SSP)**: Protocolo de comunicação entre os diversos WVS;

- ***Command Line Protocol* (CLP)**: Protocolo que permite terminais antigos acessarem o *Wireless Village Server* através de mensagens de texto;

- ***Server Mobile Core Network Protocol* (SMCNP)**: Protocolo que regula a comunicação entre o WV Server e o núcleo da rede móvel, para que o servidor tenha acesso a informações de presença.

3.3. O SERVIÇO DE PRESENÇA

O serviço de presença pode ser separado em três partes definidas, a saber: a definição da informação de presença de forma a garantir interoperabilidade, os mecanismos de autorização e transferência de informações de presença e o gerenciamento das listas de contatos.

Informação de presença não pode ser facilmente definida devido à vasta quantidade de informação disponível. Para garantir a interoperabilidade, foi definido um conjunto de atributos de presença, divididos em duas categorias: classes que guardam os Atributos de Status do Cliente e classes que guardam os Atributos de Status do Usuário.

Atributos relacionados ao status do cliente descrevem o status do software rodando no WV cliente, assim como, o hardware do dispositivo. Entre estes incluem-se atributos que descrevem o status do WV Client e seu dispositivo em relação a rede móvel, além de possuir informações mais detalhadas sobre o cliente como a sua versão e capacidades. O status na rede do cliente inclui o registro e o status online do cliente na rede, assim como, a localização do usuário e sua informação de endereço.

Atributos relacionados ao status do usuário incluem informações como a disponibilidade do usuário, suas informações de contato e a forma de contato preferida pelo usuário. Também incluem informações relacionadas à forma textual, status do conteúdo disponibilizado e o estado emocional do cliente, ou seja, seu humor de acordo com a escolha feita pelo usuário.

A **Lista de Contato** é uma lista, mantida pelo usuário, de outros usuários do WV. Ela pode ser utilizada como uma lista de distribuição para o envio de mensagens instantâneas ou informações de presença, como um mecanismo para se fazer autorização pró-ativa e para inúmeras outras funções.

Um usuário pode manipular diversas listas de contato com diferentes propósitos. O gerenciamento das listas de contato deve possuir possibilidades como criação, remoção e

modificação da lista de contatos, além de ser possível a obtenção de uma lista de contatos de outro usuário. Usuários também podem modificar o conteúdo de uma lista de contatos e recuperar o conteúdo de uma lista de contatos a partir do Servidor de Presença.

A **Autorização** de atributos de presença é dividida em *autorização pró-ativa* no qual o usuário WV autoriza os atributos de presença antes que qualquer pessoa tenha feito um pedido por estes atributos, e *autorização reativa* no qual o usuário confere autorização mediante um pedido expresso. Na autorização pró-ativa, os atributos que podem ser enviados são definidos pela lista de atributos. Na autorização reativa, o usuário pedindo a informação declara quais atributos ele deseja receber.

Na autorização pró-ativa, o *Publisher*[2] cria sua lista de contatos e adiciona usuários manualmente na mesma, autorizando pró-ativamente os membros dessa lista a acessar as informações de presença contidas na lista de atributos padrão, a qual também é definida pelo *Publisher*. Ele também pode especificar um conjunto de atributos de presença que serão associados para cada usuário em uma lista, mesmo que a lista possua seus próprios atributos. Os atributos do usuário sempre sobrescreverão os atributos da lista de contato. Quando um usuário está em diferentes listas de contatos com atributos diferentes, então o usuário tem direito ao conjunto de atributos contidos em todas listas.

O *Publisher* gerencia todo esse processo a partir de funções do tipo *create Contact List*, *delete Contact List*, *get contact list*, *add contact list member* e *remove contact list member*. Outras funções de gerenciamento incluem o *update* de atributos de presença de uma lista de contatos ou usuário, assim como, a remoção de atributos de presença de um usuário. Outras funções de suporte também são disponibilizadas, tais como *create attribute lists, delete attribute lists, get attribute lists, update attribute lists* e *attach and/or detach attribute lists to users and/or contact lists.*

Na autorização reativa, o usuário, utilizando o *Instant Messaging* – IM, pode fazer o pedido de alguns ou todos os atributos de outro usuário. O serviço de presença envia um pedido de autorização para o usuário IM e caso este seja autorizado, coloca-o na *Watcher List* ou lista de observação. Não existe uma autorização parcial para a lista de atributos e caso nenhuma lista seja criada no pedido, então se envia a *Default Public Attribute List* ao usuário que fez o pedido.

[2] O termo *Publisher* refere-se ao usuário que cria e gerencia sua própria Lista de Contatos e será utilizado na sua forma original no decorrer do texto.

A Lista de Observação é uma lista de usuários, definida pelo sistema para cada usuário, com a função limitada de guardar os usuários que foram autorizados a receber informações de presença deste usuário. Ser parte desta lista não garante que a autorização é válida ou que o usuário pedindo o acesso tenha acesso a qualquer informação de presença. Observa-se que cada usuário deve ter uma Lista de Observação no servidor. O *Publisher* tem direito de receber sua Lista de Observação com as informações de quem está inscrita nela a qualquer momento, não importando se é uma autorização pró-ativa ou reativa.

Tanto a autorização pró-ativa quanto a reativa podem ser canceladas pelo *Publisher* a qualquer momento. Um usuário inscrito também pode pedir para ser removido de uma lista a qualquer momento, ou seja, a inscrição deste usuário seria cancelada e seu nome removido da Lista de Observação.

A **Transferência** de informações de presença se dá no momento em que o usuário entra na Lista de Observação ou a cada vez que essa informação é atualizada. Cada usuário é capaz de modificar seus atributos de presença, tanto implicitamente (e.g. mudando sua localização) quanto explicitamente (e.g. alterando sua disponibilidade).

3.4. O SERVIÇO DE TROCA DE MENSAGENS INSTANTÂNEAS (*INSTANT MESSAGING*)

O serviço de troca de mensagens instantâneas é o responsável por prover mecanismos de criação e recebimento de mensagens, bem como fornecer relatórios de status da entrega das mesmas. Dois métodos de recebimento são suportados: o mecanismo de entrega direta (*Push*) e o método de notificação e busca (*Notification/Pull*). O destinatário de uma mensagem pode ser um ou mais usuários individuais, um grupo, ou usuários dentro de uma Contact List.

No tipo de entrega direta, o *Instant Messaging Service Element* envia a mensagem ao destinatário logo que ela chega. Este método de entrega é recomendado para mensagens de texto pequenas ou urgentes. Já no tipo *Notification/Pull*, a mensagem fica no servidor, e este manda apenas uma notificação de nova mensagem para o destinatário.

Na hora do envio de uma mensagem, pode ocorrer que o destinatário esteja off-line, não logado no sistema WV. Isso pode ser indicado por um atributo de presença, que o usuário pode verificar antes de enviar uma mensagem. O servidor pode implementar uma

funcionalidade de *store-and-forward*. Não sendo este o caso, a mensagem é simplesmente descartada. Caso seja, a mensagem fica armazenada no servidor por um certo período de tempo esperando o login do destinatário. Quando o login acontece, o servidor pode escolher entre entregar as mensagens uma a uma ou esperar que o usuário peça uma lista de mensagens não entregues.

A especificação prevê ainda a opção de bloquear mensagens utilizando-se de uma lista de bloqueio assim como também podem ter uma lista de usuários permitidos. Tais funcionalidades ainda não estão disponíveis.

O *Wireless Village* prevê que a mensagem tenha qualquer tipo de conteúdo, incluindo conteúdo multimídia. No entanto, o projeto disponibiliza apenas o envio de mensagens de texto. Futuras implementações devem ser capazes de suportar diferentes tipos de conteúdo.

Uma análise mais profunda sobre essa parte do sistema pode ser encontrada na referência [5].

3.5. O SERVIÇO DE GRUPO

O serviço de grupo de usuários lida com o conceito da criação de fóruns de discussão no estilo de salas de bate-papo formados por um conjunto de indivíduo usuários do sistema WV e até mesmo pelo provedor do serviço em si. O serviço de grupo é um componente básico do sistema, o qual permite a criação de comunidades de usuários de serviços IMPS com a finalidade de trocar informações sobre os mais diversos assuntos.

As mensagens de e para cada um dos grupos são geradas pelo serviço de troca de mensagens instantâneas descrito anteriormente. A grande diferença entre mensagens em grupo e mensagens ponto-a-ponto é que o grupo age como um mecanismo de distribuição de mensagens e, conseqüentemente, um usuário que deseja receber mensagens de um grupo e tomar parte na discussão deve se unir ao grupo.

Assim, o serviço de grupo deve oferecer formar que facilitem a criação e gerenciamento de grupos. Essas características, tal como descritas na solução WV são as seguintes: Criação e exclusão de grupos públicos e privados; criação de lista de usuários permitidos e banidos; Funções de unir-se, abandonar, convidar e procurar grupos e usuários;

Funções de modificação de propriedades do grupo, gerenciamento de listas de usuários e controle de acesso.

3.6. O SERVIÇO DE CONTEÚDO COMPARTILHADO

O serviço de conteúdo compartilhado permite aos usuários dos serviços IMPS compartilhar informações tais como imagens e documentos, ao mesmo tempo em que permite a troca de mensagens ou o bate-papo em um grupo. O compartilhamento de informações se dará através de uma função de convite geral, pela qual os usuários poderão trocar a URL do conteúdo que desejam compartilhar.

3.7. SERVIÇOS DE ACESSO

Todo cliente WV é obrigado a fazer sua autenticação e autorização em um servidor WV para poder utilizar os serviços e nesta etapa ocorre a criação de uma sessão. O término da sessão pode ser dar de ambos os lados: o usuário pode realizar o *logout* ou o servidor WV pode desconectar-se independentemente da vontade do usuário.

Quando um cliente faz o *login* num servidor WV, ocorre a chamada negociação de capacidades e serviços. Na negociação de capacidades, o cliente WV indica ao servidor suas capacidades. A comunicação é adaptada e o conteúdo transmitido de acordo com estas capacidades, alguns termos negociados são: melhor método de envio (push/pull), tipos de conteúdo aceito, cifragem de transferência aceita, tamanho máximo de conteúdo aceito e *bindings* de transporte suportados.

Na fase de negociação, o cliente WV indica as funções que ele planeja usar durante a sessão e as quais o servidor WV precisa suportar. O servidor responde indicando as funções que ele aceita em suportar baseado na disponibilidade e perfil do usuário.

Outro fator importante é o envio de uma mensagem do tipo *Keep Alive* regulada por um timer quando não houver comunicação durante a sessão. O propósito desta mensagem é indicar para o servidor WV que o cliente está *online* e pronto para se comunicar.

Um outro serviço de acesso disponível é o de informações sobre o provedor de serviços. Através deste serviço, uma mensagem é enviada ao celular do cliente contendo o

provedor do serviço WV, ou seja, seu nome, logotipo, texto descritivo e um URL. Este serviço é utilizado somente para a diferenciação da marca dos provedores de serviço.

3.8. SERVIÇOS COMUNS

Serviço de procura geral. Este serviço é utilizado para que um usuário localize outros usuários ou informações contidas em um serviço WV. Por enquanto, a procura será limitada a usuário e grupos. Os critérios que podem ser utilizados pela busca são o username, o primeiro nome do usuário, seu sobrenome, seu email ou seu alias. Grupos podem ser procurados se utilizando o id do grupo, seu nome ou tópico. Também é possível se procurar grupos que foram criados por um determinado usuário ou grupos no qual determinado usuário esteja cadastrado.

Serviço de Convite geral. O serviço de convite geral permite que um cliente WV convide um ou mais usuários para atividades como: *chat* em grupo, troca de informações de presença e troca conteúdo compartilhado. No pedido de convite, o usuário pode escrever o motivo para este convite e o receptor deve responder se aceita ou não o convite, assim como mandar um texto explicando sua escolha. O usuário que fez o pedido também pode cancelá-lo caso mude de idéia com o tempo. Cada uma dessas habilidades é especificada como um tipo de mensagem contendo um código que identifica os erros, assim como, os diferentes parâmetros utilizados para seu pedido e resposta. As variáveis e as primitivas utilizadas em cada transação.

3.9. MECANISMOS DE TRANSPORTE DE MENSAGENS ENTRE UM CLIENTE E UM SERVIDOR WV (CSP TRANSPORT BINDING)

O Wireless Village especifica quatro diferentes tipos de transporte que podem ser utilizados para transportar mensagens no CSP. São eles o WSP, HTTP, HTTPS e SMS. A Figura 3.4 mostra um modelo lógico de comunicação entre o WV Client e o WV Server:

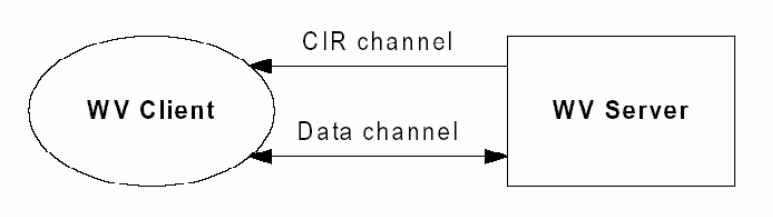

Figura 3.4 – Modelo lógico de comunicação

Neste modelo de comunicação, o canal CIR tem o propósito de estabelecer um canal de comunicação entre as entidades Cliente WV e Servidor WV.

Como os protocolos WSP, HTTP e HTTPS são assimétricos, isto é, a comunicação sempre se origina do cliente para o servidor, é preciso haver adaptações na hora em que o servidor tiver que iniciar uma transação. O início de transação pelo servidor pode ser feito das seguintes formas:

- O servidor insere sua requisição dentro de uma mensagem de resposta pendente do cliente para o servidor;
- O servidor utiliza o canal CIR para enviar um pedido imediato de polling para o cliente, e então o servidor manda sua requisição na parte de resposta da mensagem de polling;
- Na ausência de um canal CIR, é preciso que o cliente realize um polling (envio de mensagens periodicamente) para que a requisição do servidor possa ir na resposta da mensagem do polling;

Se o SMS for utilizado, não há necessidade de um canal CIR, já que ambos, cliente e servidor, podem originar transações e o canal está sempre disponível.

Para o nosso projeto, optamos por utilizar a terceira opção, o polling, já que para criar um canal CIR teríamos que implementar um "servidor" no lado do Cliente, que pudesse receber mensagens TCP ou UDP.

4. TECNOLOGIAS UTILIZADAS NA IMPLEMENTAÇÃO

Este capítulo trata das tecnologias utilizadas na confecção do Ambiente de Desenvolvimento de Serviços Baseados em Presença. Tais tecnologias constituem a base para o desenvolvimento de quaisquer aplicações ou serviços *wireless* e, portanto, o entendimento do conceito de tais aplicações está intimamente ligado ao entendimento das características principais dessas tecnologias. A escolha dessas tecnologias também foi influenciada devido ao fato de serem todas tecnologias e padrões *open source*. Esta foi a filosofia de desenvolvimento escolhida para a criação deste trabalho. As tecnologias utilizadas foram: Java 2 Platform, Micro Edition (J2ME); Servlets; Conceitos da Linguagem XML; e O Banco de Dados MySQL.

4.1. JAVA 2 PLATFORM, MICRO EDITION (J2ME)

O *Java 2, Micro Edition* (J2ME) é a ferramenta que será utilizada para o desenvolvimento da parte Cliente do ambiente de desenvolvimento de serviços. Esta é a parte que realiza toda a interface com o usuário e, conseqüentemente, deve ser capaz de lidar com as limitações existentes na maioria dos dispositivos móveis.

A plataforma J2ME oferece o poder e os benefícios da linguagem Java (J2SE – *Java 2, Standard Edition*) só que para os dispositivos móveis – incluindo uma interface gráfica flexível, modelo de segurança robusto, grande quantidade de protocolos de transmissão embutidos, e suporte para aplicativos usados pela rede ou localmente. Portanto, com o J2ME, os aplicativos podem ser escritos somente uma vez para uma grande quantidade de dispositivos móveis, podendo ser baixados dinamicamente da Internet e utilizando toda a capacidade própria dos dispositivos.

Numa visão geral, o J2ME pode ser descrito como sendo uma plataforma que define:

- Uma série de Máquinas Virtuais Java (JVM), própria para cada dispositivo, e respeitando suas limitações e requerimentos;
- Um grupo de bibliotecas e API’s, rodando sobre as JVM, conhecidas como Configurações e Perfis (*Configurations and Profiles*);

- Várias ferramentas para configuração e carregamento de *software* nos dispositivos.

O conjunto formado pelas Máquinas Virtuais Java e as Configurações e Perfis constituem o chamado ambiente *runtime*[3] J2ME. A Figura 4.1 apresenta a arquitetura J2ME com as configurações e perfis utilizados de acordo com o dispositivo móvel para o qual o aplicativo está sendo desenvolvido.

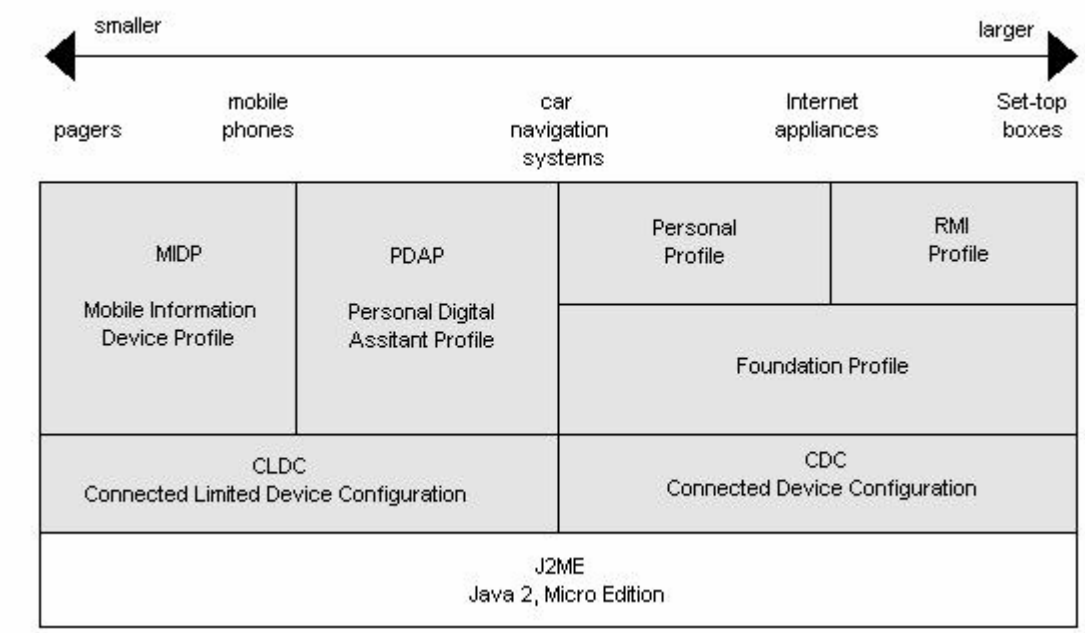

Figura 4.1 – Arquitetura J2ME segundo os dispositivos móveis utilizados

Conforme visto na figura acima, a configuração mais apropriada dispositivos móveis do tipo telefones celulares e *pagers* é a *Connected Limited Device Configuration* (CLDC) que foi desenvolvida para dispositivos com 160 a 512 KB de memória disponível para a plataforma Java, alimentados por bateria e com conexão intermitente à uma rede e com banda limitada. Utilizando os recursos do CLDC está o perfil *Mobile Information Device Profile* (MIDP) que especifica um conjunto de API's para uso em dispositivos móveis. O conjunto CLDC + MIDP é o que será utilizado para o desenvolvimento do Cliente em nosso sistema. A seguir temos uma explanação mais detalhada da arquitetura J2ME, com enfoque no conjunto utilizado.

[3] Esse termo será muito utilizado para definir um ambiente sendo executado em tempo real.

4.1.1. A Configuração CLDC

Configurações são compostas de uma máquina virtual e um conjunto mínimo de bibliotecas de classes. Elas provêem a funcionalidade básica para um particular conjunto de dispositivos com uma mesma característica comum, como conectividade a uma rede e pequena quantidade de memória.

As bibliotecas das API's CLDC podem ser dividas nas seguintes categorias:

- **Classes que são um subconjunto das APIs J2SE**: Essas classes são localizadas nos pacotes `java.lang`, `java.io` e `java.util` e são derivadas das API's J2SE;
- **Classes específicas ao CLDC**: Essas classes estão localizadas no pacote `javax.microedition` e em seus subpacotes;
- **Classes herdadas**: O CLDC herda um número de classes do sistema, de entrada/saída e de *utilities* da plataforma J2SE.

A Tabela 4.1 mostra as classes herdadas da plataforma J2SE e a Tabela 4.2 mostras a classes herdadas do J2SE e que tratam exceções e erros.

Tabela 4.1 – Classes herdadas da plataforma J2SE

Pacote	Classes
`java.lang`	`Boolean, Byte, Character, Class, Integer, Long, Math, Object, Winnable, Runtime, Short, String, StringBuffer, System, Thread, Throwable`
`java.io`	`ByteArrayInputStream, ByteArrayOutputStream, DataInput, DataOutput, DataInputStream, DataOutputStream, InputStream, OutputStream, InputStreamReader, OutputStreamWriter, PrintStream, Reader, Writer`
`java.util`	`Calendar, Date, Enumeration, Hashtable, Random, Stack, TimeZone, Vector`

Tabela 4.2 – Classes herdadas da plataforma J2SE e que tratam exceções e erros

Pacote	***Classes***
`java.lang`	`ArithmeticException, ArrayIndexOutOfBoundException, ArrayStoreException, ClassCastException, ClassNotFoundException, Error, Exception, IllegalAccessException, IllegalArgumentException, IllegalMonitorSateException, IllegalThreadstateException, IndexOutOfBoundException, InstantiationException, InterruptedException, OutOfMemoryError, NegativeArraySizeException, NumberFormatException, NullPointerException, RuntimeException, SecurityException, StringIndexOutOfBoundException, VirtualMachineException`
`java.io`	`EOFException, IOException, InterruptedException, UnsupportedEncodingException, UTFDataFormatException`
`java.util`	`EmptyStackException, NotSuchElementException`

Os pacotes básicos que compõem o CLDC podem ser vistas na Figura 4.2.

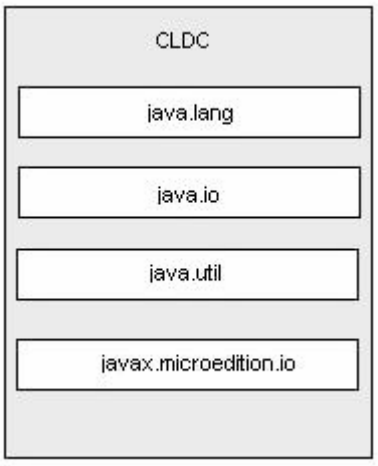

Figura 4.2 – Pacotes que compõem o CLDC

O CLDC possui uma série de limitações que são impostas pela Máquina Virtual Java que o suporta (KVM). Tais limitações ocorrem pois sua implementação ou era muito cara ou causaria um problema de segurança. Estas são as limitações impostas ao CLDC:

- Nenhum suporte a **números reais**;
- Sem suporte à **finalização** para limpeza de recursos;
- **Correção de erros limitada** aos erros em *Runtime***;**
- Sem suporte a **Java Native Interface (JNI);**
- Sem suporte ao ***Classloading***[4] para o usuário;
- Sem suporte à classe ***Reflection***;
- Sem suporte a grupos de ***threads*** ou ***threads daemon***.

Apesar de tais limitações, o CLDC é a configuração mais utilizada no desenvolvimento de aplicativos para dispositivos móveis, juntamente com o perfil MIDP, visto a seguir.

4.1.2. O Perfil MIDP

Perfis são mais específicos que configurações. Um perfil é baseado em uma configuração e adiciona API's para a interface do usuário, armazenamento persistente, e qualquer outra coisa que seja necessário para o desenvolvimento de aplicativos executáveis.

Aplicativos MIDP fornecem a fundação para aplicativos com uma interface gráfica avançada e intuitiva. A interface gráfica do usuário é otimizada para um display de tamanho pequeno, métodos de entrada específicos e outras funcionalidades nativas de dispositivos móveis. O MIDP fornece uma navegação intuitiva e entrada de dados a partir de um total mapeamento das teclas do telefone. Os aplicativos MIDP são instalados e executados localmente, podendo operar em modo desconectado ou através de uma rede e têm a habilidade para armazenar e gerenciar seguramente toda informação local. A Figura 4.3 indica os principais pacotes adicionados pelo MIDP.

[4] *Classloading* é a ação que ocorre quando o ambiente Java executa uma tarefa devido a uma nova classe ser carregada em tempo real – *runtime* – para ser utilizada.

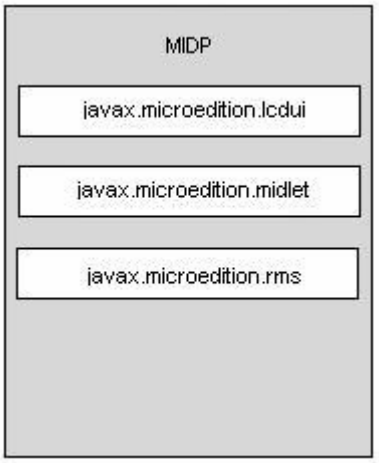

Figura 4.3 – Principais pacotes que compõem o MIDP

O MIDP oferece uma série de funcionalidades as quais são imprescindíveis para o programador em alto nível. Através destas funcionalidades, o perfil MIDP possibilita ao desenvolvedor a confecção dos mais variados tipos de software para dispositivos móveis, aliando facilidade, conectividade e segurança. São elas:

- **Interface gráfica de alto nível**: As funcionalidades desta interface gráfica incluem classes pré-definidas para mostrar e selecionar listas, imagens, editar texto, mostrar diálogos de alerta e adicionar barras de rolagem, tudo isso em *forms*[5] que podem incluir qualquer número de itens pré-definidos;

- **Funcionalidades para jogos e multimídia**: o MIDP disponibiliza bibliotecas para o gerenciamento de gráficos, som, vídeo e outros tipos de multimídia;

- **Conectividade extensiva:** suporta padrões líderes de transmissão de dados, incluindo HTTP, HTTPS, datagramas, *sockets*, *Server sockets* e comunicação serial. MIDP também suporta *Short Message Service* (SMS) e *Cell Broadcast Service* (CBS) de redes GSM ou CDMA utilizando a *Wireless Messaging* API (WMA), pacote opcional do MIDP;

- **Provisionamento Over-the-Air:** habilidade de instalar e atualizar aplicativos dinamicamente *over-the-air* (OTA);

[5] Classe Java utilizada pelo MIDP para gerenciar os objetos gráficos no *display*.

- **Segurança fim-a-fim:** provê um modelo de segurança robusto – construído através de padrões abertos – que protege a rede, os aplicativos e os dispositivos móveis. O uso de HTTPS maximiza padrões tais como o SSL e o WTLS para permitir a transmissão de informação cifrada.

A Tabela 4.3 apresenta todos os pacotes que compõem o MIDP, juntamente com uma breve descrição de cada um deles.

Tabela 4.3 – Pacotes que compõem o MIDP.

Pacotes de Interface do Usuário	
`javax.microedition.lcdui`	Define as funcionalidades para implementação de uma interface gráfica para aplicativos MIDP.
`javax.microedition.lcdui.game`	O pacote da API de jogos fornece um conjunto de classes que permite o desenvolvimento de um conteúdo rico para jogos em ambiente *wireless*.
Pacotes de Tempo de vida dos Aplicativos	
`javax.microedition.midlet`	O pacote MIDlet define as interações entre os aplicativos e o ambiente no qual eles são executados.
Pacotes de persistência	
`javax.microedition.rms`	O MIDP fornece um mecanismo para que os MIDlets armazenem e busquem informações persistentemente.
Pacotes de conexão	
`javax.microedition.io`	O perfil MIDP inclui suporte a *networking* baseado no *Generic Connection Framework* do CLDC.
Pacotes de chave pública	
`javax.microedition.pki`	Certificados são utilizados para autenticar informação de conexões seguras.
Pacotes de Som e Tons	
`javax.microedition.media`	A Media API do MIDP 2.0 é um bloco compatível com a MMAPI (JSR 135).
`javax.microedition.media.control`	Esse pacote define os tipos de Controle específicos que podem ser usados pelo *Player*.
Pacotes centrais	
`java.lang`	Classes MID da linguagem incluídas a partir do J2SE.
`java.util`	Classes MID de utilitários incluídas a partir do J2SE.

Aplicativos MIDP são representados por instâncias da classe `javax.microedition.midlet.MIDlet` a qual contém o *framework* para o desenvolvimento dos MIDlets.Cada MIDlet tem um ciclo de vida específico que é refletido nos métodos e comportamentos da classe MIDlet, a qual fornece dispositivos para começar, terminar, pausar e destruir aplicativos MIDlet no ambiente Java. O ciclo de vida do MIDlet é explicado a seguir e é mostrado na Figura 4.4.

1. Quando o MIDlet está quase pronto para ser executado, uma instância é criada. O construtor do MIDlet é executado e o MIDlet está no estado *Paused*;

2. A seguir, o MIDlet entra no estado *Active* depois que o gerenciador do aplicativo chama `startApp();`

3. Enquanto que o MIDlet está em *Active,* o gerenciador de aplicativos pode suspender sua execução ao chamar `pauseApp()`colocando o MIDlet de volta ao estado *Paused*. Um MIDlet pode colocar si mesmo no estado *Paused* ao chamar `notifyPaused();`

4. O gerenciador de aplicativos pode terminar a execução do MIDlet ao chamar `destroyApp()` no qual o MIDlet entra no estado *Destroyed* e pacientemente espera pelo coletor de lixo. Um MIDlet pode destruir a si ao chamar `notifyDestroyed()`.

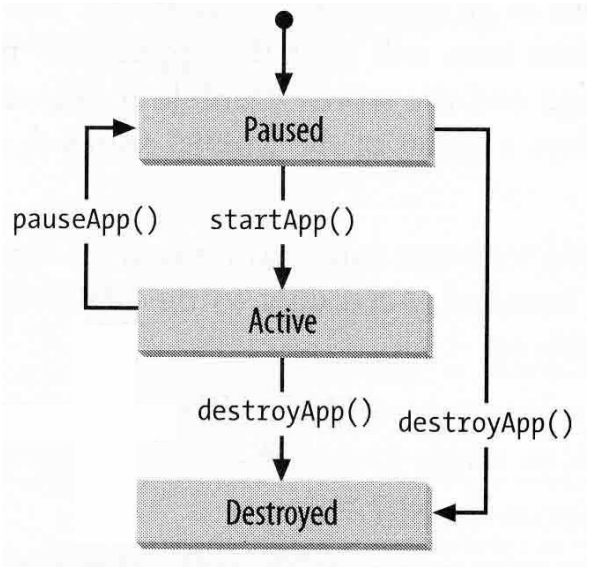

Figura 4.4 – Ciclo de vida do MIDlet

No que diz respeito ao interfaceamento com redes de comunicação, deseja-se escrever MIDlets que sejam capazes de rodar pela rede e que sejam suportados por todos dispositivos com MIDP. Além disso, os detalhes de baixo nível das redes e protocolos devem ser transparentes para a aplicação. Assim, o CLDC fornece uma abstração independente dos diferentes tipos de conexão. Essa camada de abstração é conhecida como *Generic Connection Framework* (GCF). O MIDP estende essa abstração e fornece a implementação de algumas interfaces provenientes desse *framework*. A Figura 4.5 retrata a interface `HTTPConnection`, disponibilizada pelo MIDP e utilizada para a toda a comunicação neste trabalho.

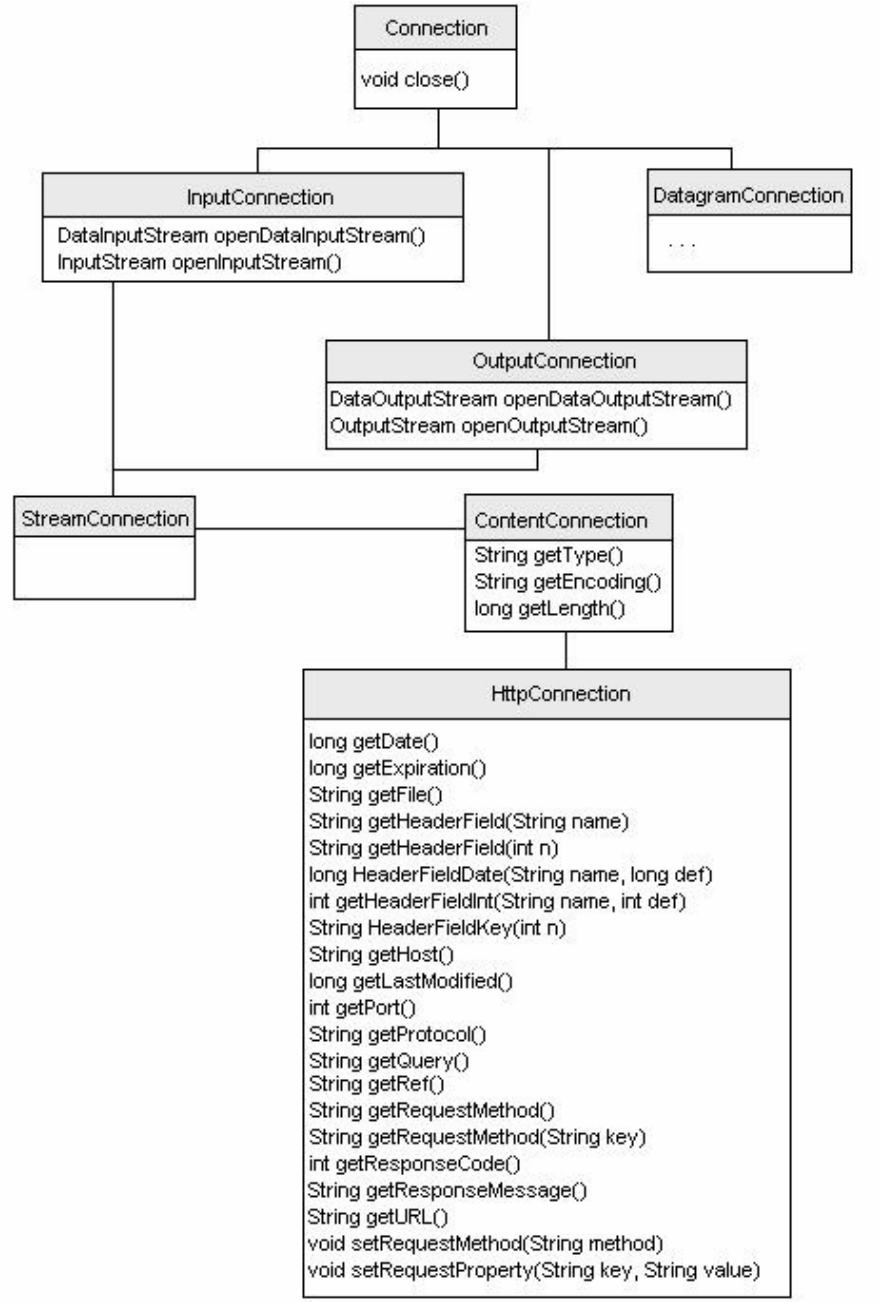

Figura 4.5 – Diagrama de classes para `HTTPConnection`

Outra classe importante que o GCF provê é a classe `Connector`. A interface acima não provê um método aberto para se estabelecer uma conexão. Assim, a classe `Connector` provê métodos estáticos que permitem que um aplicativo estabeleça uma conexão, independentemente do tipo ou protocolo utilizado na mesma. A sintaxe utilizada para abrir uma conexão é:

```
Connector.open (“<protocol>:<address>;<parameters>”);
```

O desenvolvimento de programas Java MIDP não é diferente do desenvolvimento de qualquer outro aplicativo Java. Arquivos Java são criados, compilados e então são empacotados. No entanto, existem alguns passos que devem ser seguidos na criação de um programa J2ME, são eles:

1. Criar o arquivo Java (`.java`) usando um editor de texto;

2. Compilar o arquivo `.java` com o comando `javac`. Esse é o mesmo comando `javac` que o JDK J2SE fornece. O comando `javac` produz arquivos `.class`;

3. Preverificar o arquivo `.class`. O arquivo classe produzido pelo `javac` não é utilizável em dispositivos J2ME. O passo da preverificação simplifica o passo de carregamento da classe no dispositivo MIDP. Um novo arquivo `.class` com informações de preverificação é produzido;

4. Empacotar os arquivos `.class` verificados em um arquivo JAR (`.jar`);

5. Carregar o arquivo JAR e o arquivo JAD no dispositivo. O arquivo JAD descreve a maneira como o programa MIDP deve ser instalado e também é obrigatório;

A Figura 4.6 ilustra o processo de criação de um programa J2ME.

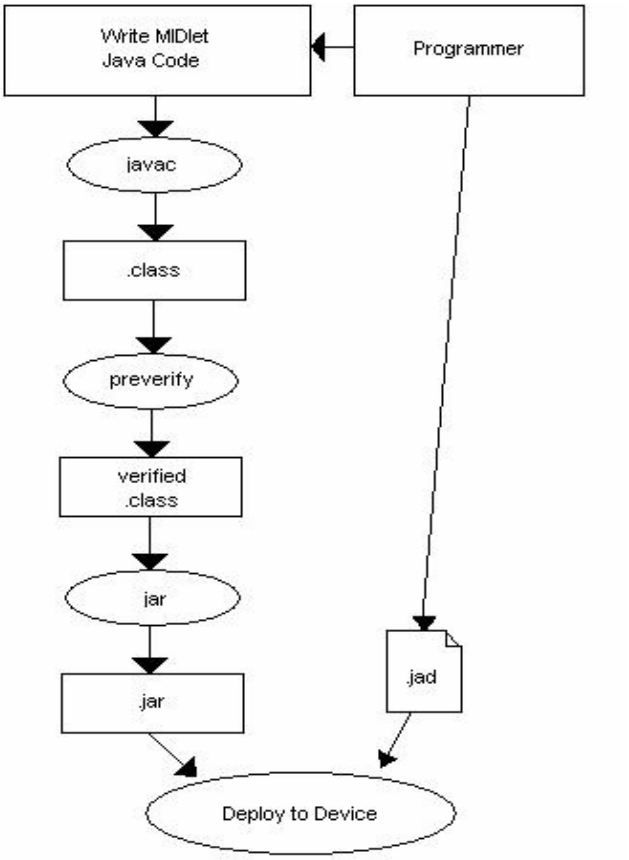

Figura 4.6 – Lançamento de um aplicativo J2ME

4.2. SERVLETS

Servlets[6] podem ser definidos como as classes Java que implementam, direta ou indiretamente, a interface `javax.Servlet.Servlet`, processam pedidos HTTP vindos de um cliente *Web*, e estendem a classe `HTTPServlet`. *Servlet*s que processam outros tipos de pedidos devem estender a classe `GenericServlet`. Além disso, para o funcionamento de servidor *Web* utilizando a tecnologia de *Servlets*, são necessário um certo número de classes de suporte que fornecem um ambiente completo para se programar e executar os *Servlet*s.

As funcionalidades oferecidas pela classe `Servlet` são: a inicialização e o controle do ciclo de vida do *Servlet*, processamento e manipulação de pedidos, criação de respostas para os pedidos, manutenção dos clientes *Web* através de sessões, manutenção da informação contextual, filtragem dos pedidos e respostas, chamado para outros componentes *Web*, e troca de informações com outros componentes *Web*.

[6] A palavra *Servlet* tenta denominar o aplicativo que roda no Servidor em uma arquitetura Cliente-Servidor.

4.2.1. Inicialização e ciclo de vida dos *Servlet*s

Instâncias de *Servlet*s são criados quando um pedido do cliente é recebido e uma instância do *Servlet* ainda não existe. Quando um *Servlet* é criado, suas classes são carregadas e uma instância do *Servlet* é criada. Em seguida, o método `init()` do *Servlet* é chamado para que seja possível a inicialização dos métodos de serviço como `doGet()` ou `doPost()`.

Quando o *container*[7] decide remover uma instância do *Servlet*, ele chama o método `destroy()` do *Servlet* para executar essa tarefa, então o *Servlet* e toda informação contida no mesmo é perdidos.

Pode-se definir algumas classes do tipo `listener` para o *Servlet* a qual fazem a captura de eventos do ciclo de vida do *Servlet* que ocorrem durante a execução do mesmo. Eventos do ciclo de vida incluem a criação e destruição de *Servlet*s, a manipulação de atributos, além da criação, invalidação e *timeout* de sessões.

A Figura 4.7 apresenta todas as classes e interfaces que fazem parte da implementação de um *Servlet* Java.

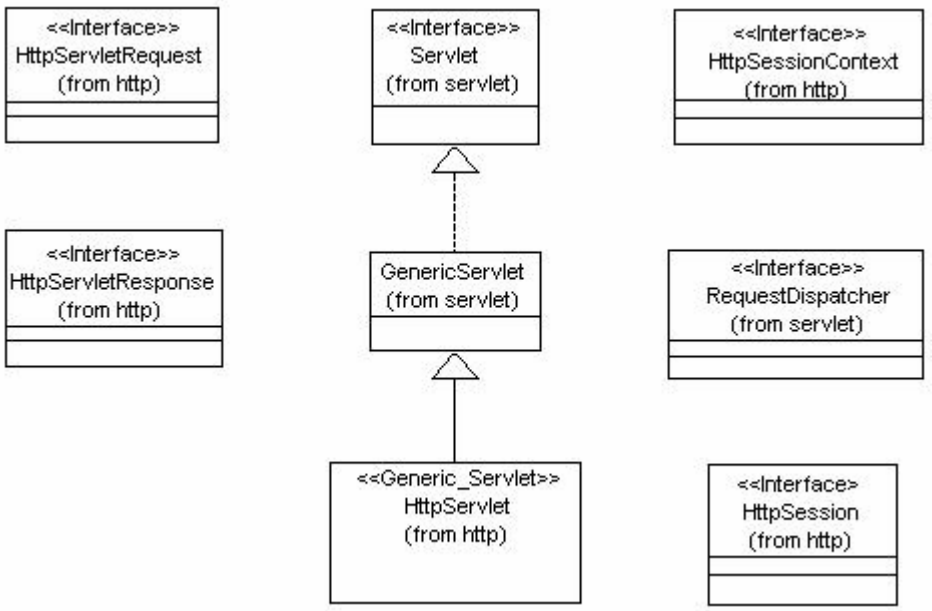

Figura 4.7 – Classes e interfaces de um Servlet

[7] Objeto que faz parte do servidor Web e responsável por controlar todas instâncias da classe *Servlet*.

4.2.2. Escrevendo funções para aplicativos *Servlet*s

A maneira como um *Servlet* funciona pode ser definida nos seguintes passos:

1. O *container* recebe os pedidos e passa estes pedidos para o *Servlet* ao invocar o método de serviço que faz parte do *Servlet*;

2. Para *Servlet*s HTTP, este método é tipicamente os métodos `doGet()` ou `doPost()`. Para um *Servlet* genérico, o método `service()` é chamado;

3. Os métodos de serviço definem os objetos de pedido e resposta como seu parâmetro e o *container* passa esses objetos para o *Servlet* quando o método é invocado.

4.2.3. Estado do Cliente

Os clientes *Web* várias vezes precisam manter seu estado através de múltiplas chamadas HTTP para o *Servlet*. O protocolo HTTP não mantém estados, ou seja, é definido como *stateless*[8], então os *Servlet*s utilizam um objeto `HTTPSession` para manter o estado dos clientes. O objeto `Session` de cada cliente é mantido ao se passar um identificador entre o cliente e o servidor. Isso pode ser feito através de cookies ou através de uma técnica chamada de *URL rewriting* onde um *Servlet* insere o identificador da sessão como uma string de parâmetro no URL quando o cliente não habilitou a utilização de *cookies*. Esta técnica é a preferida quando da comunicação com dispositivos *wireless*, os quais geralmente tem uma memória pequena e não suportam *cookies*.

Uma nova sessão pode ser criada ou uma sessão já existente encontrada usando o método `getSession()` no objeto que fez o chamado. Assim que uma sessão é criada ou encontrada, os métodos `getAttribute()` e `setAttribute()` podem ser chamados para acessar ou modificar o estado da sessão do cliente.

[8] Palavra em inglês que denota um objeto que não possui nenhum estado.

4.3. CONCEITOS DA LINGUAGEM XML

A linguagem XML (*eXtensible Markup Language*) é, como o próprio nome indica, uma linguagem do tipo *markup*. Tais linguagens são amplamente utilizadas para a representação dos mais diversos tipos de informação, desde as mais simples até as mais complexas.A utilização deste tipo de linguagem também é bastante difundida em razão da facilidade que proporcionam quando da representação e transmissão da informação, garantindo a portabilidade e a comunicação entre os mais diversos tipos de dispositivos, com as mais diversas tecnologias.

Nesse sentido, a linguagem XML costuma ser muito utilizada em um contexto onde não se sabe muito acerca dos elementos e do conteúdo de um documento, conhecendo-se porém a estrutura do mesmo. O conteúdo do documento, assim como os elementos e atributos utilizados estão a critério do desenvolvedor. É possível se desenvolver a formatação, o conteúdo e as especificações para a representação da sua informação, assim permitindo uma interoperabilidade entre diferentes sistemas. Esta é uma característica fundamental do XML: a definição rígida de padrões de conteúdo e formatação confere ao XML a possibilidade de efetuar transações sobre a informação com rapidez e segurança.

4.3.1. A sintaxe do XML

Um exemplo de documento XML pode ser visto abaixo [6]:

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

A primeira linha no documento, o Cabeçalho XML, deve sempre ser incluída. Ela defina a versão XML do documento. Neste caso, o documento segue a especificação da versão 1.0:

```
<?xml version="1.0"?>
```

A próxima linha define o primeiro elemento do documento, o Elemento Raiz. O elemento raiz é o elemento de mais alto nível em um documento XML e deve ser a primeira

tag[9], a abertura, e a última *tag*, o fechamento, dentro de um documento. Esse elemento fornece um ponto de referência que permite que o tradutor XML ou aplicativos com suporte a XML façam o reconhecimento de um início e um fim de um documento XML.

```
<note>
```

As linhas seguintes definem quatro elementos, denominados elemento "filhos" do elemento raiz. São eles: *to, from, heading and body*.

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

A última linha define o fechamento do elemento raiz.

```
</note>
```

A seguir, são apresentadas as principais regras de sintaxe de um documento XML, juntamente com exemplos que ajudam a esclarecer tais regras.

1. Todos os elementos XML devem ter uma *tag* de fechamento.

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

2. As *tags* de elementos XML são *case sensitive*[10]. A *tag* `<Letter>` é diferente da *tag* `<letter>`. Assim, as *tags* de início e fechamento devem ser escritas da mesma forma.

```
<Message>This is incorrect</message>
<message>This is correct</message>
```

3. Todos os elementos XML devem estar devidamente aninhados.

```
<b><i>This is incorrect</b></i>
<b><i>This text is bold and italic</i></b>
```

[9] O termo *tag* foi mantido em seu formato original na língua inglesa devido a sua grande utilização nessa forma na literatura técnica.
[10] O termo *case sensitive* denota a existência de diferenciação entre letras maiúsculas e minúsculas.

4. Todos os documentos devem ter um único Elemento Raiz. Todos os outros elementos devem estar corretamente aninhados ao elemento raiz ou ao seu elemento “pai”.

```
<root>
  <child>
    <subchild>
    </subchild>
  </child>
</root>
```

5. Todos os valores de atributo devem estar entre aspas. Aspas duplas são mais utilizadas, pois permitem que os documentos XML sejam mapeados diretamente para programas Java. Nos exemplos a seguir o primeiro está incorreto e o segundo está correto.

```
<?xml version="1.0"?>
<note date=12/11/99>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<note date="12/11/99">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Na medida do possível, deve-se evitar a utilização de atributos para a descrição de informações. O uso de atributos está sujeito a uma série de inconvenientes, como por exemplo:

- Atributos não podem armazenar múltiplos valores (elementos podem);
- Atributos não são expansíveis (para mudanças futuras);
- Atributos não podem descrever estruturas (tais como elementos “filho”);
- Atributos são mais difíceis de manipular por código de programa;
- Valores de atributos são difíceis de serem validados através de um DTD.

Como comparação podemos verificar a diferença entre dois pequenos documentos utilizando elementos e atributos. Fica clara a vantagem do primeiro em relação ao segundo.

```
<?xml version="1.0"?>
<note>
<date>
  <day>12</day>
  <month>11</month>
  <year>99</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<note day="12" month="11" year="99"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

4.3.2. A validação de um documento XML – o DTD

Um documento XML é dito “bem formado” quando está em conformidade com as regras de sintaxe descritas acima. No entanto, para que um documento XML seja considerado “válido” ele deve, além de ser bem formado, estar de acordo com as regras impostas por uma Definição de Tipo de Documento – DTD (*Document Type Definition*). O seguinte exemplo é um documento XML válido, que apresenta a referência ao DTD que respeita.

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "InternalNote.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

O propósito de um DTD é definir os blocos construtores permitidos para um documento XML. Ele define a estrutura do documento, com uma lista de elementos permitidos. Um exemplo de documento XML e seu respectivo DTD podem ser vistos abaixo:

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

As Figuras 4.8 e 4.9 retratam o fluxo de confecção de um documento XML e de seu DTD, respectivamente.

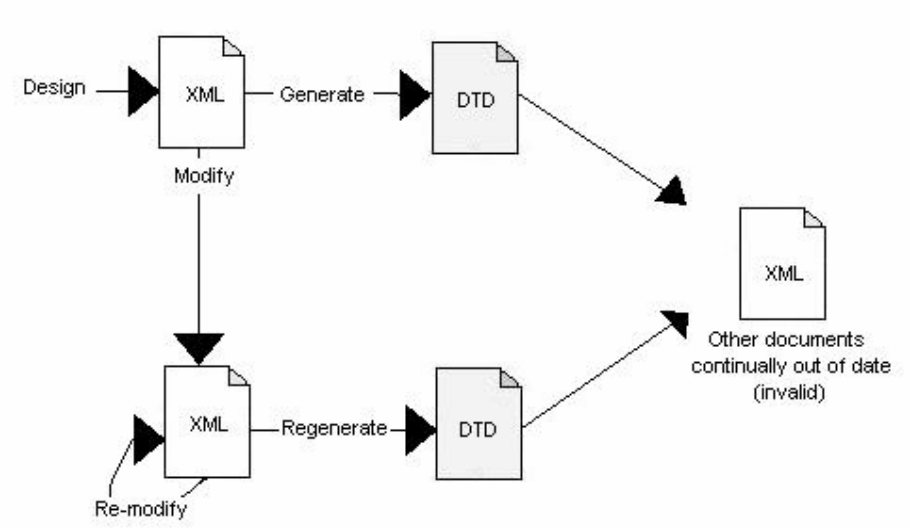

Figura 4.8 – Fluxo de confecção de um documento XML

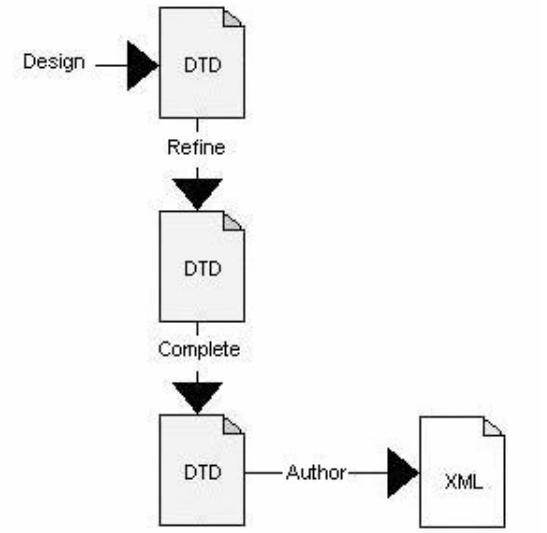

Figura 4.9 – Fluxo de confecção de um DTD

4.4. O BANCO DE DADOS MYSQL

Bancos de dados, hoje em dia, são parte fundamental da vida de quase todos seres humanos. Viver sem sistemas de bancos de dados em nosso mundo é hoje em dia praticamente impossível. Sem nem mesmo perceber, a todo o momento estamos usando um banco de dados, mesmo nas mais triviais tarefas.

Bancos, universidades e bibliotecas são três exemplos de organizações que dependem totalmente de bancos de dados. A própria Internet, usa um sistema de banco de dados para controle e funcionamento dos sites. Normalmente, bases de dados com muitas informações são armazenadas em computadores de grande porte, chamados de servidores, e que permitem o uso das mesmas informações, através de uma rede, por um número (diz-se) ilimitado de usuários.

Um dos mais rápidos programas para servidores de SQL (*Simple Query Language* – Linguagem de pesquisa simples), hoje no mercado, é o MySQL, desenvolvido pela T.c.X. DataKonsultAB. Este programa está disponível para download em sua versão original em http://www.mysql.com, em inglês, e também em http://www.mysql.com.br para sua versão brasileira, onde encontram-se projetos de tradução e documentação do MySQL em português.

Além de oferecer vários recursos não existentes em outros servidores, o MySQL tem a vantagem de ser totalmente gratuito para uso tanto comercial, quanto privado, em conformidade com a licença pública GPL.

As principais metas da equipe de desenvolvimento do MySQL é construir um servidor rápido e robusto. Os recursos acima mencionados incluem:

- Capacidade de lidar com um número ilimitado de usuários;
- Capacidade de manipular mais de cinqüenta milhões (50.000.000) de registros;
- Execução muito rápida de comandos, provavelmente o mais rápido do mercado;
- Sistema de segurança simples e funcional.

4.4.1. Acessando o Banco de Dados

Para acessar o servidor MySQL, é preciso utilizar o comando que segue. É importante lembrar, que o MySQL tem seu próprio cadastro de nomes e senhas. Você também pode usar clientes gráficos para o MySQL, como o FreeMascon e o MyNavigator.

```
----------------------------------------------------------------------
mysql -u elaine -p supersenha
Sintaxe:

mysql -u (usuário) -p (senha)
ou
mysql -h servidor -u (usuário) --password=(senha)
----------------------------------------------------------------------
```

Uma vez no prompt do MySQL, podemos começar a utilizar os comandos do MySQL e manipular os dados e o servidor. Contudo, antes de modificar a base de dados, nós devemos escolher qual desejamos usar, da seguinte forma:

```
------------------------------
mysql>use teste;
Base de dados alterada.
------------------------------
```

Troque teste, pelo nome da base de dados desejada. Você obterá uma mensagem confirmando a alteração da base de dados. Isto significa que você está conectado a ela.

Repare que o comando está seguido de ponto e vírgula, pois como em C, a maioria dos comandos do MySQL são sucedidos por ponto e vírgula.

4.4.2. Estrutura hierárquica de estruturas de dados

Uma base de dados, nada mais é do que estruturas complexas de dados. Estes dados são gravados em forma de registros em tabelas. A estrutura de hierarquia utilizada para o registro das informações pode ser vista a seguir, onde os nomes à direita representam ordens de hierarquia superior:

Base de dados < Tabela < Registro < Coluna (datatype)

Banco de dados < Tabela < Linha < Campo

As duas linhas acima mostram os termos normalmente usados para a denominação de bancos de dados e seus componentes. Os campos podem ser de diferentes tipos e tamanhos,

permitindo ao programador criar tabelas que satisfaçam ao escopo do projeto. A decisão de quais campos usar e quais não usar é muito importante, pois influi drasticamente na performance da base de dados que estamos desenvolvendo, portanto, é de bom grado um conhecimento sólido destes conceitos.,

4.4.3. Campos

Os campos são a parte fundamental de uma base de dados. É nos campos que as informações ficam armazenadas. Para uma otimização da base de dados, antes e utilizar, devemos definir os campos que desejamos usar, e especificar o que cada um pode conter. A seguir temos uma lista dos tipos de campos suportados pelo MySQL.

- **CHAR(M)**: usados para caracteres alfanuméricos, como endereços e nomes. Seu tamanho é fixo e instaurado ao ser criado. Um campo do tipo CHAR pode ter de 1 a 255 caracteres;

- **VARCHAR(M)**: campo VARCHAR, aloca apenas o espaço necessário para gravação. Contudo, vale lembrar que trocamos espaço por velocidade, pois este campo é 50% mais lento que o anterior;

- **INT(M) [Unsigned]**: suporta o conjunto dos números inteiros numa variação de -2147483648 a 2147483647. O parâmetro *Unsigned* pode ser passado, excluindo os números negativos, proporcionando um intervalo de 0 até 4294967295**;**

- **FLOAT[(M,D)]**: usados para representar números com maior precisão;

- **DATE**: A forma padrão , é 'AAAA-MM-DD';

- **TEXT/BLOB**: usados para guardar grandes quantidades de caracteres de 0 a 65535 bytes. A única diferença entre os campos BLOB e TEXT está no fato de um campo TEXT não ser *case sensitive* e os BLOBs sim;

- **SET**: permite que o usuário faça uma escolha dentre determinado número de opções. Cada campo pode conter até, 64 opções;

- **ENUM**: semelhante ao SET, com a diferença que apenas um valor pode ser escolhido.

4.4.4. Registros

Um conjunto de campos relacionados, forma o que chamamos de registro. Ex:

```
-------------------------
nome CHAR(15);
email CHAR(25);
telefone INT;
-------------------------
```

4.4.5. Tabelas

Um conjunto de registros, forma uma tabela. Como no exemplo anterior, poderíamos ter centenas de nomes diferentes cadastrados em nossa tabela de pessoas. Cada conjunto de dados corresponde a um registro. Antes de usar uma base de dados, ou dar qualquer comando, nós precisamos de uma tabela pelo menos, para armazenar os dados: Isto pode ser feito usando o comando `CREATE TABLE`, que recebe como parâmetro o nome da tabela que desejamos usar.

```
-------------------------------------------------------------
mysql> CREATE TABLE teste(
>codigo INT,
>nome CHAR(15),
>email CHAR(25),
>telefone INT);
-------------------------------------------------------------
```

As seguintes opções podem ser adicionadas a qualquer campo de uma tabela, concatenando recursos adicionais a estes campos.

- **Chave primária**: Para se definir uma chave primária, basta adicionar 'PRIMARY KEY' a definição do campo que se deseja a não duplicidade;

- **Auto incremento**: faz com que conforme novos registros são criados, automaticamente estes obtém valores que correspondem ao valor deste mesmo campo no registro anterior, somado a 1.

4.4.6. Manipulando a base de dados

Uma base de dados pode ser manipulada com quatro operações básicas: Incluir, Apagar, Alterar, e Pesquisar. A seguir temos as formas utilizadas pelo MySQL para o tratamento da base de dados:

- **Inserindo registros**: Para se adicionar dados a uma tabela, usamos o comando INSERT, como o exemplo que segue. É importante lembrar-se sempre de passar

para o comando INSERT um número de parâmetros igual ao número de campos na tabela que está recebendo os dados. Caso contrario, você obterá uma mensagem de erro. O mesmo erro também ocorre ao tentar inserir mais parâmetros do que o número de campos suportado pela tabela. Ex:

```
-------------------------------------------------
mysql>INSERT INTO teste VALUES
mysql>(NULL, 'Ernesto', 'ernesto@nbsnet.com.br',
mysql>2742729);
-------------------------------------------------
```

- **Pesquisando registros**: As pesquisas no MySQL são feitas através do comando SELECT. Este comando pode fazer desde uma simples e trivial pesquisa até uma pesquisa extremamente complexa. Ex:

Lista todos registros da tabela teste:

```
-------------------------------------------------
mysql>SELECT * FROM teste;
-------------------------------------------------
```

Lista todos os registros da tabela teste que contém "Ernesto" no campo nome.

```
-------------------------------------------------
mysql>SELECT * FROM teste
mysql>WHERE (nome = "Ernesto");
-------------------------------------------------
```

- **Apagando registros**: Isto pode ser facilmente feito usando o comando DELETE, que tem o funcionamento semelhante ao comando INSERT. Ex:

Apaga todos os registros que têm o conteúdo "2744747" no campo telefone.

```
-------------------------------------------------
mysql>DELETE FROM teste
mysql>WHERE (telefone = 2744747);
-------------------------------------------------
```

- **Alterando registros**: Utiliza-se o comando UPDATE. Ex:

Substitui o nome ‘Ernesto’ por ‘Diogo’.

```
-------------------------------------------------
mysql>UPDATE teste SET nome = 'Diogo'
mysql>WHERE nome = 'Ernesto';
-------------------------------------------------
```

Além destes comandos, o MySQL também dispõem de inúmeras outras funções e operações tais como os operadores lógicos AND, OR, NOT e ORDER BY.

5. IMPLEMENTAÇÃO

Este capítulo trata da real implementação do Ambiente de Desenvolvimento de Serviços Baseados em Presença proposto pelos autores. Este ambiente surgiu da necessidade de se ter um sistema que fornecesse as funcionalidades básicas para o desenvolvimento dos serviços IMPS, ou seja, o suporte à troca de mensagens instantâneas e, principalmente, um método para se determinar e informar a localização de cada usuário.

Nesse sentido, e levando em conta a grande expansão das redes tipo *Wireless* LAN tanto no meio acadêmico como no comercial-corporativo, decidiu-se por utilizar como fonte da informação de localização o fato de um usuário estar conectado a um determinado *Access Point* (AP). Assim, uma vez conhecida a posição de cada um dos AP's monitorados pelo sistema, é possível identificar o local (dentro da área de cobertura do AP) em que cada usuário se encontra. Contudo, uma vez que essa área de cobertura pode abranger alguns metros, a informação de localização torna-se então uma informação de **presença**, que pode ser confundida com o *Cell ID* de serviços de presença baseados em tecnologia celular.

Uma vez escolhido o método para a determinação da informação de presença, o próximo passo seria implementar as demais funcionalidades de troca de mensagens e obtenção da informação de presença. Para tal, optou-se por seguir a especificação da iniciativa *Wireless Village*, a qual intenta disponibilizar um padrão para a intercompatibilidade desse tipo de funcionalidade (tal como visto no Capítulo 3).

Isto posto, é apresentada a seguir um descrição detalhada do ambiente implementado, bem como cada uma de suas partes e propriedades. O capítulo começa com a indicação dos objetivos a serem alcançados com a implementação; a seguir é apresentada uma descrição detalhada da arquitetura do sistema, indicando cada um dos seus componentes, suas funcionalidades e o esquema de comunicação desenvolvido; a partir daí são apresentadas individualmente cada uma das partes componentes do sistema, explicitando-se suas características e formas de implementação; finalmente, são apresentados os serviços implementados sobre o sistema desenvolvido, descrevendo-se detalhadamente sua operação e funcionalidades.

5.1. OBJETIVOS

O presente trabalho tem como objetivo a implementação de um Ambiente de Desenvolvimento de Serviços Baseados em Presença, sob plataforma IEEE 802.11 e seguindo a especificação da Iniciativa *Wireless Village*.

Este ambiente deverá ser capaz de prover as funcionalidades básicas para o desenvolvimento desses serviços, tais como o suporte à Troca de Mensagens Instantâneas, bem como uma tecnologia capaz de identificar e informar o local em que cada usuário do sistema se encontra (Presença).

A partir deste ambiente, poderão ser implementados os mais diversos serviços, só limitados pela imaginação do desenvolvedor em utilizar os recursos disponíveis.

5.2. PARTES QUE COMPÕEM O SISTEMA

Para iniciar a compreensão do sistema, as seções seguintes irão identificar cada uma das partes que compõem a arquitetura, descrevendo seus modos de construção, seu funcionamento e a forma como ocorre o interfaceamento entre elas.

Os componentes identificados e já citados dentro da estrutura do sistema foram:

- Banco de Dados, comum a todas as partes e responsável pelo armazenamento das informações;
- Cliente, parte responsável pelo interfaceamento entre o usuário e o sistema;
- SAP, parte central do sistema, responsável pela autenticação e troca de mensagens entre os componentes do sistema;
- Servidor de Presença, o qual lida com todas as informações de presença, gerenciado-as e disponibilizando-as;
- Servidor de Troca de Mensagens Instantâneas, responsável, como o nome diz, pela análise e gerência de mensagens desta natureza;
- LCS (*LoCation Server* – Servidor de Localização), responsável pela obtenção e gerenciamento das informações de presença de cada usuário do sistema.

O ambiente utilizado para desenvolvimento de todo o código Java foi o *jEdit 4.2pre5*, *http://www.jedit.org*.

5.2.1. BANCO DE DADOS

O servidor de banco de dados utilizado foi o *MySQL® Database Server, version 4.0*, http://www.mysql.com .

A função do Banco de Dados do Ambiente implementado é fornecer um mecanismo para armazenamento seguro e de fácil acessibilidade das informações relativas ao cadastro de usuários, mensagens instantâneas e informações de presença.

O Banco de Dados criado foi denominado `wv.sql` e possui as seguintes tabelas:

- **User:** guarda informações relativas ao usuário, tais como seu ID e senha;
- **message:** define as informações constantes em uma mensagem instantânea genérica, tais como remetente, destinatário e conteúdo da mensagem;
- **presenceMessage:** define as informações constantes em uma mensagem instantânea de presença. Essas informações são as mesmas constantes numa mensagem genérica, adicionando-se o número do AP do local ao que se destina a mensagem;
- **Location:** é o responsável por guardar as informações de presença. É a tabela que é atualizada pelo LCS e guarda as vinculações de cada endereço MAC com seu respectivo número de AP. Além disso, informa se o usuário mudou de área recentemente (informação esta utilizada em serviços como o de Bem-Vindo, descrito a seguir).
- **APInfo:** esta tabela guarda informações estáticas acerca da localização de cada AP monitorado pelo sistema. Dentre estas informações podemos citar o número do AP, sua localização e o IP de acesso à página de configuração (utilizado pelo LCS para realizar o monitoramento).

A Figura 5.17 mostra uma representação do Banco de Dados `wv.sql`, com todas as suas tabelas.

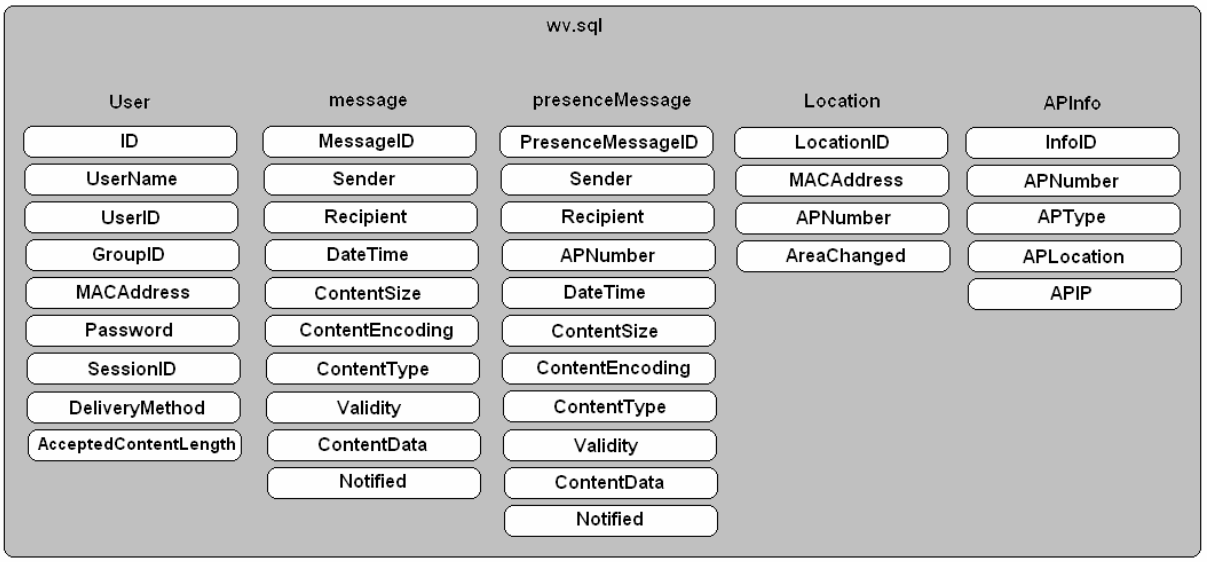

Figura 5.17 – Representação do Banco de Dados wv.sql

5.2.2. CLIENTE

No decorrer da programação, eventuais testes eram realizados através da compilação do código fonte no *J2ME Wireless Toolkit 1.0.4_01 for Windows – Sun Microsystems©, http://java.sun.com/products/j2mewtoolkit/*, que foi o emulador de dispositivo móvel adotado para o projeto.

O cliente, em suma, tem a função de iniciar a comunicação com o SAP através do envio de uma mensagem XML que será posteriormente interpretado pelo mesmo. Para isso, o cliente foi estruturado de acordo com as classes que se seguem:

- **`AbstractRequest`**: define um modelo para todas as requisições.

- **`PresenceRequest:`** escreve o código XML, baseado na especificação Wireless Village, para requisições de atributos de presença. Tal classe fornece o seguinte método:

  - **`getListRequest:`** escreve um código XML para obter a lista de contato e a envia para o servidor Wireless Village (SAP).

- **`PrimitiveRequest`**: escreve códigos XML para requisição de atributos primitivos. Tal classe fornece o seguinte método:

  - **`loginRequest`**: escreve um código XML para requerer *login* e o enviar para o servidor Wireless Village (SAP).

- **`Session`**: contem todos os atributos de sessão usados para logar e deslogar do servidor Wireless Village (SAP). Tal classe tem os seguintes métodos:
  - **`getInstance`**: cria uma sessão ou retorna a instância ativa no caso de já existir uma.
  - **`getLoginStatus`**: retorna o *status* do *login*.
  - **`getSessionID:`** retorna o *sessionID*.
  - **`setLoginStatus`**: seta o *status* de *login*.
  - **`setSessionID`**: seta o *sessionID*
  - **`setStatusDescription`**: seta a descrição do *status* de *login*.
- **`WVClient`**: *midlet* do sistema (interface gráfica)
- **`WVConnection`**: interface para estabelecer a conexão entre o Cliente e o SAP.
- **`WVConnector`**: realiza a comunicação entre o Cliente e o SAP, enviando um XML por HTTP com as devidas requisições das outras classes. Tal classe fornece o seguinte método:
  - **`communicate`**: estabelece a conexão HTTP entre o cliente WV e o SAP WV.
- **`XMLMessage`**: cria a mensagem XML a ser enviada para o servidor Wireless Village. Tal classe fornece os seguinte métodos:
  - **`createSubElement`**: cria um sub-elemento que fará parte de um contexto maior a ser inserido em um nó pai.
  - **`insertElement`**: insere o valor correspondente de um elemento.
  - **`setSessionDescriptor`**: insere os atributos de *session* nos elementos correspondentes.
  - **`setTransactionContent`**: insere o conteúdo da *transaction* no elemento correspondente.

- `toString`: imprime a mensagem XML gerada no saída padrão do sistema.
- `wrapMessage`: constrói a mensagem XML baseada na hierarquia prevista na especificação WV.

- **`XMLUtils`**: contém os utilitários para manipular a mensagem XML. Essa classe apresenta os seguintes métodos:
  - **`decode`**: converte uma *string* em uma *tag* XML usando caracteres de escape.
  - **`getNodeValue`**: busca o conteúdo de um elemento determinado pertencendo à mensagem XML.

5.2.3. SAP

O código da parte do Servidor foi testado no *Apache Tomcat 4.1.12 - Apache Software Foundation©*, http://jakarta.apache.org/.

O SAP, como já mencionado anteriormente, tem a função de mediar a comunicação entre o cliente e os demais servidores. Por esse motivo o SAP foi estruturado em três partes. A primeira trata do *servlet* que lida com as chamadas provenientes do cliente e encontra-se estruturada de acordo com as classes que se seguem:

- **`SAP_CSPservlet`**: define a *servlet* utilizada pelo sistema. É responsável por toda a mediação SAP-CLIENTE e conta com diversos métodos para tratar cada um dos diversos XML's enviados pelo Cliente.

- **`MessageInfo`**: utilizada para se definir os parâmetros essenciais da mensagem a ser enviada de volta para o cliente.

- **`SAP_Authentication`**: utilizada para realizar, através de consultas no banco de dados, a autenticação do cliente no sistema *Wireless Village*.

- **`BD`**: contem métodos auxiliares para lidar com consultas a banco de dados e é utilizada por muitas das classes do sistema que fazem tais consultas.

A segunda parte do SAP cuida das chamadas RMI (*Remote Method Invocation*) para o Servidor de Presença. Tais chamadas são feitas com o objetivo de buscar informações,

requeridas pelo cliente, de presença. Para que as chamadas RMI aconteçam, as seguintes classes foram implementadas:

- **`PresenceSAPInterface`**: interface que referencia os métodos RMI que serão utilizados entre o SAP e o Servidor de Presença. Fornece os seguintes métodos:
  - **`getAreaChanged_RMI:`** utilizado para se verificar se houve alteração no campo *areaChanged* do usuário em questão.
  - **`getAPNumber_RMI:`** utilizado para se obter o número do AP do usuário em questão.
  - **`getNPresenceMessages_RMI:`** utilizado para se obter o número de mensagens de presença destinadas ao cliente.
  - **`getPresenceMessage_RMI:`** utilizado para se obter todos os campos da mensagem de presença a ser recebida.
  - **`getContactList_RMI:`** utilizado para se obter uma lista de Contatos do Servidor de Presença.
- **`PresenceSAPInterfaceClient`**: classe utilizada pelo cliente para realizar a comunicação com o Servidor de Presença e utiliza-se da interface **`PresenceSAPInterface`** para tanto. Fornece os seguintes métodos:
  - **`getAreaChanged:`** utilizado para se obter, através do método **`getAreaChanged_RMI`** da interface **`PresenceSAPInterface`**, o valor do campo *AreaChanged* do usuário em questão.
  - **`getAPNumber:`** utilizado para se obter, através do método **`getAPNumber_RMI`** da interface **`PresenceSAPInterface`**, o número do AP do usuário em questão.
  - **`getNPresenceMessages:`** utilizado para se obter, através do método **`getNPresenceMessages_RMI`** da interface **`PresenceSAPInterface`**, o total de mensagens de presença encaminhadas para o cliente.

- **`getPresenceMessage:`** utilizado para se obter, através do método **`getContactList_RMI`** da interface **`PresenceSAPInterface`**, todos os campos da mensagem de presença a ser recebida.

- **`getContactList:`** utilizado para se obter, através do método **`getContactList_RMI`** da interface **`PresenceSAPInterface`**, a lista de Contatos do Servidor de Presença.

Finalmente, a terceira parte do SAP cuida das chamadas RMI (*Remote Method Invocation*) para o Servidor de Mensagem Instantânea. Tais chamadas são feitas com o objetivo de buscar informações, requeridas pelo cliente, de mensagem instantânea. Para que as chamadas RMI aconteçam, as seguintes classes foram implementadas:

- **`InstantMessageSAPInterface`**: interface que referencia os métodos RMI que serão utilizados entre o SAP e o Servidor de Mensagem Instantânea. Fornece os seguintes métodos:

  - **`getNMessages_RMI:`** utilizado para se obter o número de mensagens instantâneas destinadas ao cliente.

  - **`getMessage_RMI:`** utilizado para se obter todos os campos da mensagem instantânea a ser recebida.

- **`InstantMessageSAPInterfaceClient`**: classe utilizada pelo cliente para realizar a comunicação com o Servidor de Mensagem Instantânea e utiliza-se da interface **`InstantMessageSAPInterface`** para tanto. Fornece os seguintes métodos:

  - **`getNMessages:`** utilizado para se obter, através do método **`getNMessages_RMI`** da interface **`InstantMessageSAPInterface,`** o número de mensagens instantâneas destinadas ao cliente.

  - **`getMessage:`** utilizado para se obter, através do método **`getMessages_RMI`** da interface **`InstantMessageSAPInterface,`** todos os campos da mensagem instantânea a ser recebida.

5.2.4. SERVIDOR DE PRESENÇA

O Servidor de Presença, como já mencionado anteriormente, é responsável por fornecer métodos, que possam ser instanciados pelo SAP via RMI, para fornecer informações de presença. Dessa forma, o Servidor de Presença foi estruturado de acordo com as seguintes classes:

- **`PresenceSAPInterfaceImpl`**: oferece os métodos que podem ser chamados pelo SAP, via RMI, através dos objetos instanciados remotamente. Dentre os principais métodos segue-se a implementação daqueles já mencionados na inteface **`PresenceSAPInterface`** do SAP: **`getAreaChanged_RMI`**, **`getAPNumber_RMI`**, **`getNPresenceMessages_RMI`**, **`getPresenceMessage_RMI`**, **`getContactList_RMI`**.

- **`Presence_LCS`**: fornece os métodos para buscar as informações de localização no LCS.

5.2.5. SERVIDOR DE INSTANT MESSAGING

O Servidor de Mensagem Instantânea, como já mencionado anteriormente, é responsável por fornecer métodos, que possam ser instanciados pelo SAP via RMI, para fornecer informações de mensagens instantâneas. Dessa forma, o Servidor de Mensagem Instantânea foi estruturado de acordo com as seguintes classes:

- **`InstantMessageSAPInterfaceImpl`**: oferece os métodos que podem ser chamados pelo SAP, via RMI, através dos objetos instanciados remotamente. Dentre os principais métodos segue-se a implementação daqueles já mencionados na inteface **`InstantMessageSAPInterface`** do SAP: **`getNMessages_RMI`**, **`getMessage_RMI`**.

5.2.6. LCS

O ambiente utilizado para desenvolvimento de todo o código JAVA foi o *jEdit 4.2pre5*, http://www.jedit.org.

O papel do LCS é, como o próprio nome informa, atualizar o Banco de Dados com informações de presença. A forma como o LCS obtém tal informação e a disponibiliza é descrita a seguir.

Primeiramente, o LCS varre a tabela *APInfo* do Banco de Dados de forma a verificar quantos AP's estão cadastrados no sistema. A seguir, ele verifica todos os AP's, um a um, utilizando os endereços IP cadastrados na tabela. Durante essa verificação, o LCS acessa a página de configuração do AP e extrai do código HTML os endereços MAC de todos os usuários logados naquele AP.

Vale lembrar que o modo como os endereços MAC estão dispostos na página de configuração varia de fabricante para fabricante. O algoritmo desenvolvido para a obtenção dessa informação só é valida para AP's da marca *AirLAN*. No entanto, o LCS foi desenvolvido de forma modularizada e a inclusão de novos algoritmos para os demais fabricantes pode ser feita de forma simples, visto que o LCS verifica o campo *APType*, o qual contém a descrição da marca do AP e, conseqüentemente, de qual algoritmo utilizar.

De posse dos endereços MAC, o LCS realiza a atualização da informação de presença no Banco de Dados. Para realizar tal atualização, o LCS primeiramente verifica na tabela *Location* se os endereços MAC descobertos na leitura atual já estavam associados ao mesmo AP. Para estes, o LCS não realiza nenhum novo registro. No entanto, para os endereços MAC que não apareciam na tabela, o LCS irá criar uma nova entrada, setando o *flag AreaChanged* para '*Yes*'. Uma terceira hipótese é a de que o endereço MAC descoberto na leitura atual já estivesse presente na tabela, porém associado a um número de AP (*APNumber*) diferente. Isso significa que o usuário mudou de área e, assim o LCS criará uma nova entrada e excluirá as anteriores.

Todo este procedimento é realizado à uma taxa de 2 vezes por minuto (30 em 30 segundos). Este intervalo de tempo foi escolhido por ser suficiente para manter as informações de presença da tabela *Location* sempre consistentes.

Para realizar os procedimentos mencionados, o LCS foi estruturado de acordo com as seguintes classes:

- **`APMonitor`**: utilizada para monitorar todos os *Access Points* cadastrados no sistema.

- **`UpdateLocation`**: utilizada para atualizar o banco de dados de acordo com a entrada e/ou saída dos usuários de cada *Access Point*.

5.3. ARQUITETURA DO SISTEMA

Tendo sempre em vista manter a compatibilidade com a especificação da Iniciativa *Wireless Village*, a arquitetura do sistema foi projetada de forma modular e robusta, permitindo que outras possíveis funcionalidades possam ser integradas sem maiores dificuldades.

Como ponto de partida dessa arquitetura, tem-se o cliente como principal entidade, visto que este representa o iniciador de todas as comunicações realizadas no sistema. Uma vez que a especificação adotada, *Wireless Village*, tem como principal mercado alvo os dispositivos portáteis: celulares e PDA's, optou-se pela tecnologia J2ME que é considerada como a mais apropriada para estes dispositivos. Mesmo sabendo que hoje tais dispositivos ainda não são capazes de realizar comunicação WLAN, os autores acreditam que existe uma grande tendência de simbiose entre as tecnologias WLAN e GPRS, já que ambas apresentam consideráveis vantagens comparativas. Essa simbiose aconteceria através da migração automática de um sistema GPRS, quando o usuário estiver em trânsito, para um sistema WLAN, quando houver a presença de *hotspots*, tal como visto no Capítulo 2. Assim sendo, diante da inexistência de dispositivos que implementam as duas tecnologias, utilizou-se uma alternativa que, no caso, simularia os futuros dispositivos portáteis. Tal implementação consiste na utilização de um Laptop, o qual realiza conexão WLAN através de uma placa *Wireless LAN*, rodando um emulador de celular, *J2ME Wireless Toolkit 1.0.4_01 for Windows – Sun Microsystems©*, que permite conexão GPRS. Desta forma, o cliente mantém-se consistente com o proposto visto que o mesmo é programado em um ambiente de celular, utilizando a tecnologia J2ME, e, no mais, é possível a conexão deste com os AP's via WLAN.[11]

Surgindo como um mediador das comunicações do sistema, tem-se o SAP (*Service Access Point*) como a figura central[12] da arquitetura, responsável pelo tratamento das chamadas originadas ou com destino a qualquer outro elemento do sistema. Tais chamadas, quando provenientes do Cliente, são realizadas por meio da troca de mensagens no padrão XML sobre uma conexão HTTP estabelecida a cada transação.

[11] Desse ponto em diante, todas as referências que existem para o cliente fazem alusão à alternativa de projeto utilizada: laptop com emulador celular.
[12] A idéia de *central*, neste caso, é associada ao fato de ser o SAP a entidade central da arquitetura, no sentido em que media as comunicações entre os outros elementos.

Como plataforma para permitir o tratamento, por parte do SAP, das chamadas foi implementado um *Servlet*, que, para cada XML encaminhado, realiza determinado código. A essência desse código tem duas classes principais, uma primeira que trata localmente a chamada e gera de imediato um XML para o requisitante e uma segunda que encaminha a chamada, via RMI (*Remote Method Invocation* – recurso Java que permite instanciar objetos remotamente), para uma outra entidade.

Com o fim único de manter informações atualizadas sobre a localização dos clientes, tem-se uma outra entidade denominada LCS (*Location Server*). Esse servidor, que representa o ponto chave para os serviços implementados nesse projeto, extrai dinamicamente de cada *access point* vinculado ao sistema o MAC dos usuários associados ao mesmo e, para permitir a pesquisa por outra entidade, armazena essa informação em um banco de dados MySQL. Com o objetivo de manter essas informações de presença atualizadas, o LCS, utilizando a tecnologia JDBC (Java Database Connectivity), armazena-as em um banco de dados de acordo com a freqüência de busca nos AP´s, estipulada pelo administrador do sistema.

Uma outra entidade existente nesse sistema é o Servidor de Presença, *Presence Server*. Esse servidor, diferentemente dos outros acima mencionados, trata-se de uma das entidades funcionais do *Wireless Village Server* e atua no sentido de buscar e tratas as informações de presença no banco de dados, assim como disponibilizá-las para o cliente mediante solicitação.

Finalmente, para gerenciar e executar todo as tarefas relativas a troca de mensagens, tem-se o Servidor de Troca de Mensagens Instantâneas, *Instant Messaging Server*. Esse servidor, juntamente com o Servidor de Presença, constitui a base para o desenvolvimento de serviços IMPS.

Analisando em um contexto geral, o ambiente implementado consiste, basicamente, no LCS, o qual trabalha em paralelo com as chamadas ocorridas no sistema, e em uma lista de transações que são iniciadas pelo cliente e que repercutem por todas as entidades. A arquitetura completa e o formato das mensagens trocadas entre as diversas entidades pode ser visualizada na Figura 5.1, a qual ilustra a execução de um serviço IMPS genérico.

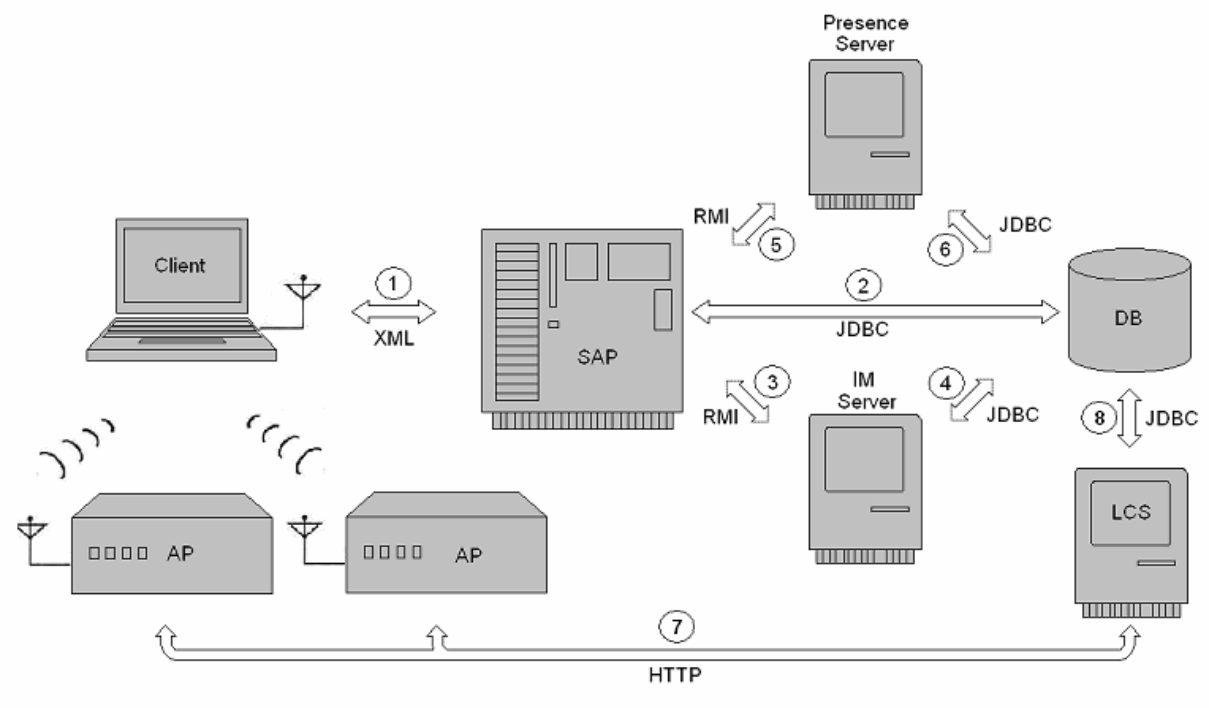

Figura 5.1 – Arquitetura completa e o formato das mensagens trocadas entre as diversas entidades do sistema

Com base na Figura 5.1, segue-se agora uma explicação passo a passo das etapas envolvidas na execução de um serviço IMPS genérico:

1. O cliente manda uma mensagem, através de uma conexão http, no formato XML, com destino ao SAP especificando o tipo de serviço desejado.

2. O SAP é o responsável pela autenticação dos usuários e, por isso, também utiliza o JDBC para o acesso ao Banco de Dados.

3. No caso de ser um serviço que requeira a troca de mensagens instantâneas, o SAP transfere o pedido de serviço para o Servidor de Mensagens Instantâneas, utilizando uma chamada RMI. Este servidor também retorna resposta ao SAP via RMI.

4. O Servidor de Mensagens Instantâneas, em resposta à chamada do SAP, realiza uma busca por mensagens no banco de dados ou então atualiza este banco com uma nova mensagem. Tais consultas e atualizações são realizadas por meio da linguagem JDBC.

5. No caso do cliente estar requisitando um serviço que exija o processamento de informações de presença, o SAP irá realizar uma chamada RMI para o Servidor de Presença, o qual, após o tratamento da requisição e eventual processamento retorna a reposta para o SAP também via chamada RMI.

6. Toda vez que o Servido de Presença percebe a necessidade de buscar uma nova informação relativa à presença dos usuários, ou no caso de atualização das mesmas, ele realiza uma consulta no Banco de Dados comum ao sistema. Caso haja necessidade, o Servidor de Presença também é capaz de atualizar o Banco de Dados, com informações inseridas pelo usuário. Tais consultas e atualizações são também realizadas por meio da linguagem JDBC.

7. Paralelo às etapas acima, o LCS já está em funcionamento. Ele é o responsável por adquirir, junto aos AP's, os endereços MAC de todos os usuários logados no sistema. Estas informações são adquiridas através das páginas HTML, encontradas via http, fornecidas para configuração de cada AP e são de caráter proprietário[13].

8. Além de obter as informações dos AP's, o LCS também é responsável por atualizar continuamente o Banco de Dados com estas informações. Assim, de acordo com a freqüência de atualização especificada, o Banco de Dados conterá a todo instante a localização em termos de AP atualizada de cada um dos usuários do sistema.

Baseado nessa arquitetura, a seqüência lógica de funcionamento operacional do sistema pode ser separada nas seguintes etapas: Inicialização, Monitoramento dos AP´s, Autenticação do Cliente, Menu de Opções e, finalmente, *Polling*.

5.3.1. Inicialização

Para que o ambiente possa funcionar de forma adequada é necessário que, inicialmente, todos os servidores sejam inicializados. Assim, o cliente, primeiramente, deve iniciar o emulador e optar pelo MIDlet de nome WVClient, o qual contém o código-cliente do projeto, observando a Figura 5.2.

[13] Significa que a forma como a página HTML para configuração dos AP's é disponibilizada varia de fabricante para fabricante, não existindo um padrão para disponibilização dessa informação.

Figura 5.2 – Tela para escolha do MIDlet

Para que o SAP funcione corretamente deve-se inicializar o *servlet* através do servidor Tomcat, responsável pelas chamadas locais. A inicialização correta do *Servlet* pode ser visualizada na Figura 5.3.

Figura 5.3 – Inicialização do Servlet

Semelhante ao SAP, o *Presence* deve ter a interface *Presence-SAP* inicializada para que a comunicação entre essas entidades se efetive de ambos os lados. Dessa inicialização resulta a interface gráfica mostrada na Figura 5.4.

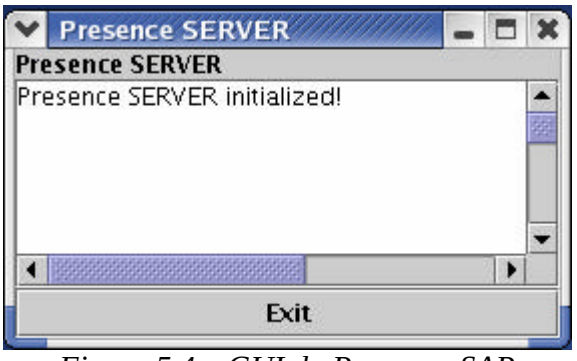

Figura 5.4 – GUI do Presence-SAP

Para se inicializar o LCS basta executar o programa Java, responsável pelo monitoramento dos AP's, sobre a Máquina Virtual. Tal inicialização resulta na tela mostrada na Figura 5.5.

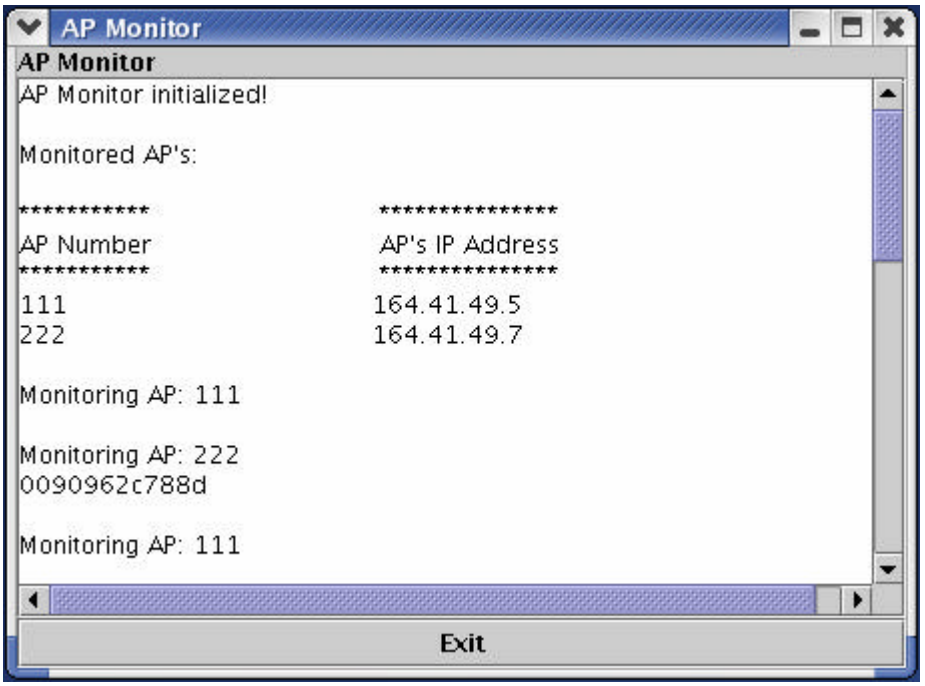

Figura 5.5 – Interface de monitoramento dos AP's realizada pelo LCS

Da Figura 5.5 pode-se ver o endereço IP de cada *Access Point* e embaixo de cada um, se for o caso, o endereço MAC dos usuários associados.

5.3.2. Monitoramento dos AP's

Nesta primeira etapa de inicialização o LCS merece atenção especial pois o mesmo é responsável, antes de tudo, por monitorar todos os AP's a fim de encontrar possíveis clientes associados ao sistema. Desta forma, assim que um cliente, em qualquer região da área de cobertura dos AP's, se associar a determinado AP o LCS o incluirá automaticamente no banco de dados, com uma *flag* indicando a recente associação, e aguardará pelo iminente *login*.

5.3.3. Autenticação do Cliente

Ao executar o MIDlet o usuário é solicitado a se autenticar junto à *Wireless Village*. Para que o sistema, como um todo, funcione corretamente é necessário que o Servidor saiba identificar os Clientes que acessam os serviços. Como a identificação do Cliente junto ao Servidor é feita no ato da autenticação e, além disso, todas as transações referentes a serviços de IMPS são efetuadas dentro de uma sessão que é identificada pelo *SessionID* obtido nessa autenticação, essa etapa é de suma importância.

A tela inicial do MIDlet é mostrada na Figura 5.6.

Figura 5.6 – Tela inicial do MIDlet

Nesta tela, o usuário entra com o seu *User-ID*, que é seu identificador único na Wireless Village, e sua senha. Ao selecionar o comando 'OK' é apresentada a tela da Figura 5.7.

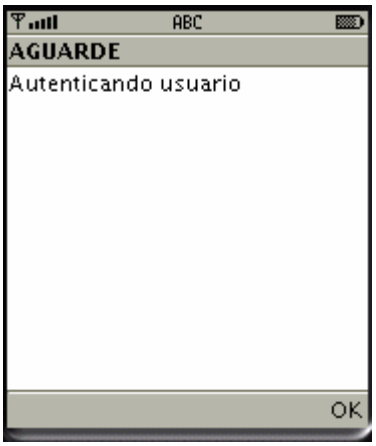

Figura 5.7 – Tela de autenticação do usuário

Neste momento o Cliente envia a mensagem (no padrão XML) *LoginRequest* ao Servidor.

Segundo o protocolo CSP, citado no Capítulo 3 e descrito em [7], a autenticação do usuário se dá a partir da seguinte transação ilustrada na Figura 5.8.

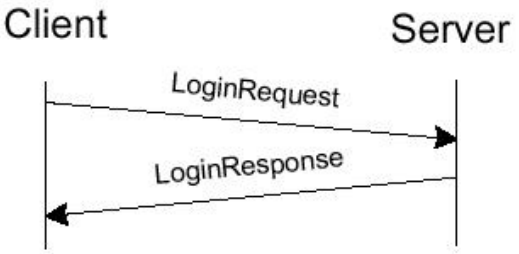

Figura 5.8 – Transação de Login, tal como especificada no protocolo CSP

Ao receber a *LoginRequest* o Servidor identifica a mensagem e acessa o banco de dados e, caso o usuário tenha entrado com *login* e senha válidos, autentica o Cliente. No caso de sucesso o Servidor envia uma mensagem de resposta (no padrão XML), *LoginResponse*, de código 200 (*Successfully logged in*) contendo o *SessionID* que, desse ponto em diante, será usado para todas as transações entre Cliente e Servidor. Caso o Cliente efetue o *login* corretamente, é apresentada ao usuário a tela da Figura 5.9.

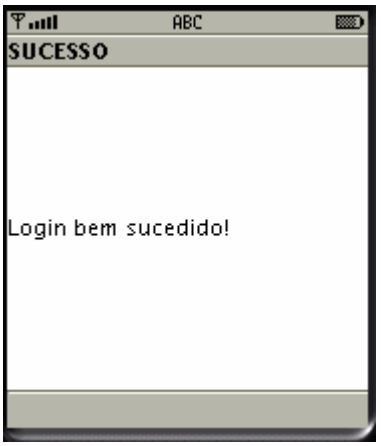

Figura 5.9 – Tela indicando Login bem sucedido

No caso de o Servidor não autenticar o usuário, a mensagem de resposta, *LoginResponse*, será sempre de código 531 (*Unknown user*.).

5.3.4. Menu de Opções

Depois da autenticação bem sucedida, é apresentada ao usuário uma lista de opções disponíveis que podem ser visualizadas na Figura 5.10.

Figura 5.10 – Tela com o Menu de Opções

5.3.4.1. Mensagem Instantânea

As três primeiras opções do menu da Figura 5.10 fazem referência aos serviços de *Instant Messaging*. Tais serviços são, muitas vezes, essenciais para complementar os serviços de presença, de forma que, uma vez detectada a localização, o serviço deve enviar uma mensagem personalizada para o cliente.

O Servidor para enviar uma mensagem para o Cliente, a faz através de uma mensagem XML que contém o elemento *MessageInfo* acrescido, dependendo do método de entrega do usuário, do elemento *ContentData.* O elemento *MessageInfo* possui sub-elementos onde estão todas as informações sobre uma mensagem instantânea:

- *MessageID* - identificador único da mensagem instantânea;
- *ContentType* - especifica o tipo de conteúdo da mensagem instantânea;
- *ContentEncoding* – codificação do conteúdo da mensagem instantânea;
- *ContentSize* - tamanho da mensagem instantânea;
- *Recipient* - destinatário da mensagem instantânea;
- *Sender* - remetente da mensagem instantânea;

- *DateTime* - data e hora de envio da mensagem instantânea;
- *Validity* - validade da mensagem instantânea.

Neste projeto foram implementadas as funcionalidades somente dos sub-elementos: *MessageID*; *ContentSize*; *Recipient*; *Sender*; *DateTime.*

Os sub-elementos: *ContentType*; *ContentEncoding*; *Validity,* existem, mas são estáticos, ou seja, são sempre preenchidos com o mesmo valor, e estão presentes apenas para atender à especificação.

Quanto ao elemento *ContentData,* este não apresenta sub-elementos e contém, unicamente, o conteúdo da mensagem.

Maiores detalhes sobre os serviços de *Instant Messaging* podem ser encontrados em [5].

5.3.4.2. Lista de Contatos

Ainda relativo à Figura 5.10, a última opção diz respeito a um dos serviços, baseados em presença, implementados no projeto e que se encontra detalhado na seção *5.4.3. Lista de Contatos no mesmo AP*.

5.3.5. Polling

Como já mencionado na introdução teórica, existem três formas do Servidor se comunicar com o Cliente para o envio de uma notificação (*notify/get*) ou de uma nova mensagem (*push*). Como o Polling é a única forma viável de se realizar tal comunicação neste projeto, foi esta a implementada.

O Polling é iniciado logo após a autenticação bem sucedida, e é interrompido somente em dois momentos: quando o usuário está lendo uma mensagem; ou quando o Cliente recebe uma resposta ao Polling diferente de *Polling-Response*, ou seja, *NewMessage* ou *MessageNotification*. O Polling é reiniciado assim que o usuário retorna à tela principal selecionando o botão 'Voltar'.

O ciclo do Polling se repete a cada 30 segundos, ou seja, a cada 30 segundos é enviada ao servidor uma mensagem (no padrão XML) de Polling.

No caso deste projeto, a resposta (do Servidor) ao Polling (do Cliente) pode ser uma dessas três:

- *Polling-Response*;
- *NewMessage*;
- *MessageNotification*.

5.3.5.1. Polling-Response

A resposta do Servidor caso não exista nenhuma mensagem instantânea não notificada (no servidor de banco de dados) é a Polling-Response. Esta resposta vai além da especificação da Wireless Village, pois a especificação não define uma resposta do Servidor neste caso.

A Polling-Response nada mais é que uma mensagem (no padrão XML) de código 200, indicando que o Polling completou seu ciclo com sucesso.

Caso exista (no servido de banco de dados) uma ou mais mensagens instantâneas não notificadas, a resposta do Servidor ao *Polling* será uma das outras duas, conforme seja o método de entrega definido pelo usuário.

A transação acontece tal como descrito na seguinte Figura 5.11.

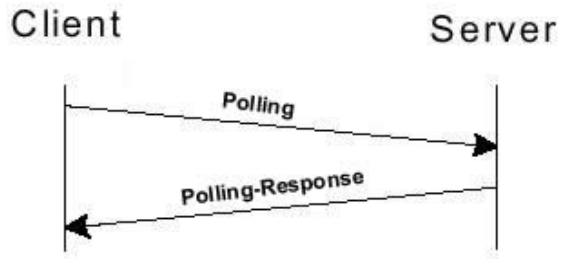

Figura 5.11 – Resposta ao Polling quando não há nova mensagem

5.3.5.2. MessageNotification

Caso o usuário tenha escolhido o método de entrega *Notify/Get* a resposta ao *Polling*, caso haja uma mensagem, seja ela oriunda do Servidor de Presença ou do Servidor de Mensagem Instantânea, não notificada (no servidor de banco de dados), será a *MessageNotification*.

A *MessageNotification* é uma mensagem (no padrão XML) que possui no seu conteúdo o elemento *MessageInfo.* Este elemento possui sub-elementos onde estão todas as informações sobre a mensagem instantânea e estão detalhados no item *5.2.4.1. Mensagem Instantânea.*

Segundo o protocolo CSP, descrito em [7], esta transação se dá tal como na Figura 5.12.

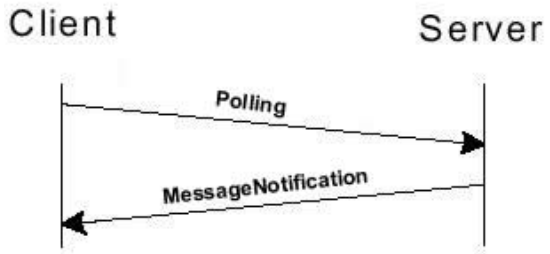

Figura 5.12 – Resposta ao Polling com indicação de nova mensagem (método Notify/Get)

Ao receber uma *MessageNotification* o Cliente verifica os sub-elementos do *MessageInfo* já mencionado e cria uma lista com apenas uma entrada equivalente ao *UserID* do remetente e o tamanho (*ContentSize*) da mensagem instantânea notificada, tal como visto na Figura 5.13.

Figura 5.13 – Tela apresentada em caso de nova mensagem (método Notify/Get)

Selecionando 'Menu' o cliente tem as seguintes opções: *ler, encaminhar* ou *apagar*. Selecionando 'Voltar' o usuário retorna à tela principal (Figura 5.10).

5.3.5.3. NewMessage

Caso o usuário tenha optado pelo método de entrega *Push* a resposta do Servidor ao Polling, no caso de haver uma mensagem instantânea não notificada (no servido de banco de dados), será a *NewMessage*.

A mensagem (no padrão XML) de resposta *NewMessage* é idêntica à *MessageNotification* acrescida de um elemento, o *ContentData*, que é o conteúdo da mensagem instantânea que é enviado juntamente com o elemento *MessageInfo.*

Esta transação é semelhante à transação *GetMessageRequest*, porém [5] não define uma resposta do Servidor à segunda mensagem (no padrão XML) do Cliente, a *MessageDelivered*, por questão de coerência foi acrescentado ao projeto esta resposta que seria a *Status*, de modo que a transação se dá de acordo com a Figura 5.14.

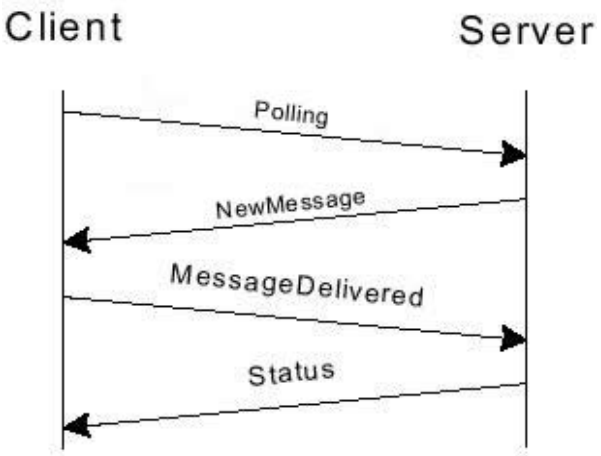

Figura 5.14 – Resposta ao Polling com indicação de nova mensagem (método Push)

Ao receber a *NewMessage* o Cliente verifica os sub-elementos do *MessageInfo* e o elemento *ContentData* e cria uma tela contendo a data de envio, o remetente e o conteúdo da mensagem, de acordo com a Figura 5.15.

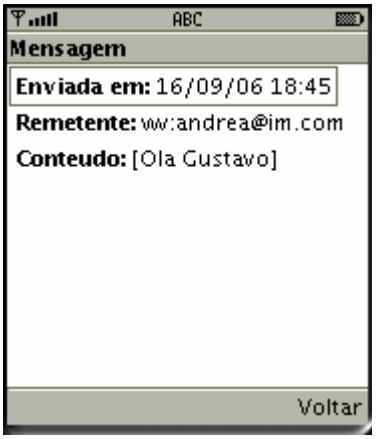

Figura 5.15 – Tela apresentada em caso de nova mensagem (método Push)

Após ler a mensagem o Cliente seleciona 'Voltar' para retornar à tela principal (Figura 5.10).

5.4. SERVIÇOS IMPLEMENTADOS

O ambiente acima descrito foi feito de forma a suportar o desenvolvimento e a implementação dos mais variados tipos de serviços IMPS. Para que um novo serviço seja adicionado ao sistema, o desenvolvedor deve criar as classes e métodos necessários para a realização das tarefas desejadas. No entanto, as tarefas básicas de envio de mensagens instantâneas e o suporte à geração e obtenção de informação de presença já estão implementados e podem ser reutilizados.

Após a criação das classes e métodos necessários, o desenvolvedor deve especificar quais mensagens deverão ser trocadas entre as entidades do sistema e adicionar tais classes às bibliotecas de cada uma das entidades utilizadas.

Seguindo essa filosofia, foram implementados três serviços que ilustram a potencialidade do Ambiente de Desenvolvimento. São eles: *Bem-Vindo, Mensagens Personalizadas* e *Lista de Contatos no Mesmo AP*.

5.4.1. BEM-VINDO

A idéia por trás do Serviço de Bem-Vindo é a de que o sistema seja capaz de identificar o momento em que o usuário se loga numa rede W-LAN e, então, enviar-lhe uma mensagem do tipo “Bem vindo ao (Lugar em que se encontra)”. Essa mensagem pode ser personalizada de acordo com o grupo que o usuário pertence, dando margem para mensagens do tipo “Caro Professor, Bem Vindo ao LEMOM”.

É evidente neste serviço a utilização de ambas as funcionalidades de um serviço IMPS: a informação de presença é necessária para se determinar o local em que o usuário se encontra; o serviço de troca de mensagens é utilizado para o envio da mensagem em si. Assim, o serviço de Bem-Vindo é uma aplicação simples, mas que ilustra bem o funcionamento do sistema como um todo.

Para a implementação desse serviço, utilizou-se do *polling*, mencionado na *seção 5.2.5. Polling*, para o estabelecimento da comunicação SAP-Cliente e, conseqüente envio, se for o caso, da mensagem de boas vindas. Nesta mensagem, por uma decisão de projeto, optou-se pelo método *push* para que o cliente veja a mensagem imediatamente.

As transações realizadas para o funcionamento desse serviço seguem aquelas da Figura 5.1 e podem ser detalhadas da seguinte forma:

1. O cliente envia a mensagem XML *Polling-Request* para o SAP e fica na espera de uma resposta que, para o serviço em questão, pode ser de três tipos. Primeiro, um *Polling-Response*, indicando que não haviam nem mensagens instantâneas nem mensagens de presença. Segundo, se tiver ocorrido uma mudança de área recente, um *MessageNotification* relativo a possíveis mensagens de boas vindas ao local entrado. Por último, caso existam mensagens instantâneas não notificadas, recebe-se um *MessageNotification* relativo as mesmas.

2. O SAP, através do algoritmo de reconhecimento de chamadas XML, reconhece o *Polling-Request* e se encarrega de obter informações para remeter a resposta. Primeiramente, dado que o cliente já se autenticou no sistema e tem um *SessionID*, o SAP faz uma consulta ao banco de dados para obter atributos do cliente. Dessa forma, obtêm-se, a partir do *SessionID*, o *UserID*, o *GroupID* e o *MACAddress*.

3. De posse dos atributos do cliente, o SAP busca no Servidor de Mensagem Instantânea, através de uma chamada RMI, a quantidade total de mensagens instantâneas não notificadas.

4. O Servidor de Mensagem Instantânea, em resposta à chamada RMI proveniente do SAP, faz uma consulta JDBC ao banco de dados do sistema. Esta consulta é feita à tabela *Message* e, simplesmente, conta-se o número total de mensagens que apresentam tanto o destionatário, *Recipient*, em questão como a flag de notificação, *Notified*, falsa. O resultado dessa contagem é, então, interpretado pelo SAP.

5. O SAP armazena, para posterior decisão de resposta, o resultado proveniente do Servido de Mensagem Instantânea e realiza uma chamada, também RMI, ao Servidor de Presença. Tal chamada envolve mais partes e começa por obter os atributos *AreaChanged*, flag que indica se houve mudança de área, e *APNumber*. Com esses dados iniciais, caso a primeira informação seja verdadeira, o SAP verifica se existe mensagem de bem-vindo para o grupo do usuário.

6. O Servidor de Presença, em resposta à chamada RMI proveniente do SAP, faz, assim como o servidor de Mensagens Instantâneas, uma consulta JDBC ao banco de dados do sistema. Esta consulta é feita de acordo com o *AreaChanged*, caso este seja verdadeiro a consulta restringe-se a contagem do número total de mensagens que apresentam tanto o destionatário, *Recipient*, igual ao *GroupID* do usuário em questão como o *APNumber* correspondente ao AP do cliente. Caso seja falso, o Servidor retorna que não existem mensagens de presença.

Por fim, após essas transações, o SAP já contém todas as informações necessárias para enviar uma resposta à solicitação de *polling* do Cliente. Assim, seguindo o algoritmo implementado pelo *servlet*, tem-se para o cliente as possibilidades descritas no diagrama da Figura 5.18.

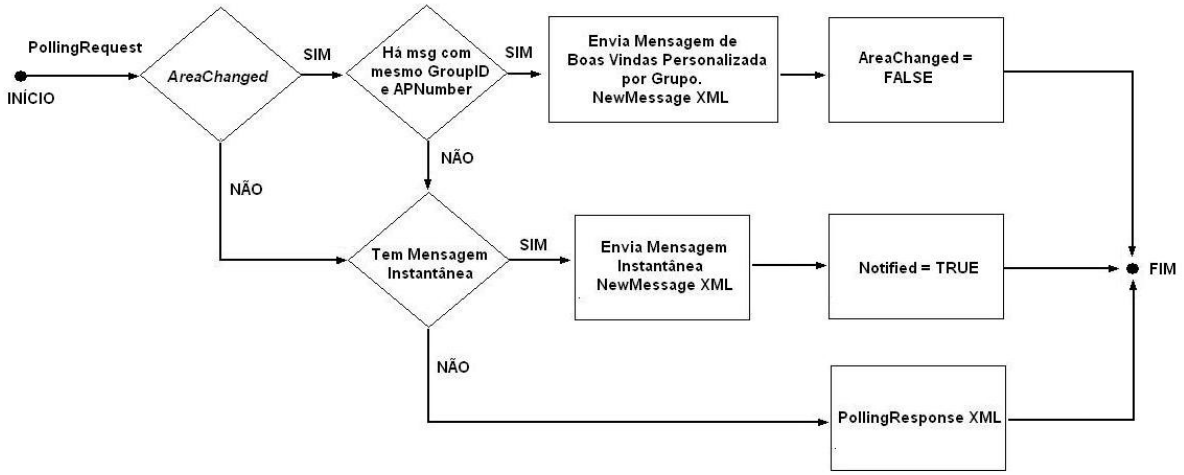

Figura 5.18 – Diagrama de seqüência do polling.

Assim, de acordo com o Diagrama da Figura 5.18, o processo pode finalizar de três formas distintas. Nas duas primeiras, uma mensagem é apresenta no celular do cliente, enquanto a última, por não ter nenhuma mensagem a ser entregue, é imperceptível para o usuário.

Um exemplo de tela do Serviço "Bem-Vindo" em execução pode ser vista na Figura 5.19.

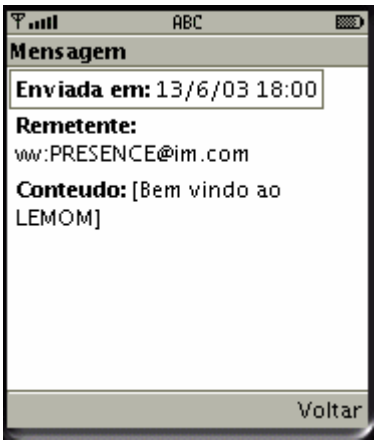

Figura 5.19 – Tela de exemplo do Serviço Bem-Vindo.

5.4.2. MENSAGENS PERSONALIZADAS

O serviço de Mensagens Personalizadas atua de forma semelhante ao serviço de Bem-Vindo. A diferença entre ambos está no fato que, ao invés de enviar apenas uma mensagem fixa e genérica, sem um destinatário conhecido, como o serviço de Bem-Vindo, o serviço de Mensagens Personalizadas permite ao administrador do sistema enviar, tal como o próprio nome informa, mensagens personalizadas, endereçadas à um usuário específico.

Analisando as transações enumeradas no caso do serviço de Bem Vindo, de acordo com a Figura 5.1, elas ocorrem exatamente da mesma forma no caso do serviço de Mensagens Personalizadas. Verificam-se acréscimos de funcionalidades apenas em duas transações descritas abaixo:

5. Nesta transação, no momento de verificar a flag *AreaChanged*, o sistema conta com uma nova possibilidade. Caso essa flag seja falsa, ou seja, possíveis mensagens de boas vindas já foram entregues, o SAP verifica a quantidade de mensagens endereçadas para um usuário específico.

6. Para poder lidar com essa nova verificação do SAP, o Servidor de Presença teve que sofrer os seguintes ajustes.Caso a *flag* de *AreaChanged* seja falsa, antes de retornar que não existem mensagens de presença, faz-se uma consulta que, basicamente, restringe-se a contagem do número total de mensagens que apresentam tanto o destionatário, *Recipient*, igual ao *UserID* do usuário em questão como o *APNumber* correspondente ao AP do cliente.

Com essas alterações no sistema, o diagrama de resposta ao *PollingRequest* passa a contar com uma nova possibilidade que pode ser melhor visualizada no diagrama da Figura 5.20.

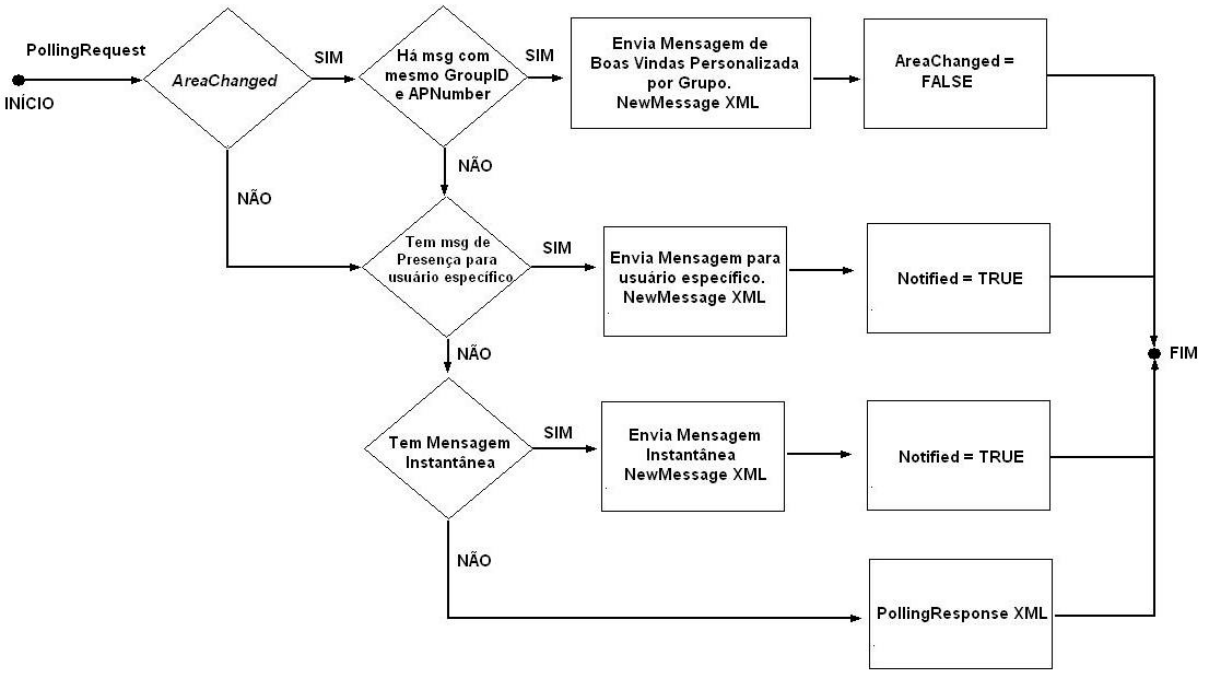

Figura 5.20 – Diagrama de seqüência do polling com o serviço de Mensagens Personalizadas.

Como pode ser visto da Figura 5.20, antes de verificar a existência de mensagens instantâneas, o sistema procura por mensagens específicas para o usuário em questão, que no caso afirmativo a retorna no celular do cliente.

Um exemplo de tela do Serviço "Mensagens Personalizadas" em execução pode ser vista na Figura 5.21.

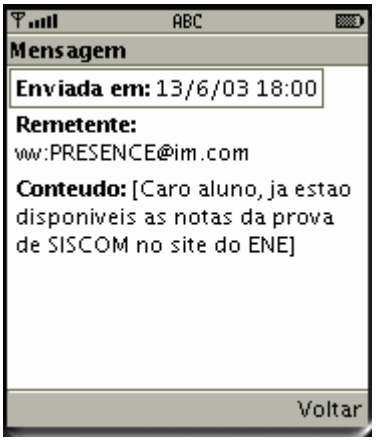

Figura 5.21 – Tela de exemplo do Serviço Mensagens Personalizadas.

5.4.3. LISTA DE CONTATOS NO MESMO AP

Outro serviço interessante e de alta viabilidade em um ambientes comercial é a Lista de Contatos no mesmo AP. Tal serviço visa permitir que, ao entrar em uma ESS, o usuário do sistema seja capaz de receber uma lista de todos os clientes que se encontram no mesmo AP que ele e, ao selecionar um desses clientes da lista, possa enviar uma mensagem instantânea.

Como utilidade desse serviço pode-se mencionar o aumento na dinamicidade dos empregados de uma grande corporação. Esses empregados teriam maior facilidade para marcar reuniões, verificar funcionários na região de proximidade, além da possível monitoração de clientes não pertencentes ao setor.

Para a implementação desse serviço foram utilizados, diferentemente do serviço de Bem Vindo, dois passos distintos. No primeiro, o serviço, a partir da solicitação do cliente, busca a lista de contatos através do Servidor de Presença e a envia, via XML, para o Cliente. No segundo passo, visto que o cliente já tem uma lista com todos os usuários associados ao mesmo AP que ele, caso ele selecione um destes, envia-se, através do Servidor de Mensagem Instantânea, uma mensagem para o usuário selecionado.

De acordo com o primeiro passo, as transações realizadas para o funcionamento desse serviço seguem aquelas da Figura 5.1 e podem ser detalhadas da seguinte forma:

1. O cliente envia, a partir da seleção no *Main Menu* do item *Carregar a Lista de Contatos*, uma mensagem XML *ListManage-Request* para o SAP através do método *getListRequest* da classe `PresenceRequest`. Feito o envio, o cliente fica na espera da lista para se encerrar a conexão HTTP.

2. O SAP, através do algoritmo de reconhecimento de chamadas XML, reconhece o *ListManage-Request* e se encarrega de obter informações para remeter a resposta. Primeiramente, dado que o cliente já se autenticou no sistema e tem um *SessionID*, o SAP faz uma consulta ao banco de dados para obter atributos do cliente. Dessa forma, obtêm-se, a partir do *SessionID*, o *UserID*, o *GroupID* e o *MACAddress*.

5. Como não há, inicialmente, conexão com o Servidor de Mensagem Instantânea, segue-se diretamente para a transação 5. Assim, o SAP realiza uma chamada, via RMI, ao Servidor de Presença para obter a lista de contatos.

6. O Servidor de Presença, em resposta à chamada RMI proveniente do SAP, faz uma consulta JDBC ao banco de dados do sistema. Estas consultas são feitas através do método *updateContactList* da classe `PresenceSAPInterfaceImpl` e começam por obter, através do método *getMACAddressFromUserID* da classe `Presence_LCS`, o endereço MAC do usuário requisitante. Em seguida, utilizando sempre a classe `Presence_LCS`, obtém-se, através do método *getAPNumber* e do endereço MAC obtido, o número do AP ao qual o usuário está associado. Na seqüência, obtém-se, através do método *getNAPContacts* e do número do AP obtido, o número de usuários da lista requisitada. Através do método *getMACAddressFromAPNumber*, do número do AP obtido, do número de usuários da lista e do endereço MAC do usuário, obtém-se os endereços MAC dos usuários da lista. Por fim, com o método *getContacts*, com a lista dos endereços MAC dos usuários da lista e com o número de contatos, obtém-se a lista. Uma vez obtida a lista ela é processada pelo SAP.

Para efetivar essa primeira etapa, o SAP envia a lista obtida para o cliente através do XML *ListManage-Response* e, assim, fecha-se a conexão iniciada pelo cliente. Com isso, o cliente processa o XML recebido e apresenta a lista na tela do celular.

Um exemplo de tela do Serviço "Lista de Mensagens no Mesmo AP" em execução pode ser vista na Figura 5.22.

Figura 5.22 – Tela de exemplo do Serviço "Lista de Contatos no Mesmo AP".

Finalizada essa etapa, o cliente inicia a segunda ao selecionar um dos usuários da lista. Quanto a essa etapa de envio da mensagem, maiores detalhes podem ser observados em [5] e visualizados na Figura 5.23.

Figura 5.23 – Tela para Envio de IM do Serviço "Lista de Contatos no Mesmo AP".

6. CONCLUSÃO

O desenvolvimento de serviços baseados em presença e em troca de mensagens instantâneas – IMPS – é, sem dúvida, um filão a ser explorado com interesse, devido à crescente demanda por este tipo de serviço e, além disso, pela possibilidade que o mesmo representa na alavancagem dos lucros das operadoras. Ainda, a franca expansão das redes *wireless* tanto em ambientes acadêmicos quanto em comerciais e corporativos vem a acrescentar o movimento em direção à mobilidade que estamos presenciando.

Nesse sentido, os objetivos deste trabalho foram alcançados, na medida em que o mesmo contribui para o referido movimento, uma vez que disponibiliza uma tecnologia inovadora e totalmente portável para a confecção de aplicações e serviços IMPS, baseados na mais popular plataforma de redes *wireless*, o IEEE 802.11.

Essa tecnologia é inovadora, no sentido em que fornece uma base para o desenvolvimento de tais serviços, dando suporte ao envio de mensagens instantâneas e adquirindo e disponibilizando informações de presença baseadas em qual AP cada usuário está logado no momento. É portável no sentido em que segue a especificação da iniciativa *Wireless Village*, a qual, tal como foi mencionado, é uma tentativa realizada pelas maiores empresas do ramo de comunicações *wireless* para desenvolver um padrão único a ser seguido por todos os desenvolvedores.

A idéia original que motivou o projeto foi o desenvolvimento de um sistema de localização que fosse capaz de determinar, baseado no em redes WLAN, a real localização do usuário, utilizando um esquema parecido com o E-OTD para redes GSM, porém retirando informações do sinal de pelo menos três AP's, ao invés de três ERB's. No entanto tal idéia não se mostrou viável visto que não foi possível a medição da potência do sinal recebido de cada AP, característica imprescindível para a implementação do sistema. Dessa forma, decidiu-se pela obtenção de informação de presença, a qual é menos precisa que informação de localização, mas que também possibilita o desenvolvimento de diversos serviços.

Uma outra dificuldade encontrada foi a inexistência, atualmente, de dispositivos móveis pessoais que utilizem a faixa de freqüência descrita no padrão IEEE 802.11. Só se encontram disponíveis comercialmente, hoje, cartões PCMCIA que interfaceiam com AP's para uso em Laptops ou outros dispositivos do gênero. Assim, como a filosofia de projeto era

o desenvolvimento de aplicações para dispositivos tais como celulares, optou-se por utilizar uma configuração na qual um simulador de telefone celular roda em um laptop, e este se comunica com o AP através de um cartão WLAN.

Um outro problema encontrado foi o fato de que o *handoff* entre AP's só é possível para AP's que estejam no mesmo ESS. Dessa forma, fica impossível a comunicação entre servidores que estejam em redes diferentes. Como forma de sanar este problema, tanto o SAP como o Cliente e os AP's receberam endereços IP fixos roteáveis e pertencentes à mesma rede. Assim, foi possível estabelecer a comunicação entre os diversos elementos do sistema e garantir a operabilidade do mesmo.

No entanto, este problema toma proporções maiores à medida que o sistema crescer horizontalmente e for necessária a inclusão de novos servidores Wireless Village Server. Para que os mesmos possam se comunicar adequadamente, utilizando o protocolo SSP, é necessário que os mesmos possuam endereços IP fixos roteáveis e que os Clientes que com ele se comunicam (utilizando o protocolo CSP) estejam na mesma sub-rede.

Como desafios futuros, pode-se citar a implementação de novos servidores WV Server, os quais seriam responsáveis pela expansão do sistema e que possibilitariam o desenvolvimento de novos serviços. Além disso, uma vez que o presente trabalho não contempla características descritas na especificação WV tais como o serviço de Grupo de Conteúdo Compartilhado, uma tarefa futura seria a implementação de tais funcionalidades.

Finalmente, uma vez que este trabalho demonstra a implementação de um Ambiente para Desenvolvimento de Serviços baseados em Presença, um próximo passo lógico seria o desenvolvimento de novos serviços, tais como os já mostrados aqui, e que utilizassem as novas funcionalidades descritas acima.

BIBLIOGRAFIA

[1] Discretix Technologies, Ltd: http://www.discretix.com/background.htm

[2] [*WV-WP] Wireless Village – White Paper, 2002,* http://www.openmobilealliance.org/wirelessvillage/docs/WV_White_Paper.pdf

[3] *Trapeze Networks*, http://www.trapezenetworks.com/tech/tech_tut.asp

[4] GAST M., *802.11 Wireless Networks – The Definitive Guide*, O'Reilly, 2002

[5] OLIVEIRA A. F., CARVALHO L. A., *Instant Messaging para Wireless Village em J2ME*, Dissertação de Projeto Final de Graduação da Universidade de Brasília, 2003

[6] XML Files – Extensible Markup Language, http://www.xmlfiles.com/

[7] *[WV-CSP] Wireless Village – Client-Server Protocol, Session and Transactions,* 2002, http://www.openmobilealliance.org/wirelessvillage/docs/WV_CSP_v1.1.pdf

[8] HORSTMANN C. S. , CORNELL G., *Core Java 2, Volume 1 – Fundamentos*, Makron Books, 2001

[9] HORSTMANN C. S., CORNELL G., *Core Java 2, Volume 2 – Recursos Avançados*, Makron Books, 2001

[10] DEITEL H.M., DEITEL P. J., SANTRY S. E., *Advanced Java 2 Platform – How to Program*, Prentice Hall, 2001

[11] ASHRI R., ATKINSON S., AYERS D., HAGLIND M., RAY B., MACHIN R., NASHI N., TAYLOR R., WIGGERS C., *Java Mobile Programming*, Wrox Press, 2001

[12] KNUDSEN J., *Wireless Java: Developing with Java 2, Micro Edition*, Apress, 2001

[13] MCLAUGHLIN B., *Java & XML Data Binding*, O'Reilly, 2002

[14] STALLINGS W., *Wireless Communications and Networks*, Prentice Hall, 2001

[15] STEELE R., LEE C. C., GOULD P., *GSM, cdmaOne and 3G Systems*, John Wiley & Sons Ltd., 2001

[16] LARMAN C., *Utilizando UML e padrões*, Bookman, 2000

[17] OLIVEIRA A. H. C., CORREIA H. B., IIDA R. F., *Implementação de uma interface neutra cliente-servidor dedicada à simulação e dos módulos de sistemas móveis e circuitos elétricos*, Dissertação de Projeto Final de Graduação da Universidade de Brasília, 2002

[18] CUSSIOLI T. J. S., RIBEIRO FILHO W., *Aplicações utilizando sistemas de localização em GSM*, Dissertação de Projeto Final de Graduação da Universidade de Brasília, 2003

[19] Tecnologia Java, http://java.sun.com

[20] *XML 1.0 Recommendation*, http://www.w3.org/TR/REC-xml

[21] *JDBC Data Access API,* http://java.sun.com/products/jdbc/related.html

[22] *Wireless Java,* http://wireless.java.sun.com/

[23] ROUSELL T., *"Java Technology: The Kingpin of Mobile Monetization."*, http://wireless.java.sun.com/developers/business/articles/kingpin/

[24] *The Wireless Village Initiative,* http://www.openmobilealliance.org/wirelessvillage/

[25] *[WV] Wireless Village – System Architecture Model*, 2002, http://www.openmobilealliance.org/wirelessvillage/docs/WV_Architecture_v1.1.pdf

[26] *[WV-FF] Wireless Village – Features and Functions specification*, 2002, http://www.openmobilealliance.org/wirelessvillage/docs/WV_Features_Functions_v1.1.pdf

[27] *[WV-PA] Wireless Village – Presence Attributes specification*, 2002, http://www.openmobilealliance.org/wirelessvillage/docs/WV_PA_v1.1.pdf

[28] *[WV-PA-DTD] Wireless Village – Presence Attributes, DTD and Examples*, 2002, http://www.openmobilealliance.org/wirelessvillage/docs/WV_PA_DTD_v1.1.pdf

[29] *MIDP Datasheet*, 2003, http://java.sun.com/products/midp/midp-ds.pdf

[30] *J2ME Datasheet*, 2003, http://java.sun.com/j2me/j2me-ds.pdf

[31] VAN PEURSEM J., *Mobile Information Device Profile v2.0 (JSR-118) Final Specification*, 2003, http://jcp.org/aboutJava/communityprocess/final/jsr118/

[32] TAIVALSAARI A., *CLDC Specification V1.1 (JSR-139) Final Specification*, 2003, http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html

[33] MCLAUGHLIN B., *Java & XML*, O'Reilly, 2001

[34] World Wide Web Consortium (W3C), http://www.w3c.org

[35] Web site de Rafael Vidal Aroca, http://www.geocities.com/rafaelaroca/doc/mysql.html

[36] DEITEL H.M., DEITEL P. J., JAVA Como Programar, 4ª edição, Bookman, 2003

[37] MAHMOUD Q., *Learning Wireless Java*, O'Reilly, 2002