Implementação de terminal para redes ad hoc sem fio utilizando soluções de software livre

Sérgio M. Sakai Danilo L. Halla Paulo H. M. Santos Ricardo Takaki Marcelo C. F. Leal José A. Martins

Resumo—**Este artigo descreve a implementação de um protótipo de terminal para redes ad hoc sem fio, utilizando *software* livre. Os principais requisitos desse terminal são: apresentar baixo custo e possibilitar transmissão de sinal de voz. Isso levou à utilização preferencial de soluções de *software* livre e/ou livre de pagamentos de *royalties*. A plataforma de hardware utilizada é de baixo custo e baixo consumo de energia, sendo baseada no processador OMAP da Texas. O desenvolvimento desse terminal envolveu a implementação de plataforma de *software*, codificadores de voz, protocolo de roteamento e agente para gerência de redes.**

***Palavras-Chave*—redes ad hoc, terminal, implementação, *software* livre, *firmware*.**

***Abstract*—This article describes an implementation of a low cost terminal for ad hoc networks, employing free software. This terminal employs voice over IP technology and it presents the functionalities to network routing and remote management. As a low cost terminal, free software solutions and/or royalties free payment were used.**

***Keywords*—ad hoc network, terminal, implementation, free software, firmware**

I. INTRODUÇÃO

Redes ad hoc sem fio são redes em que os terminais (nós) se comunicam utilizando enlaces sem fio, sem a necessidade de infra-estrutura [1]. Assim, nesse tipo de rede, os próprios terminais são responsáveis pela organização e manutenção da rede.

Para possibilitar a comunicação entre quaisquer terminais da rede, os terminais devem realizar a função de roteamento, tarefa tipicamente desempenhada por roteadores dedicados em redes com fio. Portanto, os terminais são responsáveis por descobrir com quais terminais podem se comunicar diretamente, quais as rotas para comunicação com outros terminais e por encaminhar o tráfego gerado por outros terminais.

Entre as principais características das redes ad hoc sem fio estão: topologia dinâmica, limitação de uso de potência pelos terminais, segurança limitada e limitação de largura de banda (comunicação sem fio).

As redes ad hoc sem fio são redes de transmissão por pacotes e podem prover serviços de voz e dados. Para os serviços de voz pode-se utilizar a tecnologia de voz sobre IP (*Internet Protocol*) (VoIP (*Voice over IP*)).

Sérgio M. Sakai Danilo L. Halla, Paulo H. M. Santos, Ricardo Takaki, Marcelo C. F. Leal e José A. Martins, Fundação CPqD, Campinas, E-mails: sakai@cpqd.com.br, dhalla@cpqd.com.br, psantos@cpqd.com.br, rtakaki@cpdq.com.br, mleal@cpqd.com.br, martins@cpqd.com.br. Este trabalho foi financiado pelo FUNTTEL.

O desenvolvimento de um terminal ad hoc envolve a implementação de praticamente todas as camadas, desde aplicação até a camada física, não sendo uma tarefa fácil.

Um desafio a ser enfrentado é o desenvolvimento de um terminal de baixo custo e de baixo consumo de potência sem que isso afete a qualidade do produto final, o que implica restrições de recursos computacionais e preferência pela utilização de algoritmos menos complexos. Uma opção para facilitar o desenvolvimento e favorecer o requisito de baixo custo é a utilização de soluções de *software* livre.

O desenvolvimento desse terminal faz parte do desenvolvimento de um sistema de telecomunicação baseado em arquitetura de redes ad hoc, suportando serviços de voz e dados com custo reduzido e focado nas necessidades e características do mercado brasileiro. Esse sistema possibilitará a inclusão digital da população brasileira não atendida pelos sistemas de telecomunicações atualmente em uso, como pequenas localidades, áreas rurais e bairros afastados.

Este trabalho descreve a implementação de um protótipo de terminal para redes ad hoc sem fio, utilizando *software* livre. Essa implementação está sendo utilizada como prova de conceito e contribuirá para o desenvolvimento de um terminal ad hoc de baixo custo. Ela utiliza uma plataforma de hardware de baixo custo e consumo baseada no processador OMAP da *Texas Instruments*, rodando Linux embarcado.

Na seção II são apresentados os requisitos para o desenvolvimento do terminal ad hoc. A seção III apresenta a descrição funcional do terminal ad hoc. A seção IV descreve a plataforma de hardware utilizada no desenvolvimento do protótipo e na seção V é descrita a base de *software* implementada para o desenvolvimento. A seção VI descreve a solução implementada. As conclusões deste trabalho são apresentadas na seção VII.

II. REQUISITOS DE DESENVOLVIMENTO

Os principais requisitos a serem atendidos pela implementação do terminal ad hoc são:

- Baixo custo: para que o terminal ad hoc apresente baixo custo, deve-se utilizar algoritmos menos complexos, que não exijam grande capacidade de processamento e armazenamento, soluções de *software* livre e soluções que não necessitem de pagamento de *royalties*. Além disso, os componentes de hardware devem ser de baixo custo. Assim, os recursos de processamento e armazenamento de dados utilizados na implementação são limitados.

- Baixo consumo de potência: considerando o fato de que o terminal ad hoc poderá ser alimentado por baterias, é importante que o consumo de potência seja o menor possível.
- Flexibilidade da solução: é importante que a solução apresente flexibilidade para permitir a inclusão de novas funcionalidades ou a substituição dos algoritmos utilizados.
- Provisão de serviços de voz: para a transmissão de voz deve-se utilizar a tecnologia de transmissão de voz sobre pacotes, podendo-se empregar a tecnologia de voz sobre IP.

III. DESCRIÇÃO FUNCIONAL DO TERMINAL AD HOC

O terminal ad hoc desenvolvido apresenta as seguintes funcionalidades: transmissão e recepção de sinal de voz, roteamento, gerência do terminal ad hoc e comunicação sem fio.

A. Transmissão do sinal de voz

Esse módulo é responsável pela transmissão do sinal de voz, possibilitando a originação, o atendimento e o término de chamadas de voz sobre pacotes.

Para tanto, foi utilizada a tecnologia de voz sobre IP (VoIP).

Para o estabelecimento de sessão foi empregado o protocolo SIP (*Session Initiation Protocol*) [2], pela melhor aderência à natureza da aplicação (VoIP) a ser implantada.

Para a transmissão de pacotes em tempo real, utilizou-se o protocolo RTP (*Real Time Transport Protocol*) [3] em conjunto com o protocolo RTCP (*Real Time Control Protocol*) [3], os quais são muito utilizados em aplicações de voz em redes IP.

Para a codificação do sinal de voz, foram implementados dois codificadores: o codificador *G.711 – Pulse Code Modulation of Voice frequencies* [4] e o codificador *G.726 - 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)* [5]. Esses codificadores são padronizados pela ITU-T (*International Telecommunication Union - Telecommunication Standardization Sector*).

Eles foram escolhidos devido à simplicidade de seus algoritmos (baixa necessidade de processamento e armazenamento de dados), à boa qualidade de voz resultante, à facilidade de implementação e ao baixo atraso gerado.

O codificador G.711 amostra o sinal com freqüência de 8 kHz e utiliza 8 bits para representar cada amostra, resultando em uma taxa de bits igual a 64 kbit/s. Para a quantização das amostras são utilizadas curvas logarítmicas, as quais diminuem o erro de quantização das amostras com menor amplitude. As leis de codificação utilizadas são: Lei-A e Lei-μ. Esse codificador apresenta algoritmo de baixa complexidade e baixo atraso de codificação (menor que 0,125 ms).

O codificador G.726 trabalha em quatro taxas de bit: 16, 24, 32 e 40 kbit/s, empregando freqüência de amostragem de 8 kHz. É um codificador de forma de onda e utiliza a técnica ADPCM. Esse codificador apresenta algoritmo de baixa complexidade e baixo atraso (menor que 0,125 ms).

B. Roteamento

A função desse módulo é possibilitar o roteamento dos pacotes a serem transmitidos, identificando as rotas e atualizando as tabelas de roteamento.

Como protocolo de rede foi escolhido o protocolo IP, por ser um protocolo amplamente utilizado.

Para realizar o roteamento dos pacotes IP dentro da rede ad hoc foi empregado o protocolo AODV (*Ad hoc On Demand Distance Vector*) [6] [7], o qual foi projetado para uso em redes ad hoc com grande número de nós móveis. Esse protocolo pode lidar com taxas de mobilidade baixas, moderadas e relativamente altas, bem como uma variedade de níveis de tráfego.

O protocolo AODV é um protocolo do tipo reativo puro, em que as rotas são criadas apenas quando há necessidade, não sendo necessária a manutenção de tabelas de roteamento completas.

Uma característica do protocolo AODV é a pequena quantidade de mensagens para troca de informações entre os nós, o que o torna bastante atraente para os requisitos propostos.

C. Gerência do terminal ad hoc

A função desse módulo é coletar e disponibilizar informações de gerência do terminal ad hoc para tratamento pelo sistema de gerência.

Para a implementação desse módulo foi utilizado o protocolo SNMP (*Simple Network Management Protocol*) [8] [9], sendo empregado um agente SNMP implementando a MIB (*Management Information Base*) padrão MIB II [10].

O agente SNMP é um módulo de *software* residente no terminal ad hoc responsável por coletar e armazenar informações de gerência, traduzindo-as para um formato compatível com o protocolo de gerência SNMP.

Esse agente implementa e mantém uma estrutura em árvore de objetos (MIB) que representa as informações de gerência coletadas. Um agente pode implementar várias MIBs.

D. Comunicação sem fio

Esse módulo é responsável pelo acesso ao meio e pela transmissão e recepção do sinal via rádio. Para a implementação desse módulo foi escolhido o padrão IEEE 802.11b [11] [12], o qual possibilita a operação em modo ad hoc.

Esse padrão é uma extensão do padrão IEEE 802.11 [11] e especifica as camadas física e de controle de acesso ao meio para sistemas WLAN (*Wireless Local Area Network*), sendo um padrão largamente aceito e implantado no mundo.

O padrão IEEE 802.11b possui dois modos de controle de acesso: o distribuído, ou modo *Distributed Coordination Function* (DCF), e o centralizado, ou modo *Point Coordination Function* (PCF). O modo DCF suporta a operação no modo ad hoc, possibilitando a comunicação ponto-a-ponto entre dois terminais sem a necessidade de uma unidade de controle.

As principais características desse padrão são:

- Taxas de transmissão: 1, 2, 5,5 e 11 Mbit/s

- Faixa de operação: 2,4 a 2,497 GHz
- Número de canais: 11
- Largura de banda do canal: 22 MHz
- Potência máxima de transmissão: 1 W
- Utilização da técnica *Direct Sequence Spread Spectrum* (DSSS), empregando seqüência de Barker para as taxas de 1 e 2 Mbit/s e seqüência CCK (*Complementary Code Keying*) para taxas de 5,5 e 11 Mbit/s.
- Técnicas de modulação: DBPSK (*Differential Binary Phase Shift Keying*) para a taxa de 1 Mbit/s e DQPSK (*Differential Quadrature Phase Shift Keying*) para taxas de transmissão de 2, 5,5 e 11 Mbit/s.
- Mecanismo de acesso ao meio (MAC (*Medium Access Control*):
    - *Carrier Sensing Multiple Access with Collision Avoidance* (CSMA/CA)
    - Mecanismo de *backoff time*
    - *Positive Acknowledgement* (ACK)
    - Mensagens de controle *Request-to-Send* (RTS) e *Clear-to- Send* (CTS)

IV. PLATAFORMA DE HARDWARE

Para realizar o processamento de sinais em tempo real, como requer um serviço de voz e, ao mesmo tempo, executar todos os serviços de rede e gerenciamento necessários, adotou-se a abordagem de utilização de dois núcleos diferentes de processamento, um de arquitetura DSP (*Digital Signal Processor*) e outro de arquitetura RISC (*Reduced Instructions Set Computer*). O DSP é utilizado para as atividades de processamento de sinais em tempo real e o RISC voltado para as demais atividades.

Visando uma redução de consumo de energia, optou-se pela utilização de um processador híbrido, contendo um núcleo DSP e um núcleo RISC ARM 9 (*Advanced Risc Module*) [13] em um único encapsulamento. Nesse sentido, foi utilizado o OMAP™5910 da *Texas Instruments* [14][15][16], processador que já vem sendo empregado com sucesso em outras soluções que exigem processamento de sinais em tempo real, operações lógicas complexas e baixo consumo de energia, como telefones celulares e PDAs (*Personal Digital Assistant*).

O núcleo RISC do OMAP possui uma capacidade de processamento de 150 MIPS, operando a 150 MHz, e o núcleo DSP do OMAP possui uma capacidade de processamento de 300 MIPS, operando em 150 MHz.

Para a implementação do terminal foi utilizada como plataforma de hardware, o kit de desenvolvimento *Innovator* [17] da *Texas Instruments*.

O *Innovator* dispõe de interfaces diversas para periféricos, como USB (*Universal Serial Bus*), RS-232, ethernet, microfone, alto-falante e *display*. Como recursos de memória, dispõe de 32 MB de memória *FLASH* e 32 MB de memória RAM, que somam-se aos 16 kB de memória *cache* para instruções e 8 kB de memória *cache* para dados do RISC, mais 80 kB de memória interna e 24 kB de memória *cache* para instruções do DSP.

V. PLATAFORMA DE SOFTWARE

Seguindo os requisitos de baixo custo do terminal, facilidade de desenvolvimento, maturidade da solução, aderência ao modelo de aplicações embarcadas e disponibilidade de recursos para a plataforma de hardware, a escolha do Sistema Operacional do núcleo de processamento RISC recaiu sobre o Linux.

A versão utilizada do kernel do Linux foi a 2.4 [18], devido à maior estabilidade dessa versão para a plataforma escolhida.

Como plataforma de desenvolvimento, foi gerado, a partir dos códigos fonte, um kit de desenvolvimento (SDK (*Software Development Kit*)) voltado para a plataforma ARM 9 baseado no conjunto de ferramentas GNU [19] para desenvolvimento e depuração de aplicativos.

Para o núcleo de processamento DSP, utilizou-se como sistema operacional o sistema DSP/BIOS [20][21] da T*exas Instruments*, que atende aos requisitos de operação *hard realtime* e que, apesar de ser uma solução proprietária, não inclui r*oyalties* ao custo final da solução.

Como plataforma de desenvolvimento, foi utilizado o *Code Composer Studio* [22] da *Texas Instruments*, ferramenta proprietária específica para desenvolvimento e depuração de aplicativos voltados para processadores da *Texas Instruments.*

A. Soluções de software livre

Considerando o requisito de baixo custo do terminal ad hoc, buscou-se utilizar soluções já existentes de *software* livre.

No ambiente DSP/BIOS, não existem soluções de *software* livre disponíveis. Assim, foi necessário o desenvolvimento e implementação de todo *firmware* para o DSP, a partir dos *reference frameworks* de *software* RF5/RF6 da *Texas Instruments*.

No ambiente Linux existe uma vasta gama de soluções livres já desenvolvidas e testadas. No entanto, a maioria dessas soluções foi desenvolvida para operar em ambientes PC ou acima, em termos de capacidade de hardware.

Assim, foi necessária uma seleção criteriosa das soluções a serem efetivamente empregadas e, eventualmente, adaptadas e configuradas para se adequarem ao ambiente, mais limitado, do terminal ad hoc. Somente após a realização das adaptações, foi realizada a compilação da solução para o ambiente do terminal ad hoc.

Como solução de *Shell*, evitou-se a utilização de implementações pesadas como o *Bash*. Foi utilizada uma solução mais leve e apropriada para ambientes embarcados, o *Busybox* [23]. No caso, a versão 1.0 do *Busybox*.

Na implementação do protótipo, optou-se por utilizar uma interface de telefone simulada através da utilização de bibliotecas gráficas.

A utilização dessa interface gráfica é feita via teclado e *mouse* ou *display touchscreen*.

B. Implementação do ambiente de software

Para a implantação do ambiente de desenvolvimento do DSP, foi utilizado o *software* fornecido pela *Texas Instruments,* o qual já é fornecido com as configurações necessárias para o desenvolvimento de aplicações para a plataforma.

O ambiente de *software* do RISC necessitou ser adaptado, configurado e compilado para o terminal ad hoc.

As ferramentas de desenvolvimento foram geradas a partir dos códigos fonte originais para as soluções de desenvolvimento gcc [24], gdb [25], binutils [26] e glibc [27] da GNU.

A partir do kernel original do Linux 2.4, foram inseridas as configurações e *device drivers* particulares do processador OMAP e dos periféricos do *Innovator*.

Além disso, também foram inseridas otimizações de escalonamento de processos de kernel, de forma a garantir o desempenho necessário na comunicação entre o RISC e o DSP.

VI. IMPLEMENTAÇÃO DO TERMINAL AD HOC

A implementação do terminal ad hoc é descrita a partir de suas principais funcionalidades embarcadas no processador OMAP.

A Figura 1 mostra um diagrama em blocos da plataforma hardware do protótipo do terminal ad hoc [28] e a Figura 2 mostra a pilha de protocolos de aplicação e as funções realizadas no processador OMAP e na placa WLAN 802.11.

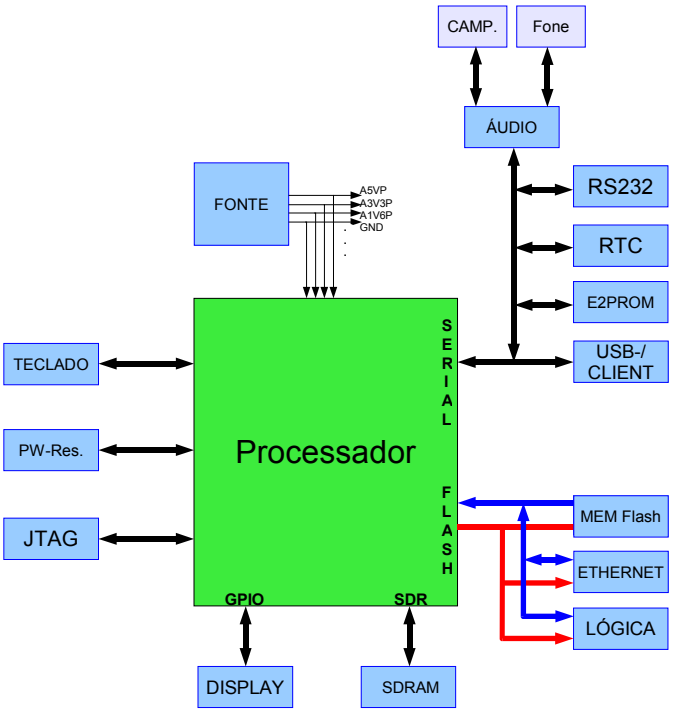

Fig.1: Terminal ad hoc.

A. *Transmissão de voz*

A implementação dessa funcionalidade pode ser dividida em duas partes: módulo de processamento do sinal de voz e módulo de telefonia e acesso à rede.

AI. Módulo de Processamento de Sinais

Esse módulo é responsável pela codificação do sinal de voz. Foram implementados os codificadores G.711- lei A e G.726.

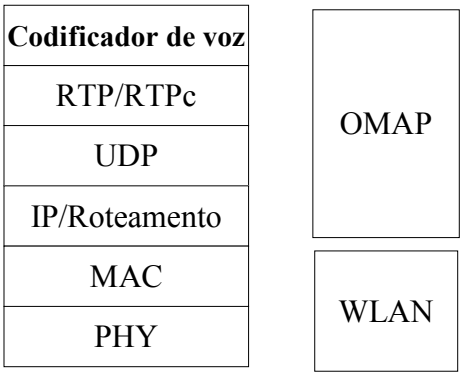

Fig.2: Pilha de protocolos do terminal ad hoc.

A implementação do codificador G.726 foi realizada utilizando-se o algoritmo padronizado pela ITU-T. A implementação em *firmware* passou por várias fases de otimização, de forma a garantir um desempenho satisfatório no DSP.

Essas otimizações, no entanto, não visaram obter o melhor desempenho possível, mas apenas o necessário para operar nas condições oferecidas.

A implementação do codificador G.711- lei A também baseou-se na especificação da ITU-T e não foi necessária nenhuma otimização extra para se conseguir um desempenho aceitável.

AII. Módulo de telefonia e acesso à rede

O módulo de telefonia e acesso à rede é responsável pela interface com o usuário, estabelecimento de sessão de voz, transmissão de pacotes em tempo real e troca de dados com o DSP.

Esse módulo é composto por três partes distintas: o agente SIP cliente, o sistema de áudio e a interface homem-máquina.

O agente SIP cliente controla as sessões do terminal. O sistema de áudio é o responsável pela comunicação com o DSP e pela transmissão e recepção de pacotes RTP.

A interface homem-máquina se constitui num mecanismo de tratamento de eventos associado a uma interface gráfica baseada nas bibliotecas Minigui, que simula a interface de um telefone.

A solução escolhida para a implementação dessa interface foi o Minigui versão 1.3.3 [29].

Para implementação do protocolo SIP, foi utilizada a biblioteca OSIP versão 2-2.2.0 da GNU [30].

Para implementação dos protocolos RTP e RTCP foi utilizada a biblioteca jRTP versão 3.1.0 [31].

B. *Protocolo de Roteamento*

Como solução de protocolo de roteamento, foi utilizado o protocolo AODV-UU versão 0.9.1 [6], sendo que não foram necessárias muitas alterações na implementação original.

O protocolo AODV-UU foi compilado para a plataforma ARM e configurado para atuar dentro da rede ad hoc sem fio.

C. *Agente SNMP*

Foi utilizado o agente Net-SNMP versão 5.2.1 [9], que implementa a MIB padrão MIB II do protocolo SNMP [10], cujos objetos representam as informações de gerência de redes baseadas no protocolo TCP/IP.

O Net-SNMP foi inicialmente compilado para contemplar a especificação do *framework* SNMPv2.

A partir dessa compilação, utilizou-se apenas o SNMPD, o qual é o agente responsável por responder a requisições de informações sobre o estado do terminal. As demais partes foram excluídas, restando apenas o aplicativo SNMPD e seus respectivos arquivos de configurações e bibliotecas associadas.

A seguir, o agente foi configurado para atuar dentro da rede ad hoc sem fio.

D. *Comunicação sem fio*

Para realizar a transmissão e recepção dos sinais, foi utilizado um adaptador 802.11b AIN *Communication* AWU2000B [32] ligado à interface USB do *Innovator*.

Esse adaptador foi configurado com os seguintes parâmetros:

- Operação em modo ad hoc
- Taxa de transmissão de 11 Mbit/s
- Potência de transmissão de 20dBm (100mW)
- Canal 6 – 2,437GHz

E. *Considerações sobre a implementação*

Após a finalização da implementação do terminal ad hoc, o mesmo foi colocado em operação em um *testbed* de rede ad hoc sem fio, realizando roteamento de pacotes, chamadas de voz através de um servidor SIP instalado na rede e sendo monitorado remotamente utilizando-se o aplicativo Getif.

As tabelas I e II mostram a quantidade de memória e o processamento consumido por alguns componentes do terminal ad hoc.

TABELA I Consumo de recursos pelo processador ARM.

	Memória Flash (kB)	Memória RAM (kB)	CPU (%)
Transmissão de voz	1258	10526	8
Roteador	430	536	0
Agente SNMP	2500	2332	0
Outros	10612	756	0
Total	14800	14160	0

TABELA II Consumo de recursos pelo DSP.

	Memória RAM (kB)	CPU (%)
G.726	16,43	43
G.711- lei A	0,97	0,42
Aplicação de controle mais SO	89,53	6,28

O item "Outros" na Tabela I refere-se aos arquivos de sistema, ao kernel em si e às demais bibliotecas presentes no sistema, inclusive a biblioteca Minigui.

Os dados de memória mostrados na Tabela II referem-se à memória consumida pela solução inteira, que inclui a aplicação de controle, os módulos do sistema operacional e o codificadores de voz. Ainda é possível otimizar a solução em termos de memória requerida.

O baixo consumo de processamento pelo Roteador explica-se pela situação sem tráfego da rede. A avaliação deverá ser repetida para a rede em plena carga.

VII. CONCLUSÕES

Neste trabalho foi apresentada a implementação de um terminal de baixo custo para redes ad hoc sem fio.

Ainda há um grande trabalho a ser realizado até a implementação da versão definitiva do terminal ad hoc.

No entanto, os resultados obtidos são encorajadores no sentido de confirmar a viabilidade do desenvolvimento e a aderência da arquitetura de hardware e *software* proposta aos requisitos do projeto.

Novos testes estão previstos para serem realizados com o terminal operando sob diferentes condições de tráfego, de forma a verificar a capacidade de operação do hardware.

O consumo de recursos de memória deve diminuir consideravelmente a partir do início dos trabalhos de otimização de *software*.

Apesar do longo caminho ainda a ser trilhado, uma etapa do desenvolvimento já foi vencida e, a partir da experiência obtida, se torna possível partir para uma próxima etapa visando a implementação de um terminal ad hoc de baixo custo.

REFERÊNCIAS

[1] C. S. R. Murthy, B. S. Manoj, "Ad Hoc Wireless Networks – Architectures and Protocols", 2004, Prentice Hall.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC-3261, June 2002.

[3] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC-3550, July 2003.

[4] ITU-T Recommendation G.711: "Pulse code modulation (PCM) of voice frequencies", ITU-T, 1988.

[5] ITU-T Recommendation G.726: "40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)", ITU-T, 1990.

[6] AODV-UU website. Em (HTTP://CORE.IT.UU.SE/ADHOC/AODVUUIMPL).

[7] Fabrício L. Figueiredo, Marcos Antônio de Siqueira, Douglas Fernandes, "Estado da arte dos protocolos de roteamento e de enlace de dados" - versão AA, Campinas, CPqD, Abril/2004 - Relatório Técnico.

[8] Projeto Net-SNMP. Em (http://www.net-snmp.org/), 2005.

[9] Em (http://www.wtcs.org/snmp4tpc/getif.htm), 2005.

[10] K. McCloghrie and M. Rose, ”Management Information Base for Network Management of TCP/IP-based internets: MIB-II”, RFC 1213, March 1991.

[11] “EANSI/IEEE Std 802.11, 1999 Edition, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification.

[12] ANSI/IEEE Std 802.11b, 1999 Edition. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band) Specifications.

[13] “The ARM Linux Project”. Em (http://www.arm.linux.org.uk), 2005.

[14] “Texas Instruments OMAP homepage”. Em (http://focus.ti.com/omap/docs), 2005.

[15] “Linux Community for Texas Instruments OMAP Processors”. Em (http://linux.omap.com) , 2005.

[16] Kipisz, S., “Building Linux for the Innovator Development Kit for OMAP Platform”, Texas Instruments, 2003.

[17] InnovatorTM Development Kit for the OMAP Platform : http://focus.ti.com/omap/docs/omapgenpage.tsp?navigationId=12341&templateId=5663&path=templatedata/cm/omaptools/data/innovator

[18] “The Linux 2.4 Kernel's Startup Procedure”. em (http://billgatliff.com/articles/emb-linux/startup.html/index.html) .

[19] “Gnu website”. Em (http://www.gnu.org) , 2005.

[20] David Dart, Shawn Dirksen, DSP/BIOS Kernel Technical Overview, Texas Instruments, 2001.

[21] DSP/BIOS Link Release Version 1.10 Release Notes, Texas Instruments, 2004.

[22] John Stevenson, Code Composer Studio IDE v2 White Paper, Texas Instruments, 2001.

[23] “Busybox website”. Em (http://www.busybox.net/), 2005.

[24] “Gcc home page”. Em (http://gcc.gnu.org) , 2005.

[25] “Gnu gdb home page” (HTTP://WWW.GNU.ORG/SOFTWARE/GDB/GDB.HTML) ,2005.

[26] “Gnu binutils home page” (HTTP://WWW.GNU.ORG/SOFTWARE/BINUTILS/), 2005.

[27] “Gnu C library home page”. Em (http://www.gnu.org/software/libc/libc.html) , 2005.

[28] João Luis Coelho, Sérgio Massami Sakai, “Descrição do ambiente de implementação de firmware” - versão AA, Campinas, CPqD, Fevereiro/2005 - Relatório Técnico.

[29] “Minigui website”. Em (http://www.minigui.org) , 2005.

[30] “Osip website”. Em(http://www.gnu.org/software/osip/), 2005.

[31] “Jrtplib website”. Em (http://research.edm.luc.ac.be/jori/jrtplib/jrtplib.html) , 2005.

[32] www.aincomm.com.tw