Implementação de um Sistema de Voz sobre IP para Plataformas Linux e μClinux

Francisco Helder C. dos Santos Filho e Luís Geraldo P. Meloni

***Resumo*— Este artigo descreve um serviço de Voz sobre IP (VoIP) baseado no Protocolo de Iniciação de Sessão (SIP). O sistema RT-DSPhone foi desenvolvido para as plataformas Linux e μClinux. O μClinux é um sistema operacional com um *kernel* compacto, desenvolvido para dispositivos embarcados com espaço reduzido. Descreve-se uma família de serviços necessários para a funcionalidade do serviço SIP, e uma visão geral do ambiente de implementação do serviço VoIP. Finalmente, apresenta-se o sistema RT-DSPhone, uma aplicação C desenvolvida para rodar tanto em máquina PC como em dispositivo embarcado. Foi utilizada a linguagem SDL para descrever e especificar as funcionalidades do sistema.**

***Palavras-Chave*— VoIP, SIP, Linux, μClinux, sistema embarcado, Protocolo, SDL.**

***Abstract*— This paper describes Voice over IP (VoIP) service, which is based on Session Initiation Protocol (SIP). The RT-DSPhone system was developed for μClinux or Linux platforms. The μClinux is an operating system with a very small *kernel*. This fact is important due to the limited footprint of embedded devices. A family of SIP servers is described being necessary for functionality of VoIP server and an overview of implementation environment of a SIP service. Finally we present the RT-DSPhone system, a C application developed to run in a PC platform as well as embedded device. For system description and especification, a SDL language was used.**

***Keywords*— VoIP, SIP, Linux, μClinux, embedded system, Protocol, SDL.**

I. Introdução

Nos últimos anos, vários estudos vem sendo realizados na área de aplicações multimídia na Internet, as quais permitem a transmissão de áudio, video e dados. Esse tema tem despertado o interesse de profissionais da área de redes de computadores e usuários em geral, principalmente o interesse pelo serviço de transmissão de voz sobre IP, ou de forma mais restrita, de telefonia IP, que assume maior importância nesse cenário [1]. Juntamente com o crescimento da Internet, verificou-se uma expansão na utilização de dispositivos embarcados ligados à redes de computadores, para explorar a convergência de serviços como áudio e video no tráfego de dados. É inevitável que o número de dispositivos embarcados irá continuar a crescer rapidamente [3]. Atualmente, é inegável a grande participação do sistema operacional Linux em vários segmentos, o qual inicialmente voltado para servidores, cada vez mais vem se popularizando como opção efetiva para *Desktops*, PDAs, dentre outras.

Atualmente existem vários protocolos para as comunicações em tempo real, entre eles o *Session Initiation Protocol* (SIP) da *Internet Engineering Task Force* (IETF) e o H.323 da *International Telecommunications Union* (ITU). O SIP é um protocolo de sinalização para iniciar, manter e terminar uma sessão de áudio, vídeo ou texto entre dois usuários finais, utilizando um protocolo de descrição de Sessão para negociar essa sessão multimídia.

Este artigo inicia-se apresentando um resumo sobre a arquitetura SIP, suas funcionalidades e serviços. Na sequência, são apresentadas as ferramentas utilizadas para a implementação do sistema VoIP em um dispositivo embarcado, tais como compilador cruzado e sistema operacional Linux, conhecido como μClinux, gerados para executar a aplicação na placa alvo. Além disso, são descritas as características e arquitetura do sistema SIP implementado. Também se provê uma comunicação entre uma placa PowerQUICC II da Motorola e um computador pessoal, usando o protocolo SIP para estabelecer essa sessão. Finalmente são apresentadas algumas conclusões.

Francisco Helder C. dos Santos Filho, Departamento de Comunicações, Faculdade de Engenharia Eletrica e de Computação, Universidade Estadual de Campinas, Campinas, Brasil, E-mails: heldercs@decom.fee.unicamp.br.

II. Revisão do Protocolo SIP

O SIP é um protocolo de sinalização usado para iniciar, manter e terminar uma sessão de áudio, vídeo ou texto entre dois usuários finais, utilizando o *Session Description Protocol* (SDP) para negociar essa sessão multimídia. A sessão SIP envolve dois ou mais participantes, podendo usar comunicação *unicast* ou *multicast*. O SIP foi desenvolvido por um grupo de trabalho da IETF. O protocolo SIP foi publicado na RFC 3261 [5], sendo baseado no *Simple Mail Transfer Protocol* (SMTP) RFC 821, usado como protocolo base do serviço de e-mail e também no *Hyper Text Transfer Protocol* (HTTP) RFC 2616, protocolo base da Web, sendo um protocolo de texto baseado no modelo cliente/servidor.

A. Usando o SIP para sinalização

O SIP é um protocolo de controle atuando na camada de aplicação, que trata sessões multimídia (conferências) como chamadas telefônicas pela Internet. O SIP é um protocolo estruturado em camadas, o que significa que seu comportamento é descrito em termos de um conjunto de estágios independentes, provendo suas funcionalidades.

Uma rede SIP é composta de quatro tipos de entidades lógicas. Cada entidade tem funções específicas e pode participar de uma comunicação SIP como um cliente (iniciando pedido), ou como um servidor (respondendo ao pedido). A seguir definimos as quatro entidades lógicas SIP:

- ***User Agent*:** É um componente dividido em duas partes, o *User Agent Client* (UAC) e o *User Agent Server* (UAS). O UAC é uma entidade lógica que gera pedidos SIP e

recebe resposta para esses pedidos, enquanto o UAS é a entidade lógica que envia respostas para pedidos SIP.

- ***Proxy Server*:** Um *Proxy* é uma entidade que atua tanto como servidor, como cliente para o propósito de intermediar os dois usuários finais. O proxy pode passar adiante um pedido sem nenhuma mudança para seu destino final ou pode alterar alguns parâmetros antes de passar o pedido.
- ***Redirect Server*:** Um servidor de redirecionamento aceita o pedido SIP, mapeia o endereço SIP da chamada ou o mais novo endereço e retorna o mesmo para o cliente. Diferentemente do *proxy*, o servidor de redirecionamento não passa o pedido para outros servidores.
- ***Registrar*:** Um registrador é um servidor que aceita pedidos REGISTER. Os registradores são necessários para se manterem informados da localização atual de um usuário. O endereço IP de um usuário pode mudar sob várias circunstâncias, tais como conexão por meio de um provedor de Internet que fornece endereços dinâmicos ou um usuário móvel. Para ser capaz de alcançar esse usuário a partir de seu endereço SIP, uma entidade na rede SIP precisa manter o mapeamento entre endereços SIP e endereços IP.

A Figura 1 mostra uma chamada entre dois usuários sem a participação da entidade *proxy*, pois o endereço do destinatário é conhecido. O usuário `sip:userA@143.106.50.220`, executando o sistema RT-DSPhone no computador pessoal rodando o Linux, realiza uma chamada para o usuário `sip:userB@143.106.50.218`, executando o sistema RT-DSPhone no dispositivo embarcado PowerQUICC II rodando o μClinux.

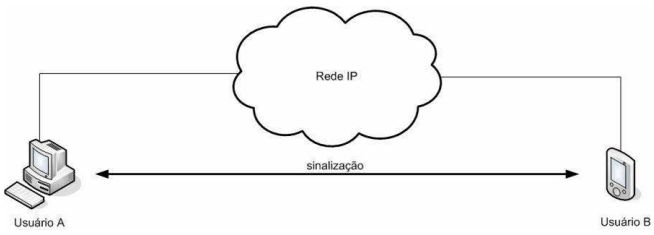

Fig. 1. Cenario de comunicação entre dois usuarios.

Para iniciar uma sessão SIP (de áudio ou texto), como o usuário A conhece o endereço do usuário B, o `userA` gera um pedido INVITE que é enviado diretamente para o destinatário do pedido para o estabelecimento da sessão. Esse pedido é entregue para um UAS, que pode potencialmente aceitar o pedido de INVITE. O UAS deverá enviar uma mensagem para o originador do pedido, que é representada pelo envio de uma resposta 2xx, informando que o pedido de INVITE foi aceito. Se o pedido de INVITE não for aceito, uma resposta 3xx, 4xx, 5xx ou 6xx é enviada, dependendo do motivo pelo qual foi rejeitado. Antes de enviar uma resposta final, o UAS pode enviar uma resposta provisória (1xx) para o originador do pedido, informando que a chamada está em progresso.

Depois de receber possivelmente uma ou mais respostas provisórias, o UAC irá receber uma ou mais respostas finais 2xx ou diferente de 2xx. Uma vez recebida a resposta final, o UAC deve enviar um ACK para toda resposta recebida. O procedimento para enviar um ACK depende do tipo de resposta. Para uma resposta final de 300 a 699, o processamento do ACK é feito na camada de transação e seguido de um conjunto de regras [5]. Para as repostas 2xx, o ACK é gerado pelo núcleo do UAC.

Uma resposta 2xx para um INVITE estabelece uma sessão, que cria um diálogo entre o UA criador do pedido de INVITE e o UA que gera a resposta. Portanto, quando múltiplas respostas 2xx são recebidas de diferentes UAs remotos, cada resposta 2xx estabelece um diálogo diferente. Todos esses diálogos fazem parte de uma mesma chamada.

A primeira etapa consiste em abrir uma conexão de sinalização entre os pontos de origem e destino da sessão. O originador da chamada SIP pode usar sinalização UDP ou TCP - a sintaxe das mensagens é independente do protocolo de transporte usado. Quando se usa o TCP, a mesma conexão pode ser usada para todos os pedidos e respostas SIP (não para os dados de mídia), ou uma nova conexão TCP pode ser usada para cada transação. Se o UDP for usado, o endereço e a porta a serem usados para as respostas aos pedidos SIP estarão contidos no campo *Via* do cabeçalho do pedido SIP. Se nenhuma porta for especificada no endereço, a conexão é feita com a porta 5060 (padrão definido pela RFC) tanto para o TCP como para o UDP. Para encerrar a sessão um dos usuários pode enviar um BYE para o outro usuário.

III. Qualidade de Serviço em Voz sobre IP

A qualidade de serviço (QoS - *Quality of Service*) refere-se à capacidade de uma rede de computadores em prover melhor serviço para um tráfego de rede sobre várias tecnologias. Intuitivamente, qualidade de serviço expressa, em ultima análise, o grau de satisfação do usuário em relação aos serviços buscados na rede.

O protocolo IP, como outras tecnologias de redes de pacotes, foi construído e aperfeiçoado para transportar dados, mas não voz ou vídeo. Anteriormente, o requisito considerado para a verificação da qualidade de serviço era a garantia de integridade dos dados, ou seja, os dados não deveriam ser danificados ou perdidos. Atualmente, com o desenvolvimento da tecnologia de redes, tornou-se possível o transporte de dados em tempo real sobre uma rede IP. Para esse novo tipo de tráfego, no entanto, é fundamental a caracterização e o controle das flutuações de tráfego na rede, conhecidas como *jitter*. O controle dessas flutuações é especialmente importante em aplicações em tempo real que precisam manter um *buffer* de pior caso para garantir a entrega oportuna dos pacotes.

A. Fatores que Influenciam na Qualidade de Serviço

Vários são os fatores que influenciam na qualidade de transmissão de voz em uma rede IP, que são:

- Atraso;
- Perda de Pacotes;
- Banda;
- Eco.

B. Técnicas de Qualidade de Serviços

A largura de banda é o limitador da capacidade de transmissão de dados na rede. Uma largura de banda inadequada pode causar desde atrasos de pacotes até a perda dos mesmos. Como o tráfego na rede é irregular e a rede é não-orientada à conexão, é necessário definir características para reservar recursos com a finalidade de conferir tratamento adequado para pacotes que fluem através da mesma.

Existem várias técnicas de priorização de pacotes que podem ser usadas. Entre elas estão o COS, TOS, DiffServ e IntServ.

C. Mecanismos para prover Qualidade de Serviço

Dois dos principais problemas em se transportar voz sobre IP são o eco e o atraso, pois a rede IP foi desenvolvida sem o intuito de trafegar voz e vídeo. Ao transportar voz sobre IP, torna-se muito difícil controlar o eco e o atraso, sendo necessária tecnologia de ponta para otimização de todos os componentes, de forma a tornar o serviço aceitável para todos os usuários. Exemplos de mecanismos utilizados para prover Qualidade de Serviço são:

- Canceladores de eco;
- Reconstrução de pacotes;
- Controle de *buffer* de *jitter* sobre o atraso.

IV. O Sistema Operacional μClinux

O μClinux ou micro-controle Linux é um sistema operacional baseado no sistema Linux, que foi desenvolvido para microprocessadores embutidos. A limitação de memória é o principal fator para o baixo custo dos processadores, onde o preço dos componentes é crucial. O μClinux é uma solução gratuita, pois pertence à comunidade de software livre.

A. Arquitetura do μClinux

O μClinux é a forma mais popular do Linux embutido. Atualmente o *kernel* do μClinux suporta uma variedade de plataformas, incluindo entre elas, Axis ETRAX, ARM, ATARI, 68k e o PowerQUICC II utilizado neste trabalho. A única diferença entre o Linux e o μClinux é o suporte para processador sem MMU (*Memory Management Unit*) e o fato dele ser projetado para dispositivos com restrições de memória. Como o Linux, o μClinux também possui suporte para conectividade, incluindo uma completa pilha TCP/IP. Além disso, também provê suporte para diferentes sistemas de arquivo, tais como EXT2, SAMBA, NFS (*Network File Systems*) entre outros. Tem-se tentado desenvolver um μClinux o mais compatível possível com o Linux padrão, fazendo com que as aplicações desenvolvidas dentro do ambiente Linux padrão, possam ser suportadas em ambiente μClinux, geralmente com poucas mudanças, limitadas somente pelo hardware do sistema embarcado e a capacidade de certos periféricos.

Quando o *kernel* é compilado com as opções básicas, tais como suporte para o processador, interpretador de comandos (console) e ferramentas de rede, o seu tamanho é da ordem de 900 KB. O arquivo imagem, composto pelo *kernel* do μClinux, sistema de arquivo e aplicação VoIP, é gerado num formato especial de compactação, tendo o tamanho variando de 500 até 900 KB. Com esse tamanho, é necessário um espaço em memória de 2 MB, para executar o μClinux.

B. Configurando o Ambiente μClinux

Compilar um *kernel* do μClinux é semelhante a compilar o *kernel* de um Linux padrão. A seguir são demonstrados os passos realizados na compilação do μClinux para plataforma PowerQUICC II da família MPC8260 da Motorola [4]. Para o processo de geração e compilação da imagem foram utilizadas as ferramentas de desenvolvimento conhecidas como *toolchain*.

Para a configuração básica do μClinux, utiliza-se o comando "make xconfig" que abre uma janela principal com um menu base de opções, como mostrado na Figura 2.

Fig. 2. Janela de configuração basica.

Clicando na opção `Target Platform Selection` na janela da Figura 2, uma segunda janela abre com opções de plataforma, como mostrado na Figura 3. Na opção `Vendor/Product` a placa MPC8265 da Motorola é selecionada. Na opção `kernel Version` é escolhida a versão 2.4 do linux e em `Libc Version` é escolhida a biblioteca `uClibc`. Por fim, é selecionada a opção `Customize Vendor/User Settings` para poder incluir a aplicação de VoIP desenvolvida neste trabalho.

Fig. 3. Janela de configuração da plataforma alvo.

Para desfazer todas as configurações no sistema, é usado o comando "make clean". Este comando irá remover os diretórios espelhos e imagens. A opção mais completa é a "make distclean", que remove todas as configurações criadas, junto com as configurações básicas de plataforma e biblioteca. Depois deste comando o sistema retorna para o estado inicial e todas as configurações deverão ser feitas novamente.

1) Carregando o μClinux no Dispositivo: O armazenamento da imagem gerada é feito diretamente na memória. Um caso mais simples de execução do *kernel* é quando a posição inicial do *kernel* é carregada em um endereço da memória que representa o endereço inicial para o processador.

A opção mais segura e flexível consiste em iniciar a placa com um pequeno código residente chamado de *bootloader*, que serve para iniciar o *offset* da memória. O *bootloader* é responsável pelas configurações iniciais da placa, tal como a iniciação básica do hardware, e permite carregar o arquivo imagem para a placa. O *bootloader* utilizado neste trabalho foi o U-Boot (*Universal Boot Loader*), que é usado na plataforma PowerQUICC II. O U-Boot inicia as configurações básicas do processador, tais como reconhecimento dos registradores, mapeamento dos endereços de memória,s etc. Depois de iniciar a placa, a imagem pode ser carregada através da interface serial ou Ethernet, utilizando o protocolo TFTP (*Trivial File Transfer Protocol*).

No μClinux, o sistema de arquivos e o *kernel* podem ser compilados em partes separadas. Na distribuição padrão do Linux, o *kernel* é embutido como parte do sistema de arquivos. Esta configuração requer um sofisticado *bootloader*, no qual faz um tratamento especial de cada parte na memória. O μClinux permite a partição do sistema de arquivo e *kernel*, sendo posicionados na memória em diferentes caminhos.

O arquivo imagem que é carregado para memória está comprimido. A primeira fase é descomprimir o *kernel* e o sistema de arquivo, e em seguida é iniciado o sistema. Isto requer que o *bootloader* usado seja capaz de tratar descompressão de arquivos. O tamanho das imagens geradas que podem ser carregadas na Flash ou RAM e da aplicação RT-DSPhone são apresentadas na Tabela I.

TABELA I
DESCRIÇÃO DO TAMANHO DAS IMAGENS GERADAS

Arquivo	Tamanho (MB)
image.bin	2
image.elf*	6.2
linux.bin*	0.86
RT-DSPhone	0.02

O *kernel* do Linux já provê suporte para software em tempo real, com características *Preemptive*, sendo que o *kernel* do μClinux também tem algum enfoque para o suporte em tempo real. Existem dois tipos de enfoque para o suporte em tempo real no μClinux: o *Real-Time Linux* (RTLinux) e o *Real-Time Application Interface* (RTAI).

V. SISTEMA RT-DSPHONE IMPLEMENTADO

O sistema foi desenvolvido na linguagem ANSI C padrão, utilizando-se as ferramentas de compilação gcc da GNU e o compilador cruzado powerpc-gcc desenvolvido para o processador PowerQUUIC-II da Motorola. Esse processador é amplamente utilizado nos sistemas de comunicações atuais. A escolha do ambiente Linux deve-se a grande robustez desse sistema operacional e a sua crescente utilização no mundo das comunicações e sistemas embarcados. Foi também utilizada a pilha oSIP [2], que é uma ferramenta desenvolvida pela GNU para prover suporte no desenvolvimento da aplicação VoIP.

A. Implementação da Camada de Transação

O protocolo SIP é dividido em camadas. A camada mais baixa é a de sintaxe e codificação das mensagens. A segunda camada é a camada de transporte, que define como um cliente envia pedidos e recebe respostas, e como um servidor recebe pedidos e envia respostas. A terceira camada é a camada de transação, responsável por enviar pedidos pelo *Client Transaction* usando a camada de transporte, para o *Server Transaction*. A camada de transação trata as retransmissões, associando as respostas aos pedidos e disparo de *timeouts*. A camada superior à camada de transação é chamada *Transaction User* (TU).

Toda transação tem um lado cliente e uma lado servidor, também conhecido como transação cliente e transação servidor. Uma transação cliente é responsável por receber pedidos da TU e entregar esses pedidos para uma transação servidor, usando a camada de transporte. A transação servidor recebe e repassa as repostas para a TU. O objetivo de uma transação servidor é receber pedidos da camada de transporte e repassar para a TU, assim como receber uma reposta da TU e passar para a camada de transporte para ser transmitido pela rede.

A principal característica implementada pela pilha oSIP é representada pelas quatro máquinas de estados que são aplicadas para diferentes transações definidas pela RFC 3261[5]. O SIP define quatro máquinas de estados, como mostrado abaixo:

- ICT : Invite Client Transaction;
- NICT: Non Invite Client Transaction;
- IST : Invite Server Transaction;
- NIST: Non Invite Server Transaction.

Em seguida é descrito o diagrama de estados *Invite Client Transaction* e *Invite Server Transaction*, explicando os processos de envio e recepção de um INVITE por parte do UAC e UAS, respectivamente, como mostrado nas Figuras 4 e 5.

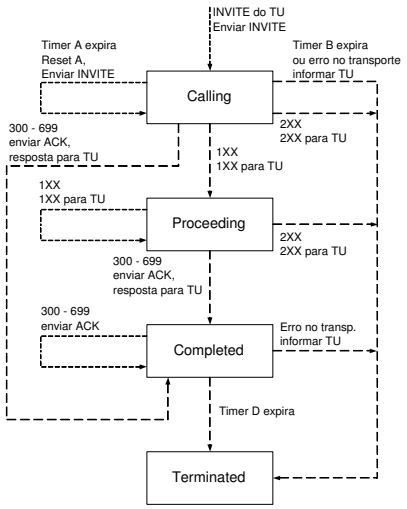

Fig. 4. INVITE Client Transaction.

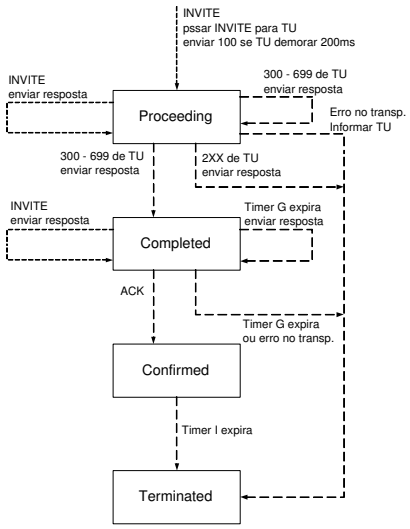

Fig. 5. INVITE Server Transaction.

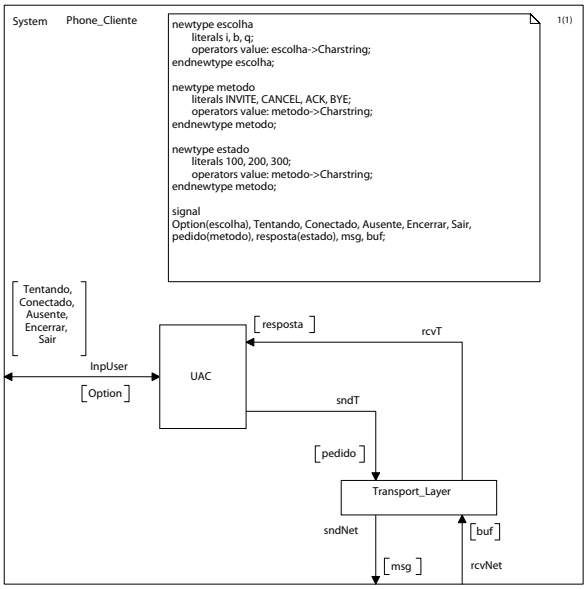

Fig. 6. Diagrama do sistema Phone Cliente.

B. Descrição do Sistema SIP Implementado

O sistema SIP implementado é composto de uma entidade Phone-Cliente responsável por enviar um pedido e uma entidade Phone-Servidor, responsável por responder esse pedido. Para a especificação e descrição do sistema RT-DSPhone desenvolvido neste trabalho, foi utilizada uma linguagem chamada SDL (*Specification and Description Language*), que é uma linguagem padrão para especificar e descrever sistemas de comunicações. Tal linguagem foi desenvolvida pela CCITT (*Comité Consultatif International Téléphonique*), hoje ITU-T, na recomendação Z.100 do ITU-T [7]. O sistema foi implementado utilizando-se a ferramenta de desenvolvimento Cinderella [6], que usa a linguagem SDL para modelar, simular e gerar protótipo de código.

1) Sistema Atuando como Cliente: O sistema **Phone Cliente** consiste de dois blocos chamados **UAC** e ***Transport Layer***, como mostrado na Figura 6. O bloco **UAC** é responsável pela comunicação entre o usuário e a camada de transporte, por onde vai ser enviada e recebida uma mensagem SIP. Os sinais **Tentando**, **Conectado**, **Ausente**, **Encerrar** e **Sair** são enviados para o usuário pelo **UAC**, através do canal **InpUser**, para mostrar o estado em que se encontra a comunicação. O sinal **Option** recebe do usuário, via canal **InpUser**, uma opção definida pelo tipo **escolha**, representado pelos caracteres (i=INVITE), (b=BYE), (q=Sair). O bloco ***Transport Layer*** recebe do **UAC** uma mensagem de pedido SIP representada pelo sinal **pedido**, via canal **sndT**, convertendo essa mensagem para o formato **string**, representada pelo sinal **msg**, e a envia pela rede via canal **sndNet**. O bloco ***Transport Layer*** também recebe uma mensagem definida pelo sinal **buf**, via canal **rcvNet**. Essa mensagem é convertida para uma mensagem de resposta SIP e passada para o bloco **UAC**. Para essas transações é usado o protocolo UDP.

São definidos tipos para as mensagens de pedido SIP, que são representadas pelos sinais **INVITE**, **CANCEL**, **ACK** e **BYE**. Para as mensagens de resposta SIP, são definidos tipos representados pelos sinais literais **100**, **200** e **300**.

2) Bloco User Agent Client: A especificação do bloco **UAC** é mostrada na Figura 7. Esta contém dois processos, o processo **TU** e o processo ***Client Transaction***, que se comunicam através da rotina **Sync**. O processo **TU** se comunica com o canal **InpUser**, via rotina **InpEsc**. O processo ***Client Transaction*** se comunica com o canal **rcvT** e **sndT**, através das rotinas **rcvCT** e **sndCT**, respectivamente.

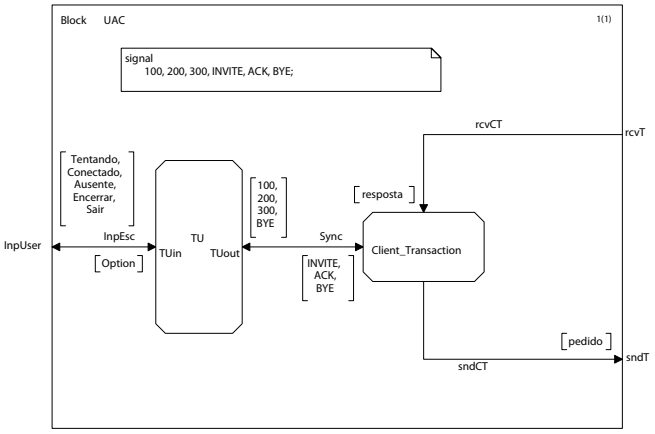

Fig. 7. Diagrama do Bloco *User Agent Client*.

3) Sistema Atuando como Servidor: O sistema **Phone Servidor** consiste de dois blocos chamados **UAS** e ***Transport Layer***, como mostrado na Figura 8. O bloco **UAS** é responsável pela comunicação entre o usuário e a camada de transporte, por onde vai ser recebida e enviada uma mensagem SIP. Os sinais **Tentando**, **Conectado**, **Ausente**, **Encerrar** e **Sair** são enviados para o usuário pelo **UAS**, através do canal **InpUser** para mostrar o estado em que se encontra a comunicação. O sinal **Option** recebe do usuário, via canal **InpUser**, uma opção definida pelo tipo **escolha**, representada pelos caracteres (b=BYE), (q=Sair) e (s=estado).

O bloco ***Transport Layer*** recebe do **UAS** uma mensagem de resposta SIP, via canal **sndT**, representada pelo sinal **resposta**, convertendo-a para uma mensagem padrão, representada pelo sinal **msg** e a enviando pela rede, via canal **sndNet**. O bloco ***Transport Layer*** também recebe uma mensagem **buf** padrão da rede, via canal **rcvNet**, convertendo-a para uma mensagem de pedido SIP e a passando para o bloco **UAS**. Para essas transações é usado o protocolo UDP.

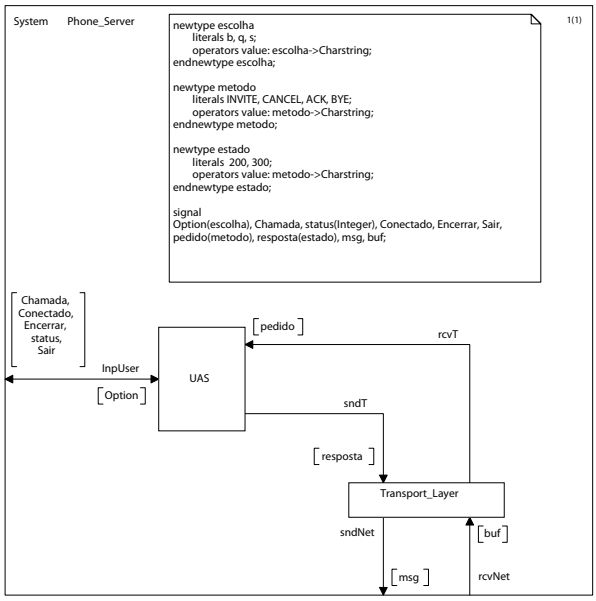

Fig. 8. Diagrama do sistema Phone Servidor.

São definidos tipos para as mensagens de reposta SIP, representadas pelos sinais literais **100**, **200** e **300**. Para as mensagens de pedido SIP, são definidos tipos representados pelos sinais **INVITE**, **CANCEL**, **ACK**, **BYE**.

4) Bloco User Agent Server: A especificação do bloco **UAS** é mostrada na Figura 9. Esta contém dois processos: o processo **TU** e o processo ***Server Transaction***. Os processos se comunicam através da rotina **Sync**, e o processo **TU** se comunica com o canal **InpUser**, via rotina **InpEsc**. O processo ***Server Transaction*** se comunica com o canal **rcvT** e **sndT**, através das rotinas **rcvST** e **sndST**, respectivamente.

VI. Conclusões

O presente trabalho tem como objetivo o estudo e desenvolvimento de um sistema que permite a comunicação entre dois participantes, que podem estar tanto em um computador como em um dispositivo embarcado. Para o estabelecimento das sessões foi utilizado o protocolo de sinalização SIP. O SIP foi também escolhido pelo 3rd *Generation Partnership Project* (3GPP), para estabelecer sessões multimídia na rede UMTS (R5), e por grandes operadoras, como a WorldCom e AOL, assumindo-se como a principal alternativa à recomendação H.323.

Foram realizados testes de modo a avaliar o nível de interoperabilidade através de critérios de avaliação definidos pelo *Technical Program Committee* e testes de interoperabilidade com outros softwares comerciais disponíveis na Internet, tais

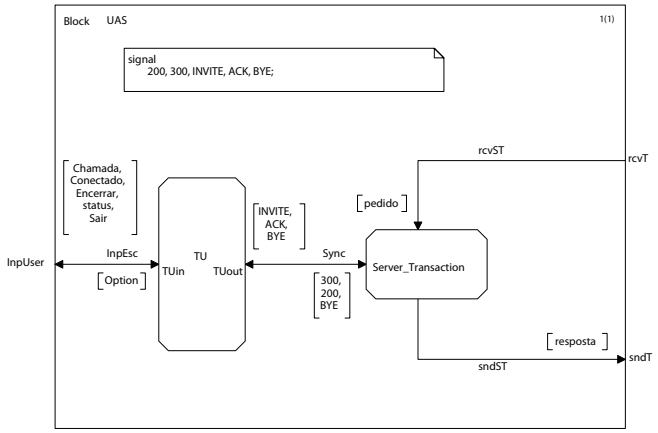

Fig. 9. Diagrama do Bloco *User Agent Server*.

como o Kphone que é um *User Agent* SIP para Linux, no qual se pode iniciar uma conexão VoIP sobre Internet. Tal *software* suporta chamadas de áudio, mensagens instantâneas, e vídeos entre dois computadores. O *User Agent* Ubiquity que atua como um "soft phone", permitindo realizar chamadas de PC-para-fone usando a Internet e é utilizado para os desenvolvedores testarem seus produtos. Por fim, o Linphone é um web phone, que foi desenvolvido por um grupo de franceses.

Embora a qualidade de serviço seja um aspecto importante, e foi por isso revista neste trabalho, convém mencionar que não foram ainda implementadas soluções no RT-DSPhone que garantam a qualidade de serviço associada à transmissão de dados sobre redes IP.

Agradecimentos

Agradeço ao aluno de doutorado Helcio Wagner da Silva pelos debates técnicos, que foram de grande valia para o desenvolvimento do trabalho e na revisão deste artigo, e ao aluno de doutorado Glauco F. Gazel Yared pela revisão deste artigo.

Referências

[1] J. Paulo P. de Sousa, sIPTel - Um Sistema de IPtel com suporte para vıdeo utilizando o protocolo SIP", Tese de Mestrado, 2003.
[2] GNU software, *libosip Reference Manual*", 2002.
[3] Michael Barr, *Programming Embedded Systems*", 1a edição, Editora O'Relly, Janeiro 1999.
[4] Freescale Semiconductor, *MPC862 PowerQUICC Family Users Manual*", Maio 2003.
[5] J. Rosemberg e H. Schulzrinne, *SIP: Session Initiation Protocol*", RFC 3261, Outubro 2002.
[6] Cinderella software, http://www.cinderella.dk", 2005.
[7] Jan Ellsberger, Dieter Hogrefe, Amardeo Sarma, *SDL - Formal Object-Oriented Language for Communication Systems*", 1a Edição, Editora Prentice Hall, Outubro 1997.