Wagner Vinicius Vieira

Mobilidade com o Protocolo SIP: Concepção de uma Plataforma de Testes e Análise de Cenários

São José – SC

dezembro / 2010

Wagner Vinicius Vieira

Mobilidade com o Protocolo SIP: Concepção de uma Plataforma de Testes e Análise de Cenários

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Eraldo Silveira e Silva, M. Eng.

Co-orientador:
Prof. Emerson Ribeiro de Mello, Dr.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

dezembro / 2010

Monografia sob o título *"Mobilidade com o Protocolo SIP: Concepção de uma Plataforma de Testes e Análise de Cenários"*, defendida por Wagner Vinicius Vieira e aprovada em 23 de dezembro de 2010, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Eraldo Silveira e Silva, M. Eng.
Orientador
IFSC - Campus São José

Prof. Ederson Torresini, M. Eng.
IFSC - Campus São José

Prof. Odilson Tadeu Valle, M. Eng.
IFSC - Campus São José

Não confunda derrotas com fracasso nem vitórias com sucesso. Na vida de um campeão sempre haverá algumas derrotas, assim como na vida de um perdedor sempre haverá vitórias. A diferença é que, enquanto os campeões crescem nas derrotas, os perdedores se acomodam nas vitórias.

Roberto Shinyashiki

Agradecimentos

Agradeço primeiramente a Deus. Por estar sempre iluminando e guiando minha vida. E por ter me dado forças para chegar ao fim desta jornada.

Aos meus pais, Itamar e Neuseli, por deixarem à mim a escolha e decisões da vida, após mil conselhos é claro. Mas acima de tudo, confiar em mim, e por me apoiar.

À Letícia, minha noiva, que com sua presença amorosa, tem compartilhado bons e maus momentos ao meu lado. Por todo carinho, e compreensão.

Aos amigos e colegas de Curso, que estiveram presentes oferecendo trocas de conhecimentos e apoio.

Aos professores presentes em minha vida acadêmica. Em especial ao meu mestre orientador Eraldo Silveira e Silva. Obrigado pela ajuda sempre atenciosa, e por indicar os caminhos quando necessário. Agradeço as exigências e cobranças, pelas conversas motivadoras e por acreditar em meu potencial.

E finalmente a todos que compartilharam direta ou indiretamente para que este trabalho fosse concluído.

Muito obrigado!

Resumo

O estudo da comunicação de Voz sobre IP (VoIP) e redes sem fio nunca foi tão visado como nos dias de hoje. O *Session Initiation Protocol* (SIP) é um dos principais protocolos de sinalização usado em sessões de comunicação VoIP. Dois objetivos foram determinados para este trabalho: a definição de uma plataforma de testes com o SIP, no âmbito de sistemas embarcados e, em segundo lugar, a análise do uso do SIP em um contexto de mobilidade. Tendo em vista que o comportamento do protocolo DHCP influencia no desempenho do SIP em situações de mobilidade, inicialmente optou-se pelo uso de uma versão do DHCP para sistemas embarcados: *micro Dynamic Host Configuration Protocol client* (uDHCPc). Foram então realizados experimentos no sentido de proceder com medições dos tempos gastos na aquisição de endereçamento IP com o uDHCPc, tendo sido implementado um esquema com *scripts* para auxiliar o uDHCPc na reconexão com uma rede ao alcance. A biblioteca escolhida para auxiliar na plataforma foi a PJSIP. Na sequência foi feita uma junção das características de uma aplicação desta biblioteca (*simple_pjsua*), com as do uDHCPc, realizando-se ao fim um cenário de mobilidade. Obteve-se a confirmação de um re-registro junto ao Asterisk, após mobilidade e descoberta de uma nova rede. Os tempos obtidos durante os experimentos foram mensurados e ilustrados em tabelas. A mobilidade de sessões em andamento foi discutida mas não implementada.

Palavras-chave: Sessão SIP, Sistema embarcado, Mobilidade, Linux Mínimo.

Abstract

The study of communication and Voice over IP (VoIP) and wireless networks has never been targeted as today. The Session Initiation Protocol (SIP) is a major signaling protocols used in VoIP communication sessions Two goals were established for this work: the definition of a test platform with SIP in the context of embedded systems and, secondly The analysis of the use of SIP in the context of mobility. Given that the behavior of the DHCP protocol influences the performance of SIP in mobile situations, initially it was decided to use one version of DHCP for embedded systems: micro Dynamic Host Configuration Protocol (uDHCPc). We then performed experiments in order to proceed with measurements of time spent in acquiring IP address with the uDHCPc and was implemented with a scheme to assist the uDHCPc scripts in reconnecting with a network in range. The library chosen to assist in the platform was PJSIP. Following was a combination of characteristics of an application of this library (simple_pjsua) with the uDHCPc, taking place after a mobility scenario. We obtained confirmation of a re-registration with Asterisk, mobility and after discovery of a new network. The times obtained during the experiments were measured and illustrated in tables. The mobility of sessions in progress was discussed but not implemented.

Keywords: SIP Session, Embedded System, Mobility, Linux Minimum.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Contexto e Motivação do Trabalho

A migração de serviços tradicionais das redes telefônicas para a Internet vem se acentuando a cada ano. A comunicação em tempo real de voz e imagem pode ser implementada sobre a arquitetura *Transmission Control Protocol/Internet Protocol* (TCP/IP) através da utilização de protocolos específicos para a sinalização e transporte da mídia.

Alguns problemas advém do uso de uma rede IP, baseada em comutação de pacotes, quando utilizada como base para serviços de telefonia. Um deles é a questão da qualidade de serviço, que ainda não é comparável aos circuitos dedicados proporcionados pela telefonia tradicional. Um problema adicional é o tratamento adequado da mobilidade de terminais. Na telefonia baseada em redes celulares, os procedimentos de troca de estação base com comunicações em andamento são realizados em intervalos relativamente pequenos, imperceptíveis ao usuário final. Em redes IP, a mudança de pontos de acesso de uma rede sem fio pode implicar na mudança de prefixo IP (uma nova sub-rede). Neste caso, existe a formação de um novo endereço IP e este endereço deverá ser, de alguma forma, levado em conta na comunicação. Mesmo quando o terminal não possui comunicações em andamento faz-se necessário de alguma forma publicar o novo endereço onde está localizado.

Existem diferentes soluções para o tratamento deste problema dentro da tecnologia IP. Na camada de rede, o IP Móvel (PERKINS, 1996) e seus derivados se apresentam como soluções transparentes às aplicações, embora sejam apontados problemas tais como a triângularização da comunicação (SCHULZRINNE; WEDLUND, 2000). Na camada de aplicação pode ser utilizado o protocolo de sinalização SIP (ROSENBERG et al., 2002). Este protocolo é usado para estabelecer, modificar e terminar sessões multimídia e vem se tornando um padrão de fato para a sinalização de serviços de telefonia em redes IP (RIBEIRO, 2008). Ele fornece elementos para criar infra-estruturas de servidores de rede (servidores *proxy*, servidores de registros) de forma a apoiar a localização de usuários e agregar serviços de apoio à comunicação.

O *Session Initiation Protocol* (SIP) proporciona mecanismos para tratar diferentes manifestações da mobilidade (SCHULZRINNE; WEDLUND, 2000), tais como a mobilidade de sessão, de terminal, pessoal, e de serviço. Entretanto, o desempenho deste protocolo no sentido de restabelecer a comunicação, no caso da mobilidade de terminal, depende também de outros fatores. Um deles é a capacidade do terminal de estabelecer o mais rápido possível um novo endereço IP e, neste ponto, surge o protocolo DHCP, responsável pela aquisição dinâmica de endereços IP na rede.

Normalmente os terminais que se movimentam são enquadrados na categoria de sistemas embarcados que possuem limitações em termos de suporte de execução (ex. sistemas operacionais) e de hardware. As implementações do DHCP e do SIP devem ser ajustadas às condições destes terminais. Estas implementações, na sua maior parte, não estão devidamente preparadas para o tratamento da mobilidade de terminais entre sub-redes. É dentro deste contexto que se enquadra este trabalho.

A motivação inicial para o estudo deste tema foi o interesse por parte do autor na área de redes sem fio e mobilidade, despertado a partir do trabalho intitulado de Plataforma para o Estudo de Mobilidade na Camada de Rede (SILVA; MEDEIROS, 2009). Aliou-se a isto a oportunidade de auxiliar na agregação de conhecimentos para um futuro projeto na área de comunicações móveis em redes IP, almejado pelo grupo de pesquisa de Telecomunicações do IFSC campus São José.

1.2 Objetivos

O objetivo geral deste trabalho é investigar o comportamento conjunto do protocolo SIP com o protocolo DHCP frente à mobilidade entre sub-redes IP no contexto de sistemas embarcados.

Como objetivos específicos pode-se citar:

- A definição de implementações do SIP e *Dynamic Host Configuration Protocol* (DHCP) a serem utilizadas em um sistema embarcado;
- A realização de testes para verificar o desempenho da implementação do DHCP frente à mobilidade entre sub-redes;
- A proposição de melhorias da resposta do DHCP frente à mudança de sub-redes;

- A realização de testes do comportamento da implementação do SIP frente à mobilidade e na sua relação com o protocolo DHCP;
- A proposição de melhorias da resposta do SIP frente à mudança de sub-redes, restringindo-se neste ponto a um novo registro na nova rede.

1.3 Organização do texto

O texto está organizado da seguinte forma: O capítulo 2 é dedicado à fundamentação teórica do trabalho, sendo descritos os conceitos de mobilidade, o protocolo SIP e a mobilidade, o protocolo DHCP, e por último, alguns conceitos associados a sistemas embarcados. No capítulo 3 é apresentado o estudo sobre a capacidade de resposta do uDHCPc - a implementação escolhida do DHCP - frente a mobilidade. Um proposta de melhoria da resposta deste protocolo é então descrita, usando a ferramenta *ip monitor* e um *script* de alerta. No capítulo 4 é inicialmente apresentada a implementação SIP escolhida: a biblioteca PJSIP. Como será visto ela não tem capacidade de tratar a mobilidade e, por consequência, são apresentadas soluções para implementação destes mecanismos. Uma avaliação de desempenho simples é descrita. Por fim, no capítulo 5 são apresentadas as conclusões do estudo em questão e as perspectivas futuras.

2 *Fundamentação Teórica*

Neste capítulo são apresentados os fundamentos teóricos associados a este trabalho. Inicialmente são apresentados conceitos básicos associados à mobilidade, sendo caracterizado, em particular, o problema da mobilidade de terminais entre sub-redes.

Na sequência é apresentado o protocolo SIP, sendo discutidos os mecanismos que permitem a este protocolo tratar a mobilidade entre sub-redes em nível de aplicação. Um dos problemas associados à mobilidade entre sub-redes é o do tempo de aquisição de um novo endereço IP. Neste ponto, o protocolo DHCP, amplamente usado na aquisição dinâmica de endereços, impacta diretamente o comportamento do protocolo SIP frente à mobilidade. Por ser também explorado neste trabalho, o DHCP é brevemente revisto.

Finalmente, como pretende-se definir uma arquitetura para tratamento de mobilidade em um sistema embarcado de baixo custo, na seção subsequente são apresentados brevemente conceitos associados a estes sistemas, sendo discutidas as limitações que os mesmos impõem às aplicações.

2.1 Conceitos em Mobilidade

2.1.1 Tipos de Mobilidade

Segundo (SCHULZRINNE; WEDLUND, 2000) pode-se considerar quatro tipos de mobilidade: de terminal, de sessão, pessoal e de serviço.

A mobilidade de terminal tem a característica de permitir que um dispositivo em movimento mude de pontos de acesso e mesmo assim continue mantendo sessões que estão em andamento. O dispositivo deve também continuar a ser acessível, no sentido de receber novas requisições por partes de outros terminais.

A mobilidade de sessão prevê que o usuário mantenha uma sessão de mídia mesmo quando muda de terminais, ou seja, ele pode transferir uma sessão iniciada em um dispositivo móvel

para outro dispositivo, por exemplo, seu *desktop*.

A mobilidade pessoal permite que um usuário seja contactado através de um único endereço mesmo quando localizado em um dentre vários possíveis terminais. O contrário também é válido, um usuário pode ser alcançado por vários endereços, em diferentes terminais.

A mobilidade de serviço permite ao usuário manter o acesso a seus serviços mesmo quando está em movimento, seja pela troca de dispositivo, seja pelo acesso a uma nova de rede. Ou seja, suas configurações pessoais, agenda de endereços, registro de chamadas, lista de discagem e outros, estarão disponíveis em qualquer local e dispositivo.

É interessante observar que a mobilidade de terminal pode produzir um *handover* (mudança de ponto de acesso) horizontal ou vertical. O *handover* horizontal ocorre entre os pontos de acesso que se utilizam da mesma tecnologia. Por exemplo, na Figura 2.1, um *smartphone* está inicialmente conectado à "rede1" disponibilizada pelo ponto de acesso AP1. Ao se movimentar, ele pode entrar no alcance da "rede2" de um outro ponto de acesso, o AP2, que se utiliza da mesma tecnologia. Ao perder conexão com a primeira rede, o *smartphone* fará associação com o AP2, usando a mesma interface de acesso.

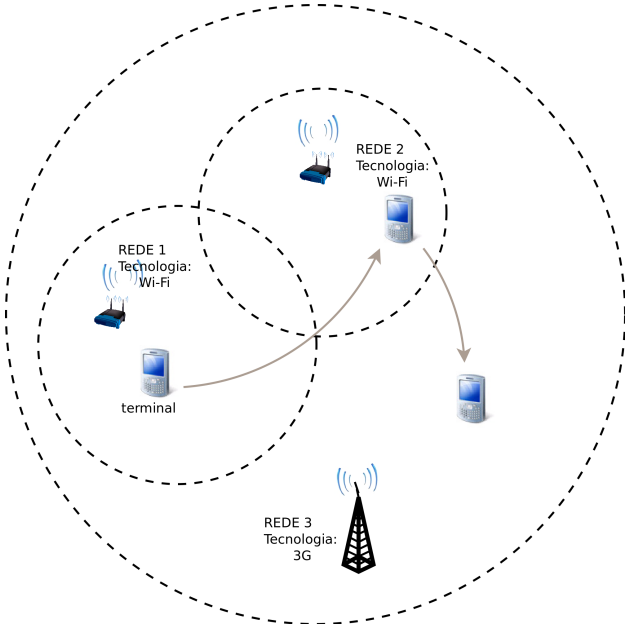

Figura 2.1: Handover Horizontal e Vertical.

O *handover* vertical envolve a manutenção de sessões de comunicação quando existe uma mudança de uso de uma interface para outra (possivelmente de tecnologias diferentes). Por exemplo, um *smartphone* dotado de interfaces para redes sem fio *Wireless Fidelity* (Wi-Fi) e para a tecnologia 3G, pode inicialmente manter uma sessão Wi-Fi e quando entrar na área de cobertura de uma rede 3G pode decidir migrar a sessão para esta rede, conforme a Figura 2.1.

2.1.2 O Problema da Mobilidade entre Sub-redes

Nas redes IP, a comunicação ocorre através da troca de pacotes, que são enviados ao destino final com base no endereço IP (salvo sistemas de roteamento especiais envolvendo a fonte). Sabe-se que é o prefixo de rede associado a este endereço(KUROSE; ROSS, 2003) que é publicado pelo sistema de roteamento da Internet. Portanto, quando um terminal muda de rede, ele deverá formar um novo número *Internet Protocol* (IP) associado ao novo prefixo.

Considere o cenário mostrado na Figura 2.2. Os pontos de acesso AP1 e AP2 funcionam como pontes e estão conectados a mesma sub-rede 10.0.0.0/24. O terminal móvel TM pode se movimentar entre estes pontos de acesso sem necessitar mudar de endereço IP. Se o terminal correspondente CN envia um fluxo destinado ao endereço de TM então, esteja ele associado a AP1 ou AP2, receberá corretamente os pacotes, uma vez que protocolos, tais como o ARP, se encarregarão de resolver possíveis inconsistências no mapeamento de endereços da camada de enlace na camada de rede.

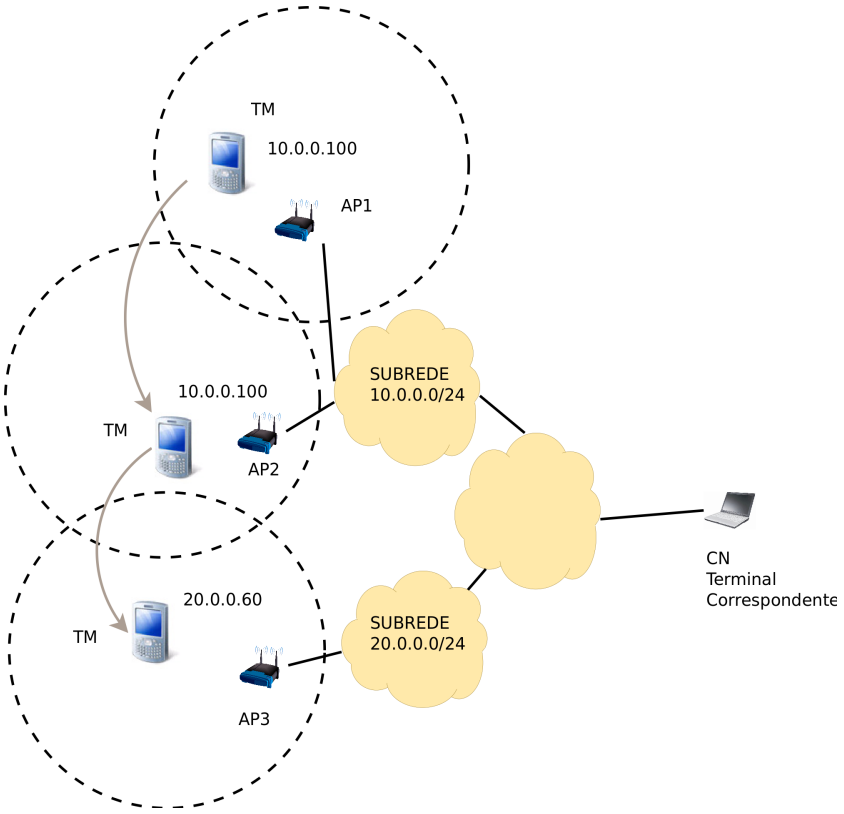

Figura 2.2: O Problema de Mobilidade entre Sub-redes.

Quando o TM se associa com o AP3 posicionado na sub-rede 20.0.0.0/24, este deverá formar um novo endereço nesta rede. O TM não pode continuar a usar o endereço antigo pois os pacotes destinados a este endereço serão encaminhados para a rede de origem. Tecnologicamente é possível atualizar o sistema de roteamento da Internet para um único endereço de

hospedeiro. Entretanto, este procedimento é inviável em termos de escalabilidade.

A mudança de endereço IP no terminal impacta diretamente na camada de transporte. Uma conexão TCP é rompida e mesmo fluxos utilizando o UDP terão que ser reajustados para o novo endereço.

É interessante observar que uma mudança de sub-rede possui ainda outros fatores que afetam o desempenho do sistema e que estão interrelacionadas com as camadas inferiores. Pode-se citar (DUTTA et al., 2006) questões tais como: detecção de uma nova rede de acesso, a decisão de realizar ou não o *handover*, o tempo de descoberta de servidores DHCP e/ou formação de novo endereço IP.

2.1.3 Soluções para a Mobilidade de Terminais entre Sub-redes

Existem soluções para o problema da mudança de sub-rede para as diversas camadas da arquitetura TCP/IP, por exemplo: o IP Móvel na camada de rede, o *Mobile Stream Control Transmission Protocol* (MSCTP) na camada de transporte e o SIP na camada de aplicação.

O protocolo IP Móvel assume que um terminal pode conviver com dois endereços em nível de rede: o endereço domiciliar (HoA) que identifica o terminal e o *Care-of-Address* (CoA) que o localiza. Qualquer fluxo provindo de um terminal comunicante será destinado ao *Home Address* (HoA), que, em princípio é o endereço público do terminal móvel. Quando o terminal visita uma rede, ele se registra com um agente na rede domiciliar (*home agent*). Este agente recebe os pacotes destinados ao HoA e os encaminha via túnel para o terminal, no endereço CoA. Esta abordagem faz com que a mobilidade seja transparente para a camada de transporte que receberá pacotes intactos destinados ao HoA. O *Mobile Internet Protocol version 6* (MIPv6) (PERKINS, 1996) possibilita que os registros sejam feitos diretamente com o terminal correspondente evitando a triangulação na comunicação.

O *Stream Control Transmission Protocol* (SCTP) (STEWART et al., 2000) pode ser visto como um meio termo entre os protocolos TCP e o UDP. Ele permite manter sessões entre vários pontos comunicantes. O MSCTP (RIEGEL, 2002) é uma variação do SCTP que proporciona a mobilidade terminal em nível de transporte. Ele implementa mecanismos que permitem a configuração dinâmica de endereços IP (QUINTERO, 2007), ou seja, é possível inserir novos pontos comunicantes dinamicamente. Desta forma, se um terminal móvel em comunicação, recebe um novo endereço, ele pode inserí-lo de forma dinâmica no grupo comunicante.

O SIP oferece mobilidade terminal, pessoal, de sessão e de serviço (SCHULZRINNE; WEDLUND, 2000). Ele proporciona facilidades como a localização de usuários, transferência

e conferência de chamadas. Como ponto negativo pode-se citar o fato de que não é uma solução transparente para as aplicações. Na sequência é feita uma descrição mais detalhada do protocolo SIP por ser objeto de estudo deste trabalho.

2.2 O Protocolo SIP

2.2.1 Visão Geral do Protocolo

O SIP é um protocolo de sinalização, situado na camada de aplicação. Foi definido pela *Internet Engineering Task Force* (IETF) na *Request for Comments* (RFC) 2543 em março de 1999 e revisado em junho de 2002, sendo definido desta vez na RFC 3261 (ROSENBERG et al., 2002).

É um protocolo baseado em texto, que herda características do *HyperText Transfer Protocol* (HTTP) e do *Simple Mail Transfer Protocol* (SMTP). Ele tem por finalidade criar e gerenciar sessões entre aplicações para troca de fluxos de informações, iniciando, modificando e encerrando tais sessões. Trabalha em conjunto com outros protocolos, tais como o *Real-Time Transport Control Protocol* (RTCP), *Real Time Streaming Protocol* (RTSP), *Media Gateway Control Protocol* (MEGACO), *Session Description Protocol* (SDP), porém independe de qualquer um destes protocolos para exercer seu funcionamento básico de sinalização, além de não depender do tipo de sessão que está sendo estabelecida.

O SIP dispõe de um serviço de mapeamento de nomes e redirecionamento que permite que o usuário seja encontrado através de um identificador global independendo da sua localização na rede. Este idenficador é o SIP URI que se apresenta da forma: *sip:usuario@dominio.com*.

Transações SIP

O protocolo SIP possui um modelo de transações requisição/resposta. Uma requisição originada por um cliente evoca um determinado método (função) no servidor. A linha de requisição de uma mensagem SIP é formada por um método, um endereço e a identificação da versão SIP utilizada, sendo característica desse tipo de mensagem a utilização de uma linha de requisição no início. A versão atual do SIP possui seis métodos básicos:

- ***INVITE*** **-** É o envio de convite para participar de uma sessão. Na mensagem *INVITE* pode haver uma descrição da sessão. Esta descrição é realizada por um outro protocolo, o SDP. Estas informações podem conter, por exemplo, padrões de codecs suportados

pelo agente e porta a ser utilizada pelo *Real-Time Transport Protocol* (RTP) para troca de mídia.

- ***ACK -*** é a confirmação do recebimento da resposta final para uma requisição *INVITE*. Pode conter uma mensagem SDP de descrição da sessão negociada entre os clientes.
- ***BYE -*** Solicita o término da sessão, liberando os recursos associados à conexão;
- ***CANCEL -*** Solicita que seja cancelada uma requisição ainda pendente;
- ***REGISTER -*** é o método responsável pelo registro do endereço do agente do usuário com um servidor de localização. Este método é o centro das atenções deste trabalho pois a mobilidade de terminal deve implicar em um novo registro de localização;
- ***OPTIONS -*** Informa a capacidade e disponibilidade dos servidores SIP.

Em resposta às requisições SIP existem seis classes de respostas. Uma resposta conclui uma transação. Essas respostas são representadas por três digitos, onde o primeiro digito é o código de status e os outros dois servem para distinguir as mensagens.

Funcionalmente, a geração de uma requisição é realizada por um agente cliente, enquanto o agente servidor recebe a requisição, a processa e encaminha uma resposta para o cliente. Tipicamente um telefone SIP possui um agente cliente e um agente servidor.

As mensagens SIP possuem uma linha inicial indicando a natureza da mensagem, uma sequência de cabeçalhos seguidos de uma linha em branco e, finalmente, um corpo da mensagem que é opcional.

Exemplo de Estabelecimento/Encerramento de Chamada

Aqui será detalhado um exemplo básico de comunicação utilizando o protocolo SIP, descrevendo a sequência de troca de mensagens (PISA, 2008).

A Figura 2.3 contém o exemplo de uma sessão SIP. Ela ilustra o desejo do “usuario1” em chamar o “usuario2”. Como pode ser observado, o início da sessão se dá quando o “usuario1” envia um INVITE ao “usuario2”. Esta mensagem possui o identificador do usuário, além do tipo de dado que ele deseja receber, o protocolo de transporte e a porta onde receberá os pacotes. Ela é enviada por *User Datagram Protocol* (UDP) na porta 5060, que é utilizada como porta padrão nesse tipo de comunicação com o protocolo SIP.

Após recebimento do INVITE, o “usuario2” responde com a mensagem “200 OK”, que possui como conteúdo seu endereço IP, a porta em que deseja receber os dados, o tipo de protocolo de transporte, e a codificação.

Concluindo a troca de mensagens inicial, é então realizado o envio de uma confirmação de estabelecimento de comunicação por parte do “usuario1”. Essa afirmação de sucesso é feita através da mensagem ACK.

Estabelecida a sessão, ambos enviarão os dados conforme características especificadas.

Para terminar a sessão é utilizada a mensagem BYE, que pode ser enviada por qualquer um dos dois usuários. Como confirmação é feito envio com a “200 OK”.

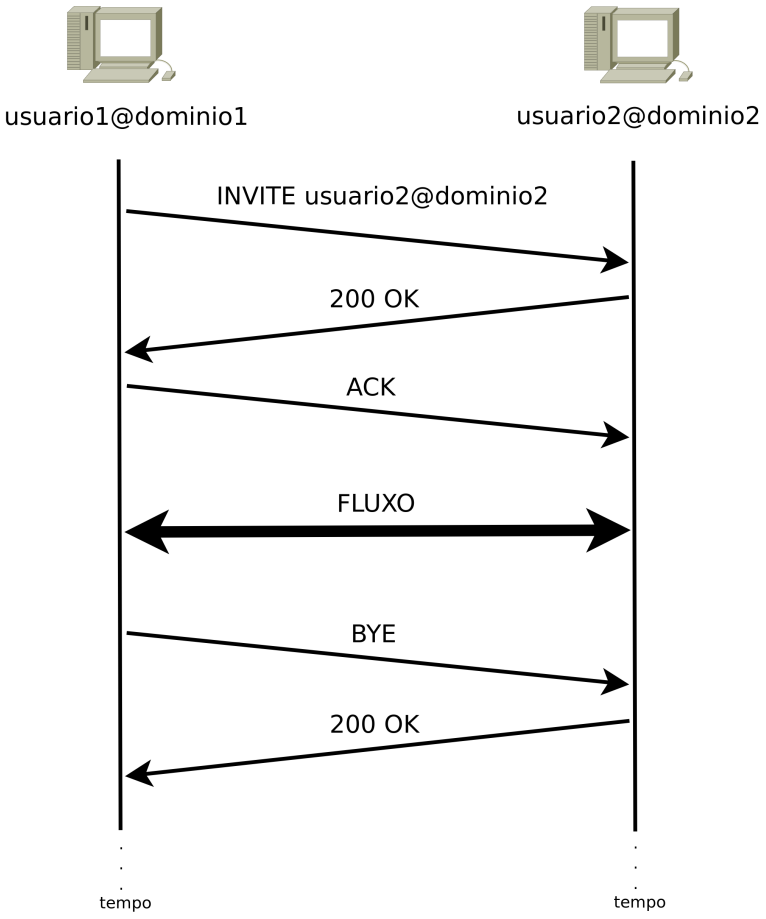

Figura 2.3: Sessão SIP.

Um exemplo de uma mensagem INVITE (sem o corpo da mensagem que, no caso, é uma mensagem SDP) é mostrada na Figura 2.4. Há campos de grande importância, como o VIA, cuja interpretação foi necessária para este trabalho. É nele que são guardadas informações do caminho percorrido pelo pacote, ou o caminho a percorrer. Estão ainda indicados neste campo, o tipo de conexão para o transporte, o endereço para aguardo de resposta ao pedido, a porta onde utilizada na recepção da resposta, e a identificação da transação (parâmetro BRANCH).

Em síntese, o campo VIA determina onde o requisitante espera receber a resposta para a sua requisição. No caso da mensagem do exemplo, o requisitante espera que a resposta seja enviada para o endereço IP 127.0.0.1, porta 5060, usando o protocolo UDP. Na resposta a transação será identificada pelo *tag* BRANCH, que será uma cópia da cadeia BRANCH da requisição. Servidores PROXY SIP poderão acrescentar campos VIA na mensagem, de forma que a resposta possa ser roteada pelos diversos servidores por onde passou.

```
▽ Session Initiation Protocol
  ▽ Request-Line: INVITE sip:service@10.0.0.1:5060 SIP/2.0
      Method: INVITE
    ▽ Request-URI: sip:service@10.0.0.1:5060
        Request-URI User Part: service
        Request-URI Host Part: 10.0.0.1
        Request-URI Host Port: 5060
      [Resent Packet: False]
  ▽ Message Header
    ▽ Via: SIP/2.0/UDP 127.0.0.1:5060;branch=z9hG4bK-548-1-0
        Transport: UDP
        Sent-by Address: 127.0.0.1
        Sent-by port: 5060
        Branch: z9hG4bK-548-1-0
    ▽ From: sipp <sip:sipp@127.0.0.1:5060>;tag=548SIPpTag001
        SIP Display info: sipp
      ▽ SIP from address: sip:sipp@127.0.0.1:5060
          SIP from address User Part: sipp
          SIP from address Host Part: 127.0.0.1
          SIP from address Host Port: 5060
        SIP tag: 548SIPpTag001
    ▽ To: sut <sip:service@10.0.0.1:5060>
        SIP Display info: sut
      ▽ SIP to address: sip:service@10.0.0.1:5060
          SIP to address User Part: service
          SIP to address Host Part: 10.0.0.1
          SIP to address Host Port: 5060
      Call-ID: 1-548@127.0.0.1
    ▽ CSeq: 1 INVITE
        Sequence Number: 1
        Method: INVITE
    ▽ Contact: sip:sipp@127.0.0.1:5060
      ▽ Contact Binding: sip:sipp@127.0.0.1:5060
        ▽ URI: sip:sipp@127.0.0.1:5060
            SIP contact address: sip:sipp@127.0.0.1:5060
      Max-Forwards: 70
      Subject: Performance Test
      Content-Type: application/sdp
      Content-Length:   129
  ▷ Message Body
```

Figura 2.4: Cabeçalho da Mensagem INVITE.

O campo FROM indica de onde foi originado o pacote, mostrando com detalhes o nome do usuário e endereço *Uniform Resource Identifier* (URI). Indica o originador da chamada, seu endereço, a porta utilizada na comunicação, e a "tag SIP" contendo uma *string* aleatória com propósito de identificação.

O TO possui as mesmas características do campo FROM, porém mostra para quem foi

direcionado o pacote. O CALL-ID possui o identificador da chamada. Já o CSEQ indica o metódo e o número de sequência da chamada. Ele é incrementado para cada novo pedido dentro de um diálogo.

O campo CONTACT é usado para indicar aos agentes comunicantes onde enviar requisições futuras. Contém a URI SIP, o número máximo de saltos, o assunto, tipo de conteúdo, e o tamanho.

A Arquitetura de servidores SIP

O SIP permite implementar uma arquitetura de servidores definida para apoiar na localização, encaminhamento de mensagens e implementação de regras de comunicação. É importante salientar que são componentes funcionais, podendo uma mesma máquina comportar vários deles. Segue descrição de cada um:

- ***User Agent* (UA):** é capaz de fazer pedidos de inicialização de sessão *User Agent Client* (UAC) e também de responder a requisições *User Agent Server* (UAS), operando com o conceito Cliente-Servidor. Os pontos finais da sinalização SIP, tipicamente um telefone IP ou *softphone* se utiliza de um UA cliente e um UA servidor de forma a permiti-lo gerar e receber requisições;

- ***PROXY Server*:** atua como intermediário em uma comunicação SIP, encaminhando as mensagens de outros clientes que não podem fazer as requisições diretamente. Ele pode ser utilizado nos domínios SIP para realizar autenticação e implementar regras de uso do sistema. A formação em trapézio mostrada na Figura 2.5 ilustra este servidor;

- ***Redirect Server*:** fornece aos *User Agent*s (UAs) informações sobre o endereço do servidor atual, para que possa contatar o endereço diretamente.

- ***Registrar Server*:** é o servidor de registros que armazena informações de localização dos UAs. Trabalha em conjunto com os outros servidores. Tipicamente um telefone IP ou *softphone* se registra com este servidor, de forma a permitir mapear o endereço SIP URI (um endereço lógico) em um endereço físico IP onde o telefone está. Normalmente um servidor de registros fica coalocado com o servidor PROXY do domínio SIP.

Uma estrutura típica referida como "trapézio SIP" é apresentada na Figura 2.5 (ROSENBERG et al., 2002). O nome se justifica pela forma em que estão conectados os clientes e servidores. Através do desenho formado pelo pontilhado é possível visualizar um trapézio.

Nessa estrutura, do ponto de vista de cada terminal, existe um servidor SIP em seu domínio, para onde devem ser encaminhados as mensagens INVITE. O endereço deste servidor PROXY pode ser estaticamente configurado ou pode ser dinamicamente recebido por um servidor DHCP.

```
                 atlanta.com  . . . biloxi.com
            .      proxy              proxy     .
          .                                      .
 Alice's  . . . . . . . . . . . . . . . . . . . .  Bob's
softphone                                        SIP Phone
   |                |                |                |
   |    INVITE F1   |                |                |
   |--------------->|    INVITE F2   |                |
   |  100 Trying F3 |--------------->|    INVITE F4   |
   |<---------------|  100 Trying F5 |--------------->|
   |                |<-------------- | 180 Ringing F6 |
   |                | 180 Ringing F7 |<---------------|
   | 180 Ringing F8 |<---------------|     200 OK F9  |
   |<---------------|    200 OK F10  |<---------------|
   |    200 OK F11  |<---------------|                |
   |<---------------|                |                |
   |                         ACK F12                  |
   |------------------------------------------------->|
   |                    Media Session                 |
   |<=================================================>|
   |                         BYE F13                  |
   |<-------------------------------------------------|
   |                      200 OK F14                  |
   |------------------------------------------------->|
   |                                                  |
```

Figura 2.5: Trapézio SIP (ROSENBERG et al., 2002).

No exemplo há dois servidores PROXY devidamente identificados, além de dois usuários, Alice e Bob. Cada usuário possue um domínio, são eles: “atlanta.com” e “biloxi.com”. Note que estes domínios permitem localizar os servidores PROXY de cada domínio.

Anexo à estrutura há também um exemplo de chamada SIP. Nele, além das mensagens trocadas pelos clientes, estão também representadas as mensagens enviadas pelos servidores. Tais mensagens recebem uma numeração, percedida pela letra ’F’, que indica a sequência em que são enviadas.

A mensagem INVITE, nomeada como “F1” é enviada por Alice a seu servidor. Ela tem como objetivo convidar o usuário Bob para uma sessão de comunicação. Ao receber o INVITE (F2), o servidor PROXY de Alice realiza a requisição INVITE ao servidor que está ligado a ele, o “biloxi.com”. Em seguida responde para Alice com a mensagem TRYING (F3), confirmando o recebimento e indicando estar em busca do usuário.

O servidor “biloxi.com” age da mesma maneira, respondendo com TRYING a quem o requisitou (F5), e encaminhando o INVITE (F4).

O usuário Bob, ao receber a mensagem INVITE, responde com um RINGING (F6) e um "200 OK" (F9), que são encaminhadas ao endereço de Alice através dos servidores. Nestas mensagens, Bob informa sua disponibilidade para entrar em contato e indica sua localização. Assim, a comunicação poderá ser feita diretamente pelos clientes.

Alice responde com ACK (F12), e em seguida estabelecem a sessão. A troca de pacotes pode ser feita então de hospedeiro a hospedeiro (diretamente entre os pontos comunicantes).Qualquer um dos usuários pode tomar a iniciativa de finalizar a sessão. Basta solicitar através da mensagem BYE, que no exemplo é enviada por Bob (F13).

2.2.2 O Protocolo SIP e a Mobilidade

O protocolo SIP possui mecanismos que permitem tratar a mobilidade **pessoal**, de **serviço**, de **sessão**, e de **terminal**, tal como conceituadas na sessão 2.1.

Na Figura 2.6 é mostrado um exemplo de mobilidade de **sessão** usando transferência de chamada. Inicialmente a comunicação é realizada entre os dispositivos A e B1. Em certo momento da chamada, o usuário de B1 deseja fazer uma mobilidade para o dispositivo B2. Para tal mudança ter sucesso ele deve comunicar ao dispositivo A que destino ele deverá atingir. Para isso ele deve utilizar o método REFER, indicando o novo dispositivo.

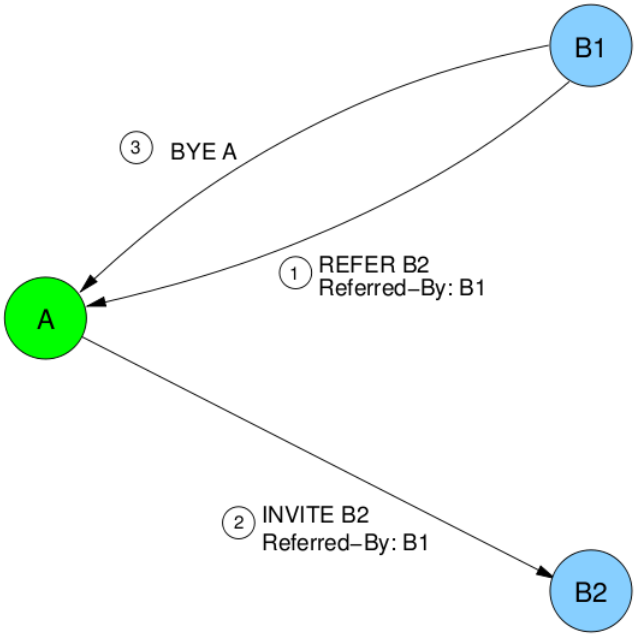

Figura 2.6: Exemplo de Mobilidade de Sessão (SCHULZRINNE, 2001).

Após conhecer o novo destino em que deve encaminhar os pacotes, o dispositivo A deve lhe enviar uma nova mensagem INVITE para estabelecimento da conexão. Junto as mensagens

REFER e INVITE segue a descrição de quem está fazendo referência.

A mobilidade de **serviços** está associada a troca de dispositivos por parte do *host*. Nesse caso, os serviços disponíveis no dispositivo utilizado inicialmente (como configurações pessoais, endereços, lista de chamadas), estarão disponíveis também no dispositivo ao qual está conectado atualmente.

A Figura 2.7 (SCHULZRINNE, 2001) ilustra um exemplo de mobilidade **pessoal**. Com ela é possível o contato com um usuário localizado em diferentes terminais utilizando um único endereço. Podemos visualizar através da Figura, que, a usuária Alice está disponível através dos endereços alice17@yahoo.com, alice@columbia.edu, 7000@columbia.edu, Alice.McBeal@columbia.edu. Com esses endereços e suas devidas tecnologias ela pode utilizar tanto a rede de telefonia convencional - Rede Pública de Telefonia Comutada (RPTC), quanto um dispositivo sem fio, ou ainda, um *Personal Computer* (PC).

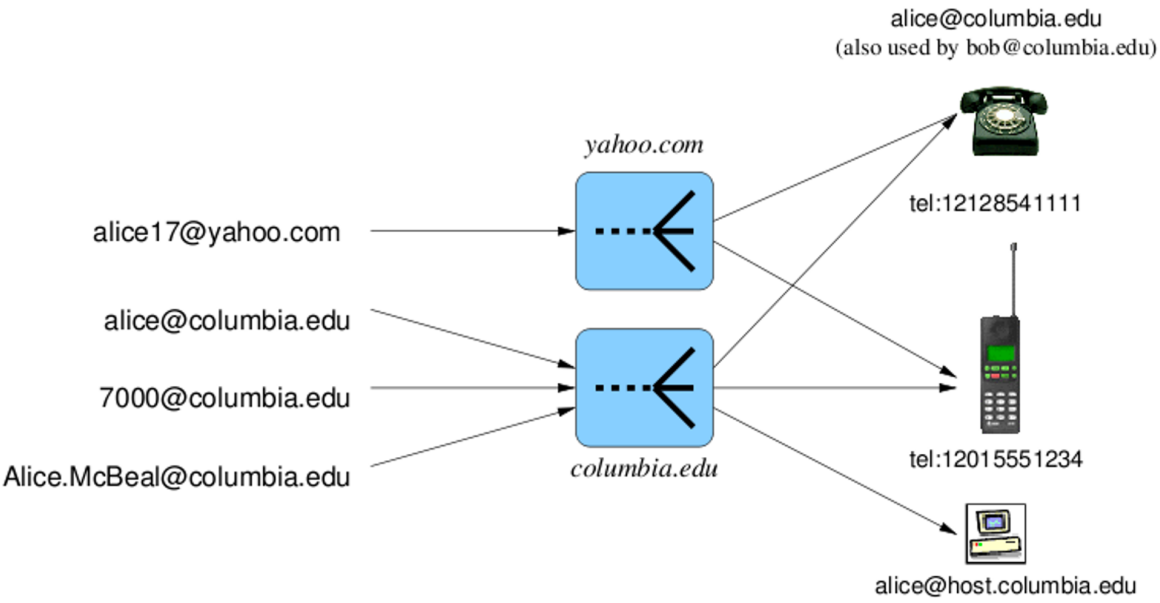

Figura 2.7: Exemplo de Mobilidade Pessoal (SCHULZRINNE, 2001).

Na mobilidade **terminal**, foco deste trabalho, o usuário pode ser encontrado após uma requisição feita na sua rede doméstica, mesmo estando em uma rede externa. Três procedimentos podem ser destacados neste processo: o re-registro, uma nova chamada após mobilidade, e a re-sinalização de uma sessão em andamento.

Inicialmente, ao mudar de rede, o usuário móvel se re-registra com seu servidor de registro (Figura 2.8), informando uma nova associação entre seu SIP URI (campo To) e seu novo endereço IP (campo Contact). Grande parte deste trabalho será dedicado a este procedimento.

Na Figura 2.9, reproduzindo (SCHULZRINNE; WEDLUND, 2000), é possível observar

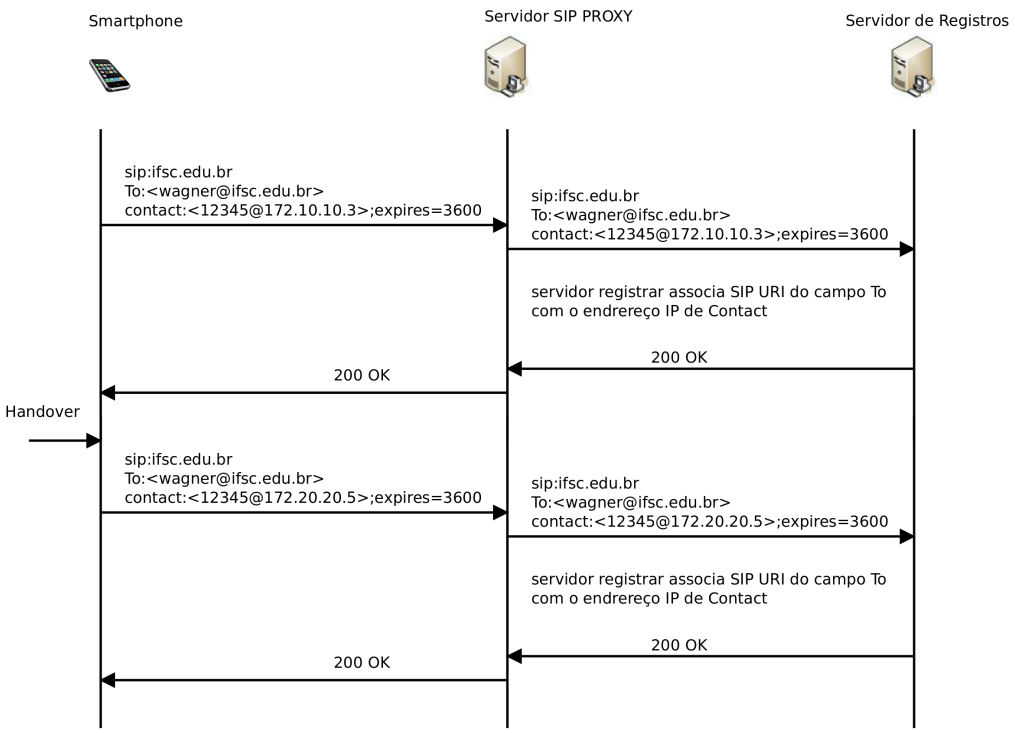

Figura 2.8: Operação de Registro Antes e Após Mobilidade.

um exemplo do procedimento, associando uma nova chamada de um terminal correspondente (*Correspondent Host* (CH)) a um terminal móvel (*Mobile Host* (MH)) que já se encontra em uma rede visitada.

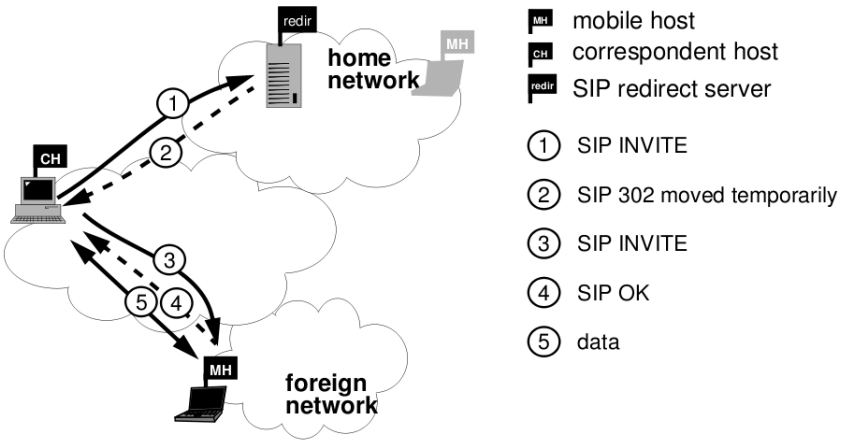

Figura 2.9: Exemplo de Mobilidade Terminal - Nova Chamada (SCHULZRINNE; WEDLUND, 2000).

Quando o terminal móvel é invocado pelo terminal correspondente a seguinte sequência de eventos ocorre: em sua *home network*, o servidor SIP (servidor de redirecionamento) informa a quem requisita (1), que o *hospedeiro* moveu-se e indica seu novo endereço (2). Consequentemente a requisição (3) é feita diretamente ao novo endereço. O MH é então encontrado, e res-

ponde à requisição (4), sendo estabelecida a comunicação (5). O servidor de redirecionamento pode ser coalocado com um PROXY SIP. Neste caso, o próprio servidor pode encaminhar o INVITE para o MH.

Um terceiro procedimento, a ser discutido na Seção 4.1 e mostrado na Figura 2.10, é necessário para que sessões em andamento sejam atualizadas com o novo endereço do terminal móvel. Neste caso é necessário que o terminal móvel envie um re-INVITE para o terminal comunicante. Este INVITE transporta uma mensagem SDP com o novo endereço IP. Note que isto é feito independente de o terminal móvel ter originado a comunicação. Finalmente, o terminal correspondente reajusta o protocolo de transporte de mídia (possivelmente o RTP) de forma que o fluxo seja encaminhado para o novo destino.

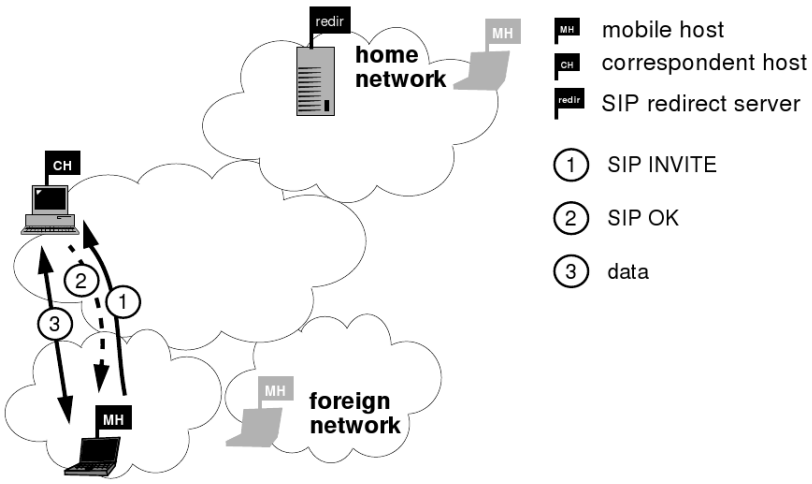

Figura 2.10: Exemplo de Mobilidade Terminal - Sessão em Andamento (SCHULZRINNE; WEDLUND, 2000).

2.3 O Protocolo DHCP

Conforme comentado anteriormente, o protocolo DHCP impacta o desempenho do SIP nos procedimentos pós-mobilidade. O DHCP é um protocolo usado para fornecer parâmetros de configuração de rede a hospedeiros, dentre esses parâmetros está o endereço IP. O DHCP é composto de um modelo cliente-servidor, no qual o servidor mantém as informações de configuração e as repassa aos clientes quando realizada alguma solicitação. Ele é o sucessor do BOOTP, e sua última versão (para IPv6) foi publicada em julho de 2003 na RFC 3315.

Graças a um servidor DHCP instalado e configurado corretamente na rede, os clientes que iniciam uma conexão nessa mesma rede e que estão ativos para o uso do DHCP podem obter dinamicamente um endereço IP e parâmetros relacionados. As configurações são passadas

como oferta de concessão de endereços aos clientes que efetuam o pedido.

Conforme (DROMS, 1997, RFC 2131) o DHCP possui três maneiras de atribuição de endereços IP:

- ***Automatic Allocation*:** O servidor DHCP atribui um endereço IP permanente para um cliente.
- ***Alocação Dinâmica*:** O servidor DHCP atribui um endereço IP a um cliente por um período de tempo limitado (ou até que o cliente renuncie expressamente o endereço).
- ***Alocação Manual*:** O endereço IP do cliente é atribuído pelo administrador da rede (sendo fixado através do MAC - Media Access Control), e o DHCP é usado apenas para transmitir o endereço já atribuído ao cliente. A rede poderá utilizar um ou mais desses mecanismos, dependendo das políticas do administrador da rede.

A Alocação Dinâmica é o único dos três mecanismos que permite a reutilização automática de um endereço que não é mais usado pelo cliente para o qual foi designado. Assim, a alocação dinâmica é particularmente útil para atribuir um endereço a um cliente que será conectado à rede apenas temporariamente, junto a um grupo de clientes que não precisam de endereço IP permanente.

Os servidores DHCP possuem campos em seus arquivos de configuração que possibilitam ajustes de valores em seus temporizadores.

No arquivo de configuração do *dhcp3-server* existe a opção 'default-lease-time VALOR', na qual o servidor verifica se a estação ainda está ativa quando decorrido o tempo indicado em VALOR. A linha 'max-lease-time VALOR' indica o tempo máximo que uma estação pode usar determinado endereço IP, caso seja requisitado um tempo diferente do 'default-lease-time' por parte do cliente.

Finalmente, pode-se destacar uma característica interessante do DHCP para fornecer dinamicamente o endereço de um servidor PROXY SIP via DHCP. Esta característica é definida na RFC3361. Desta forma, na mobilidade para redes ou domínios que exigem usar seu próprio servidor SIP, o DHCP pode forncer o endereço do mesmo. Neste caso pode ser necessário lançar mão de hierarquias de servidores SIP (SCHULZRINNE; WEDLUND, 2000).

2.4 Sistemas Embarcados

A plataforma de testes usada neste trabalho foi direcionada para uso em sistemas embarcados. O conceito de sistemas embarcados abrange uma ampla área da tecnologia existente nos dispositivos atuais. Trata-se de um sistema responsável pela execução dedicada de uma função específica ou um conjunto de funções que possuem relação entre si, ao contrário de um PC que pode executar e alternar entre diversos programas e aplicativos. Segundo Morimoto (MORIMOTO, 2009), sistemas embarcados são dispositivos "invisíveis", que se fundem no nosso cotidiano, de forma que muitas vezes sequer percebemos que eles estão lá, tendo como única diferença a limitação de executar uma única tarefa com continuidade e normalmente sem travamentos.

O Sistema Operacional Linux se torna uma excelente opção para implementações em sistemas embarcados, pois possui código fonte de qualidade disponível na internet, suporte em fóruns e listas, além da redução de custos proporcionada em estudos, implementações e afins (PINHEIRO, 2007). Deve ser salientado que existem implementações do Linux apropriadas para execução de sistemas com restrições de tempo. Nestes sistemas existem facilidades para o escalonamento de tarefas com necessidades extremas de respeitar linhas de tempo na execução (UDUGAMA, 2006).

Uma comparação entre o sistema Linux tradicional e a versão para sistemas embarcados aponta várias reduções de recursos como interface diferenciada, utilitários com implementações mais leves - consumindo menos recursos, e o próprio *Kernel*, contendo somente os *drivers* necessários para o pleno funcionamento do sistema.

A escolha do sistema apontou o Linux como opção mais viável (mais especificamente o *ubuntu-minimal*), pois além dos benefícios já listados anteriormente, o sistema possui vários aplicativos com tecnologias atuais e em estudo, e acima de tudo é distribuído como um *software* de código aberto e gratuito.

O foco em sistemas embarcados também levou a escolha de uma aplicação/biblioteca apropriada para os testes de cenários de mobilidade com registro em um servidor VoIP - *Asterisk*. No capítulo 4, a aplicação escolhida, o PJSUA, será visto com mais detalhes.

3 A Mobilidade entre Redes e o uDHCPc

Conforme citado no Capítulo 1, a capacidade de resposta do SIP frente a mobilidade entre sub-redes depende fortemente da capacidade do terminal estabelecer um novo endereço IP. O DHCP é um protocolo que é amplamente usado para a configuração dinâmica de terminais, incluindo a configuração de endereços IP, sendo natural que ele seja inicialmente investigado.

Este capítulo visa analisar o comportamento do DHCP em um contexto de mobilidade no âmbito de sistemas embarcados. Inicialmente será apresentado o uDHCPc, uma versão de cliente DHCP para sistemas embarcados. Uma primeira avaliação da capacidade de resposta do uDHCPc frente a mobilidade é então realizada.

Para contornar o baixo desempenho observado na resposta do uDHCPc, é então proposta uma solução de detecção de mudança de rede usando um *shell script*, composto por ferramentas como *ip monitor*, *awk*, *iwconfig* e outros. Esta estrutura será usada no Capítulo 4 em conjunto com o protocolo SIP.

3.1 O Micro Cliente DHCP

3.1.1 Um DHCP para Sistema Embarcados

O uDHCPc (Micro Dynamic Host Configuration Protocol Client) é uma pequena versão cliente do DHCP, voltada para sistemas embarcados. Apesar de ser uma versão reduzida, ela visa ser totalmente funcional e compatível com a RFC 2131. Encontra-se disponível para *download* através do pacote do projeto Busy-Box (ANDERSEN, 2008a) e também através do gerenciador de pacotes *apt-get*.

O uDHCPc depende apenas da biblioteca C e pode ser compilado tanto com a *glibc* (padrão da GNU) como com a uClibc (ANDERSEN, 2008b), que é uma biblioteca de tamanho reduzido para sistemas embarcados.

Quando compilado com a *glibc* e descompactado, o uDHCPc ocupa cerca de 16 KB de

espaço, quando está ligado dinamicamente. Quando estático ocupa cerca de 375 KB, também descompactado e compilado com glibc. Já compilado com uClibc e descompactado, o uDHCPc fica com cerca de 15 KB de tamanho quando está ligado dinamicamente, e 40 KB quando está ligado estaticamente.

3.1.2 Estrutura e Operação

Na estrutura do uDHCPc existem chamadas a *scripts* padrões, que são executados quando ocorre um evento (seja ele aquisição ou perda de endereço). Com a execução do *script* busca-se obter flexibilidade na realização de ações quando existe mudanças no endereçamento da interface.

O *script* principal é o *default.script*. É ele quem evoca os outros scripts do uDHCPc. São*scripts* padrões do uDHCPc: *default.bound*; *default.deconfig*; *default.leasefail*; *default.nak*; *default.renew*; e o já citado *default.script*. A função de cada um deles é:

- **Deconfig**: é usado quando o uDHCPc é iniciado e quando um aluguel é perdido. O *script* coloca a interface como *UP*, mas desconfigura o estado, e define o endereço IP da interface como 0.0.0.0. Nesse *script* é também realizada a exclusão das configurações da interface.

- **Bound**: o uso do *bound* é feito quando o endereço IP é confirmado através do ACK vindo do servidor. O *script* deve configurar a interface definindo alguns parâmetros relevantes como por exemplo *gateway* padrão, servidor DNS, *hostname*, etc. A partir deste momento a interface se torna operacional;

- **Renew**: a utilização do *renew* se dá quando um aluguel de endereço é renovado. Como a interface já está configurada o endereço IP não é alterado. Já outros parâmetros como o *gateway* padrão, a máscara de sub-rede, o servidor DNS podem sofrer modificações.

- **Nak**: é usado quando o uDHCPc recebe um pacote NAK do servidor. Quando isso ocorrer, a variável de ambiente *$message* terá o conteúdo da mensagem NAK. O processamento desta mensagem é opcional. O *script* será chamado com o *deconfig* se for necessário.

Na operação do uDHCPc, inicialmente servidor e cliente não se conhecem. Então o cliente, interessado em adquirir a configuração, envia uma requisição (DHCP DISCOVER) através de um pacote UDP pela porta 67 ao endereço *broadcast*. Assim a informação é repassada a todos

os servidores DHCP da rede. Ao receber esse pacote com pedido de configuração de rede o(s) servidor(es) analisará(ão), a partir de seus critérios de atribuição de IP's, se o cliente pode ser atendido. Se o resultado for positivo fará o envio de um pacote, contendo endereço IP disponível e demais configurações de rede ao cliente solicitante. Adicionalmente o servidor poderá fornecer endereço de *gateway*, endereços de servidores DNS, endereço *broadcast*, e outras configurações da rede. O cliente seleciona a oferta desejada e requisita o aluguel ao servidor selecionado.

A máquina de estados, Figura 3.1, indica o funcionamento geral do uDHCPc. Nela são indicados os eventos e ações realizadas pela aplicação. A ilustração foi construída com base no material disponível no *TCP/IP Guide* (KOZIEROK, 2005), com adaptações para o uDHCPc.

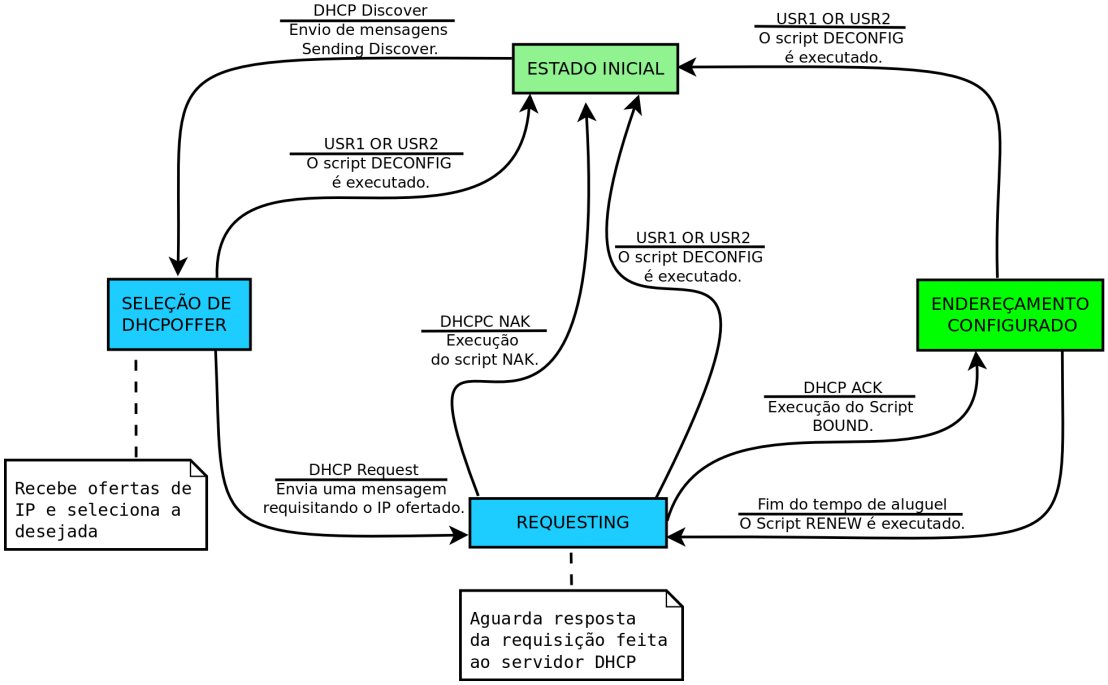

Figura 3.1: Máquina de Estados uDHCPc.

São mostrados somente alguns eventos possíveis em um ambiente utilizando o uDHCPc, visto que um diagrama englobando todas as ações se tornaria complexo demais.

A imagem indica no topo o estado inicial, na qual o processo de aquisição de endereço é iniciado. Esse estado pode ser atingido se por ventura houver negação ou falha na negociação. O evento que busca a modificação do estado inicial para um próximo estado é o *DHCP Discover*, com ele são enviadas mensagens *Sending Discover* buscando servidores DHCP que estão disponíveis no momento. O estado seguinte é o de seleção, atingido após obter ofertas de endereçamento IP de servidores DHCP. Após ser feita decisão, o próximo passo é transmitir uma mensagem requisitando tal endereçamento ofertado. Em seguida o cliente fica no estado

Requesting, esperando ouvir a resposta do servidor.

No estado *Requesting* também são possíveis diferentes rumos. Se ocorrer algo inesperado, como por exemplo, a oferta passar da validade ou o servidor negar o aluguel, uma mensagem DHCP NAK é recebida pelo cliente e seu estado volta ao inicial. O mesmo pode ocorrer no caso de recebido dos sinais USR1 ou USR2 por parte do uDHCPc. Esta característica será explorada para melhorar o desempenho do uDHCPc na aquisição de novo endereço em condições de mobilidade.

Caso haja sucesso na concessão, o servidor confirma a reserva do endereço com a mensagem DHCP ACK e repassa as configurações da rede. A partir disso é executado o *script default.bound*, que configura a interface e a leva a seu estado normal de funcionamento. Após essa obtenção de endereçamento pode ocorrer o fim de tempo de concessão, nesse caso há o *script* de renovação que é o *default.renew*.

3.2 Avaliação da Capacidade de Resposta do uDHCPc Frente a Mobilidade

3.2.1 Objetivos do Experimento

Nesta seção é apresentado um ensaio para avaliar a capacidade de resposta do uDHCPc sem nenhum apoio de detectores de estado da rede em nível de enlace. O foco deste estudo é a medição do tempo que leva o estabelecimento de um novo endereço de rede com o uso do uDHCPc.

Seja E_{up} e E_{down} os eventos que denotam o momento em que uma interface se conecta ou desconecta, em termos de enlace, a uma rede de acesso. Por exemplo, E_{up} pode ser, no caso de uma rede *ethernet*, o momento de detecção de "portadora" na rede. No caso de uma rede sem fio pode ser o momento de uma confirmação de associação com um ponto de acesso. Após este evento, a interface entra em estado *UP* e deve começar o processo de aquisição de um novo endereço, no caso com o DHCP. O evento E_{cfg} denota o momento que a interface estabelece o endereço IP válido. O tempo T_{aq} denotará o período de tempo transcorrido desde o evento E_{up} até o momento da configuração da interface com um endereço IP válido, ou seja:

$$T_{aq} = T_{E_{cfg}} - T_{E_{up}} \tag{3.1}$$

Deve ser observado que o interesse do experimento não é avaliar o tempo transcorrido entre

um evento E_{down} e a descoberta de uma nova rede resultando em um evento E_{up}. Este tempo é dependente das tecnologias envolvidas no enlace. A Figura3.2 ilustra o tempo de aquisição T_{aq}.

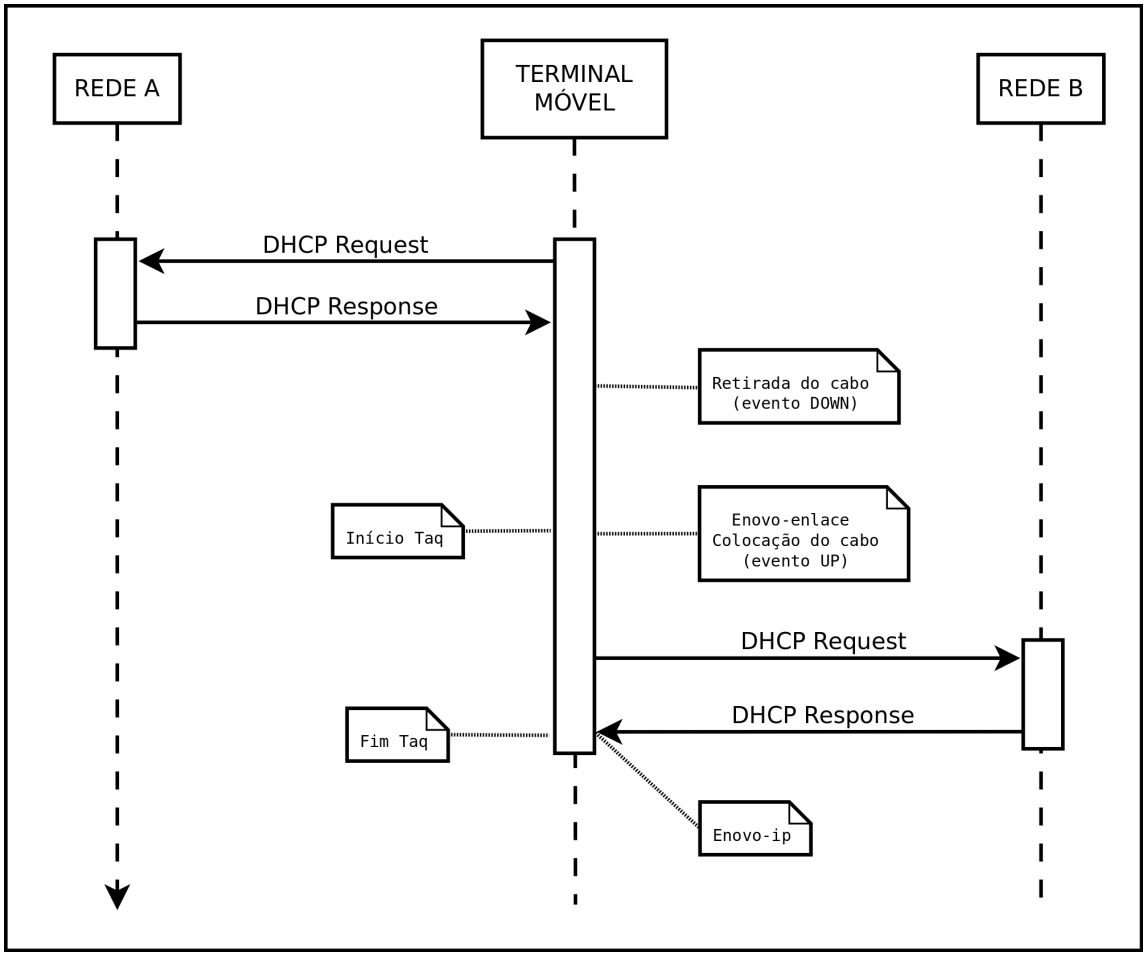

Figura 3.2: Movimentação entre Redes.

3.2.2 Descrição do Cenário

O cenário utilizado foi composto por dois servidores DHCP, e um microcomputador com função de cliente. Um dos servidores DHCP utilizados foi o disponível na rede do IFSC, já o outro foi configurado em um microcomputador do LabIC, fornecendo apenas o IP 10.10.10.10. A rede 172.18.0.0/16 representa a rede da Instituição, que fornece IPs com este prefixo.

Os componentes do cenário são ilustrados na Figura 3.3, bem como a indicação da troca de rede e os números IP posteriormente fornecidos.

Inicialmente o estudo é feito através de interfaces ethernet, pois o objetivo é analisar a capacidade do uDHCPc de adquirir um novo endereço IP após a restauração da conexão física (cabo de rede).

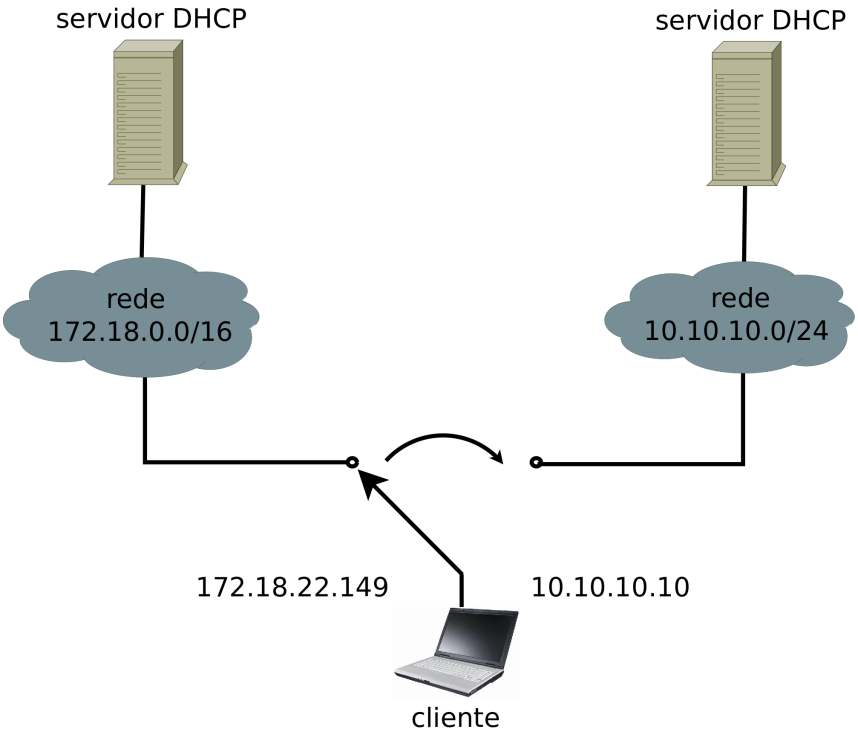

Figura 3.3: Cenário de Troca de Redes.

O *Software* no Terminal

Uma das diretrizes básicas para a realização dos experimentos é reproduzir um ambiente de execução com o sistema Linux em um sistema embarcado, onde possivelmente não estão presentes ferramentas tradicionais de um sistema *desktop*. Neste sentido procurou-se reproduzir um sistema Linux mínimo, desabilitando o *Network Manager* e o *dhclient*, que estão diretamente ligados à obtenção de endereçamento IP.

Nos teste foram desativados a partir de linhas de comando, incorporadas ao *shell script* final, automatizando assim o processo. Na linha de comando é dado um *stop* no serviço *network-manager* (service network-manager stop) e um *killall* em todos processos nomeados com *dhclient* (*killall dhclient*). Esses procesos são desativados procurando tornar as interferências no sistema as menores possíveis, assim como em um sistema Linux mínimo, que possui somente os pacotes básicos de alguns serviços, e mesmo assim proporciona um pleno funcionamento.

O uDHCPc foi instalado no terminal e não precisou de maiores configurações para poder ser inicializado. Para seu funcionamento bastou a execução do comando abaixo, indicando opcionalmente a interface utilizada.

 # udhcpc -i eth0

Finalmente, o comando *ip* foi utilizado em conjunto com alguns *scripts* com fins de monitorar o estado da interface. O comando *ip* possui diversas opções que vão desde a visualização de configurações de rede (para solução de problemas), até a configuração da rede propriamente dita (como por exemplo a atribuição de endereço IP manualmente). A opção "monitor" é uma das opções do comando *ip*. Essa opção serve para monitorar de forma contínua o estado da interface, e também o estado das rotas e endereços. Neste modo, o comando acessa diretamente a camada RTNETLINK (UDUGAMA, 2006) do *kernel* do Linux. Sua sintaxe é:

ip monitor [all | LISTofOBJECTS]

O parâmetro passado em LISTofOBJECTS é a lista de objetos que se deseja monitorar, pode conter link, endereço e rota. Se nenhum argumento é passado, o comando *ip* abre o RTNETLINK.

No caso do experimento serão utilizados comandos de filtragem como o *grep* e o *awk*, juntamente com a facilidade do *pipe* para auxiliar o comando *ip monitor*.

O *Software* do *desktop*

Com o intuito de realizar uma mudança de rede, e sabendo que havia um servidor DHCP disponível na rede IFSC, bastava a configuração de um outro servidor DHCP simples.

A configuração básica, realizada no microcomputador disponível em laboratório, pode ser vista na sequência, através da Figura 3.4.

```
ddns-update-style none;

default-lease-time 30;
max-lease-time 7200;

log-facility local7;

subnet 10.10.10.0 netmask 255.255.255.0 {
        range 10.10.10.10 10.10.10.10;
        option routers 10.10.10.1;
}
```

Figura 3.4: Configuração do Servidor DHCP.

A inserção feita no arquivo foi com o objetivo de criar o fornecimento de IP para a rede com prefixo 10.10.10.0. Na faixa de IP foi disponibilizado apenas um endereço IP, visto que era isso o necessário para o experimento. O arquivo a ser editado foi o dhcpd.conf, localizado na pasta *default* do aplicativo.

Além dessa modificação, foi realizada alteração no arquivo "/etc/default/dhcp3-server", in-

dicando o valor da interface padrão (eth1) para o parâmetro "INTERFACES". A partir desta interface seria então realizado o provimento de endereços IP.

3.2.3 Procedimento do Experimento

O procedimento do experimento pode ser descrito pelos seguintes passos:

1. Inicialmente o uDHCPc é iniciado. Em seguida o terminal móvel é conectado à rede IFSC através da sua interface eth0. Neste momento, o uDHCPc adquire um endereço IP do servidor DHCP da rede institucional;

2. O cabo da rede é então desconectado. Este evento de desconexão é monitorado pelo comando *ip monitor* e é mostrado como a linha:

   2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN

   link/ether 00:16:56:86:e5:94 brd ff:ff:ff:ff:ff:ff

   A captura do momento em que o cliente recebe o pacote DHCP é realizada com o *tcpdump*, e um *script* auxiliar, combinando o *ip monitor* com o comando *date* (ver anexo A). Esta estrutura permite identificar os intervalos de tempo exatos tomados por estes eventos;

3. O cabo de rede é conectado na rede 10.10.10.0. O *ip monitor* novamente realiza a captura de um evento, desta vez o "UP", conforme mostra a linha:

   2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP

   link/ether 00:16:56:86:e5:94 brd ff:ff:ff:ff:ff:ff

4. Após certo tempo, o terminal móvel adquire um novo endereço do servidor DHCP da rede 10.10.10.0. O uDHCPc não é informado de imediato do evento físico e por isso renova seu endereço seguindo o valor *default* de seus temporizadores;

5. As amostras de tempo associadas aos eventos do uDHCPc são capturadas através dos arquivos de log do *tcpdump*, além de visualizadas em modo gráfico com o *Wireshark*. A aquisição do novo endereço IP também é detectada pelo *log* do *ip monitor*.

Na Figura 3.5, pode ser visualizada a mensagem DHCP capturada pelo comando *tcpdump*. Ela indica a estampa de tempo em que a mensagem atingiu a interface.

```
01:30:55.060937 IP wagner-desktop.local.bootpc
> 192.168.1.1.bootps: BOOTP/DHCP, Request from
00:16:56:86:e5:94 (oui Unknown), length 300
```

Figura 3.5: Mensagem DHCP Capturada com *tcpdump*.

Já na Figura 3.6 são disponibilizadas as mensagens de saída do *script* Tempo UP/ DOWN (anexo A). Ele tem o papel de auxiliar na detecção dos eventos DOWN e UP da rede, indicando o momento exato em que ocorrem.

```
...
01:29:49-802114597
>> Paramentro DOWN encontrado <<
...
...
...
01:29:54-409386075
>> Paramentro UP encontrado <<
...
...
```

Figura 3.6: Saída do *Script* "Tempo UP/ DOWN".

3.2.4 Análise dos Resultados Obtidos

Com o procedimento anterior e uma operação simples de diferença do tempo obtido com o *tcpdump* para o tempo destacado pelo *script* (equação 3.1), foi possível montar parte da Tabela 3.1. Esta Tabela mostra dez tomadas de tempo de T_{aq} e a média.

Tempo	01	02	03	04	05	06	07	08	09	10	Média
600 s (default)	61 s	62 s	64 s	62 s	61 s	61 s	64 s	63 s	62 s	63 s	62,3 s

Tabela 3.1: Tomadas de Tempo com Variação do *lease-time*.

Os resultados obtidos com o aplicativo uDHCPc foram os esperados até o momento, considerando que estava trabalhando sozinho, e sabendo-se também, que ele somente partiria para uma nova busca de ofertas, 60 segundos após sua inicialização.

É possível melhorar estes tempos reduzindo o tempo de aluguel mas não pode ser considerada uma abordagem eficiente. Na sequência é testado um esquema para melhorar este tempo de resposta.

3.3 Avaliação de uma Proposta para Melhorar a Capacidade de Resposta do uDHCPc

3.3.1 Objetivos do Experimento

Este experimento é similar ao anterior, mas será utilizado um esquema de detecção de mudança de rede na camada de enlace usando o *ip monitor* e *script* auxiliar. O objetivo será avaliar o tempo de aquisição de um novo endereço após a movimentação entre as redes. Este tempo é representado agora pelo parâmetro T_{aqw}.

Um outro diferencial é que este experimento foi realizado usando rede sem fio Wi-Fi mas medições em rede cabeada também foram realizadas para fins de comparação.

A Figura 3.2 do teste anterior, descrevendo o momento de cada evento, pode ser também considerada para este experimento, considerando-se o evento DOWN como perda de sinal da rede e o UP como confirmação de associação com o novo ponto de acesso.

3.3.2 Descrição do Cenário

O cenário foi formado por dois *Access Point*s (APs), que trabalhavam como servidores DHCP e disponibilizavam diferentes redes através de ondas de rádio. Na "rede A" um dos APs foi configurado para prover IP a rede 10.1.1.0/24, seu endereço era 10.1.1.1. Já os endereços disponíveis no DHCP da "rede B", 10.10.10.0/24, eram oferecidos pelo AP 10.10.10.1.

Para configuração dos APs utilizados foram setadas as opções descritas na Tabela 3.2, que segue:

Acces Point	projeto1	projeto2
endereço	10.1.1.1	10.10.10.1
ESSID	projeto1	projeto2
canal	1	2
faixa de IP	.100 - 105	.100 - 105
lease time	10	10

Tabela 3.2: Configurações Access Points.

Além dos APs, foi utilizado um microcomputador portátil, contendo uma interface de rede sem fio. Esse *notebook* exerceu a função de um cliente, sofrendo mobilidade de uma rede a outra.

A Figura 3.7 ilustra o cenário deste experimento, ilustrando através das setas a movimentação

do cliente.

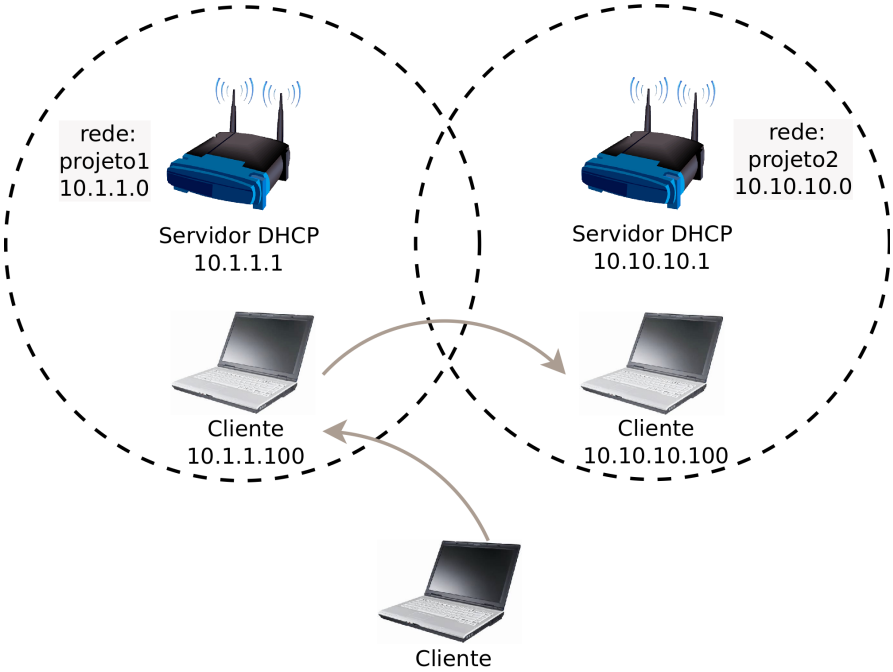

Figura 3.7: Cenário de Redes Sem Fio.

O *Software* no Terminal

Neste experimento foi utilizado um *script* tratando eventos com o *ip monitor* e realizando o envio de sinal à aplicação uDHCPc.

Abaixo é descrita a estrutura criada para tratar os eventos capturados pelo *ip monitor* (ver Figura 3.8). Para auxiliá-lo foram utilizados também os comandos de filtragem *awk* e *grep*.

A variável "var_down" recebe o resultado da filtragem executada a cada leitura de linha do comando *ip monitor*. Na sequência, a primeira estrutura *if*, linha 8, é executada se for encontrado o parâmetro DOWN na variável "var_down". Se for encontrado o DOWN, o script testa qual foi a última rede a se conectar. Sendo a rede "projeto1" por exemplo, ele fará a associação com a rede "projeto2", e em seguida envia os sinais para a aplicação uDHCPc realizar uma nova configuração de rede.

Pode-se notar o uso do comando *iwconfig*. Ele é semelhante ao ifconfig, porém, específico para redes sem fio. Com ele podem ser listadas várias características das interfaces Wi-Fi. O comando permite ainda conectar a um AP específico, passar chave de segurança, determinar o canal utilizado, entre outros. Dentre as facilidades listadas pelo comando, foram utilizadas neste experimento: conexão com um AP específico, através de determinado canal de operação, e com segurança de rede desligada.

```
...

ip monitor | while read linha
do

var_down=$(echo $linha | awk '$8 ~ "state" {print $9}' | grep 'DOWN')

   if [ "$var_down" = 'DOWN' ]; then

      if [ "$rede" = 'projeto1' ]; then
         iwconfig wlan0 mode managed
         iwconfig wlan0 channel 2 key off essid projeto2
         rede=$'projeto2'
         kill -USR2 $pid_udhcpc
         kill -USR1 $pid_udhcpc

      elif [ "$rede" = 'projeto2' ]; then
         iwconfig wlan0 mode managed
         iwconfig wlan0 channel 1 key off essid projeto1
         rede=$'projeto1'
         echo "Rede atual: $rede"
         kill -USR2 $pid_udhcpc
         kill -USR1 $pid_udhcpc
      fi
   fi
done
```

Figura 3.8: *Shell Script* DOWN.

O fluxograma a seguir, Figura 3.9, demonstra como são tratados os eventos no *script* descrito anteriormente.

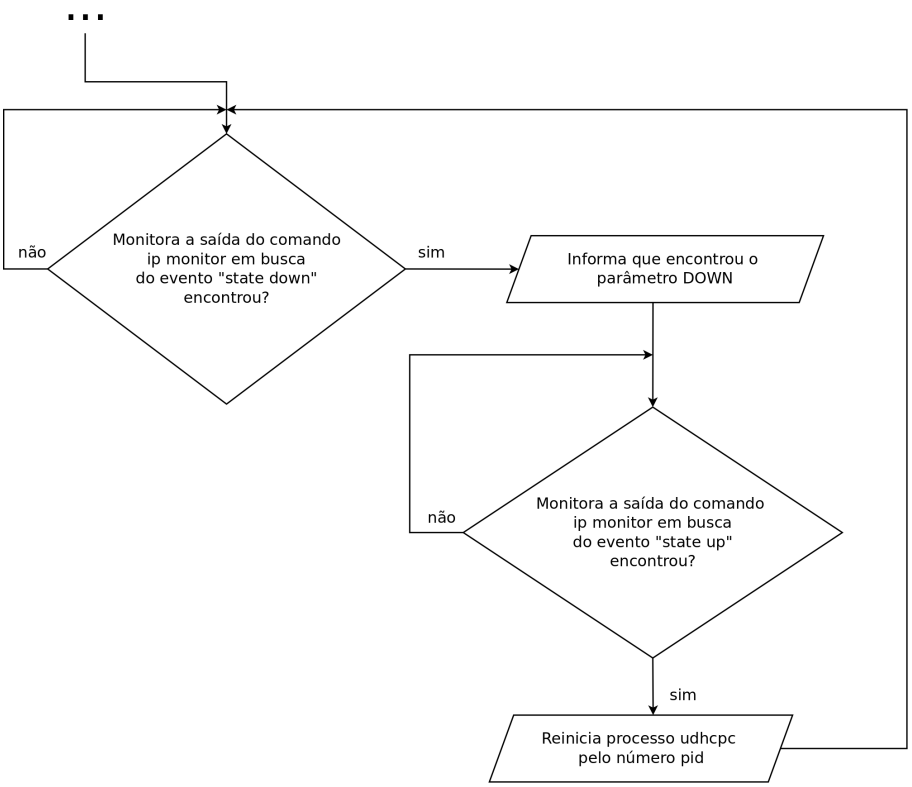

Figura 3.9: Fluxograma do *Shell Script*.

3.3.3 Procedimento do Experimento

Para que não fosse necessária a instalação dos APs em locais diferentes, a mobilidade foi simulada com o desligamento do AP inicialmente associado. Assim, haveria uma imedita perda da rede atualmente conectada, e com o uso do *script* partiria-se para uma nova conexão com a segunda rede disponível.

Após certificar-se de que os APs estavam ativos e em pleno funcionamento, considerando as configurações já descritas, o experimento pode ser iniciado. A sequência a seguir descreve os passos realizados para execução do experimento:

1. O primeiro passo foi executar o *script* principal, *monitoramento_wireless*. Nele as configurações estão preparadas para obter IP primeiramente à rede projeto1. É o que ocorre após associação com a rede e iniciado o uDHCPc;

2. Em seguida já pode ser feita a mobilidade. O AP contendo a rede projeto1 é então desativado. Este evento é sentido pelo auxílio do comando *ip monitor*, através da filtragem do

estado DOWN.

3. A associação com a rede projeto2 é então feita. Na sequência são enviados os sinais ao processo uDHCPc, com a intenção de zerar a configuração da interface e realizar um pedido de endereçamento ao servidor DHCP disponível.

4. Por fim, o terminal móvel adquire o novo endereço do servidor DHCP da rede projeto2.

As amostras de tempo associadas são medidas com a ajuda do *script Tempo UP/ DOWN* e o *tcpdump*.

O *tcpdump* é utilizado para capturar a mensagem DHCP que informa o sucesso na obtenção do endereçamento. A mensagem pode ser visualizada na Figura 3.10.

```
20:14:04.922596 IP 10.1.1.1.67
> 255.255.255.255.68: BOOTP/DHCP,
Reply, length 548
```

Figura 3.10: Mensagem DHCP Capturada com *tcpdump*.

Já o *script* combina a utilização do *ip monitor* com o comando *date*. É executado paralelamente ao experimento para indicar o momento em que a nova rede é associada à inteface. A Figura 3.11 ilustra uma captura da saída emitida pelo *script*.

```
...
20:14:01-839374215
>> Paramentro DOWN encontrado <<
...
...
...
20:14:03-407701617
>> Paramentro UP encontrado <<
...
...
```

Figura 3.11: Saída do *Script* “Tempo UP/ DOWN”.

Com a junção destes, é possível identificar o intervalo de tempo tomado por esta aquisição.

3.3.4 Análise dos Resultados Obtidos

Neste teste a intenção foi medir o tempo T_{aqw} utilizando-se do *script*. Após a versão final estar concretizada foram realizadas tomadas de tempo. Conforme já indicado anteriormente, o início da contagem deu-se com a filtragem do evento DOWN, e terminou ao receber a mensagem de resposta à requisição DHCP.

Os resultados obtidos são mostrados na Tabela 3.3. Nesta mesma Tabela são indicados tempos T_{aq}, obtidos com o uso do script no primeiro teste, com rede cabeada.

Tempo	01	02	03	04	05	06	07	08	09	10	Média
T_{aqw}	1,5148 s	1,5350 s	1,5415 s	1,5814 s	1,5972 s	1,5807 s	1,5900 s	1,5391 s	1,5315 s	1,5453 s	1,5556 s
T_{aq}	0,6179 s	0,6219 s	0,6291 s	0,9843 s	0,5675 s	1,2533 s	0,8819 s	0,5587 s	0,6306 s	0,7854 s	0,7530 s

Tabela 3.3: Tomadas de Tempo com Redes Sem Fio e Cabeadas.

Os tempos permitem o comparativo da capacidade de resposta do uDHCPc com apoio do *ip monitor* e *shell script* auxiliar em obter o endereçamento IP desejado de uma nova rede. Nesta comparação observa-se que a rede sem fio é mais lenta ao processar a concessão de novo endereçamento, resultando em praticamente o dobro de tempo utilizado.

3.4 Conclusão

Este capítulo visou analisar o comportamento do DHCP em um contexto de mobilidade e no âmbito de sistemas embarcados. Foi feita uma descrição da versão reduzida do DHCP para sistemas embarcados, o *micro Dynamic Host Configuration Protocol client* (uDHCPc), detalhando seu funcionamento e estrutura.

Em seguida realizou-se cenários de testes, com a intenção de verificar a capacidade de resposta do uDHCPc. Os cenários de testes foram realizados tanto em redes cabeadas como em redes sem fio. Com eles foi possível a realização de medições de tempo. A partir daí criou-se tabelas, que facilitaram a visualização dos tempos de resposta, deixando mais claro o tratamento do uDHCPc nas diferentes estruturas de rede.

Conclui-se que em redes cabeadas a obtenção do endereçamento IP utiliza um tempo menor. A explicação desse fato é que as redes sem fio possuem a característica de demorar um pouco mais na transmissão de mensagens.

Ao término de cada experimento foi feita uma análise dos resultados obtidos com a mobilidade do uDHCPc, sendo indicados os tempos mensurados durante os testes. Os tempos obtidos inicialmente, sem auxílios externos, não foram satisfatórios. Consequentemente foi feita uma proposta de melhora, para reduzir este tempo de aquisição. Com ajuda de um esquema que faz a captura de informação da camada de enlace e comunica o uDHCPc, essa melhora de tempo foi possível e a redução foi bastante considerável, baixando do valor médio 62,3 segundos, para 1,55 s em redes sem fio, e 753 ms em rede cabeada.

No Capítulo 4 será utilizado o esquema aqui citado, juntamente com o protocolo SIP. O intuito é refazer o registro SIP em um servidor *Asterisk* após uma mobilidade em redes sem fio.

4 A Mobilidade com o SIP

4.1 Introdução

Os mecanismos para o tratamento da mobilidade no SIP não estão previstos nas implementações abertas que foram estudadas neste trabalho. Para a consecução dos objetivos colocados no Cap.1 fez-se necessário a seleção de uma implementação e a modificação da mesma para que suportasse os testes de mobilidade entre sub-redes.

Neste Capítulo será inicialmente apresentada a biblioteca PJSIP escolhida para ser implantada sobre uma plataforma mínima com o Linux. A motivação para que se execute sobre uma plataforma com este perfil é a perspectiva de uma futura migração para uma sistema embarcado móvel que possibilite a realização de novos experimentos e pesquisas.

Na sequência são apresentadas as modificações realizadas sobre uma aplicação PJSIP para suportar a mobilidade de terminal sem sessões em andamento. Uma sequência de testes é então descrita culminando na integração do uDHCPc e *ip monitor*, discutidos no capítulo anterior. Uma análise de desempenho do sistema é mostrada.

Finalmente, são discutidas as características adicionais a serem incorporadas na aplicação para que ela seja capaz de tratar a mobilidade com sessões em andamento.

4.2 A Definição de uma Implementação SIP: o PJSIP

4.2.1 PJSIP: Motivações da Escolha

Conforme colocado, a escolha do *softphone* a ser utilizado nos experimentos foi de suma importância para a sequência do trabalho. O processo da escolha da biblioteca/aplicação envolveu dentre outros a análise das seguintes alternativas: *Ekiga*, *Twinkle*, *X-Lite*, e o PJSIP. A Tabela 4.1 resume as características das mesmas.

Dentre as alternativas o PJSIP foi o que obteve maior destaque. Ele permite o uso do proto-

Softphone	Ekiga	PJSIP	Twinkle	X-Lite
Licença	GPL	GPL	GPL	*Freeware*
Linguagem	C++	C	C++	C++
Usado em	Linux/Windows	Symbian/Vários	Linux	Linux/Windows
Protocolos	SIP/ H.323	SIP	SIP	SIP

Tabela 4.1: Comparação de *Softphones*.

colo SIP em diferentes níveis de abstração, é escrito em linguagem C, voltado a aplicações VoIP embarcadas ou não, possui licença de código aberto, é portável, possui extensa documentação, e além de tudo já é usado em várias implementações, inclusive de empresas renomadas como a Nokia (sobre o Sistema Operacional Symbian). Portanto, a escolha não poderia ser diferente, visto que com ele os rumos traçados tornavam-se possíveis.

4.2.2 Estruturas das APIs e Bibliotecas

A arquitetura do PJSIP é bem subdividida, e sua documentação está disponível na *web* (PRIJONO, 2007). Para parte inicial de seu entendimento a Figura 4.1 é utilizada.

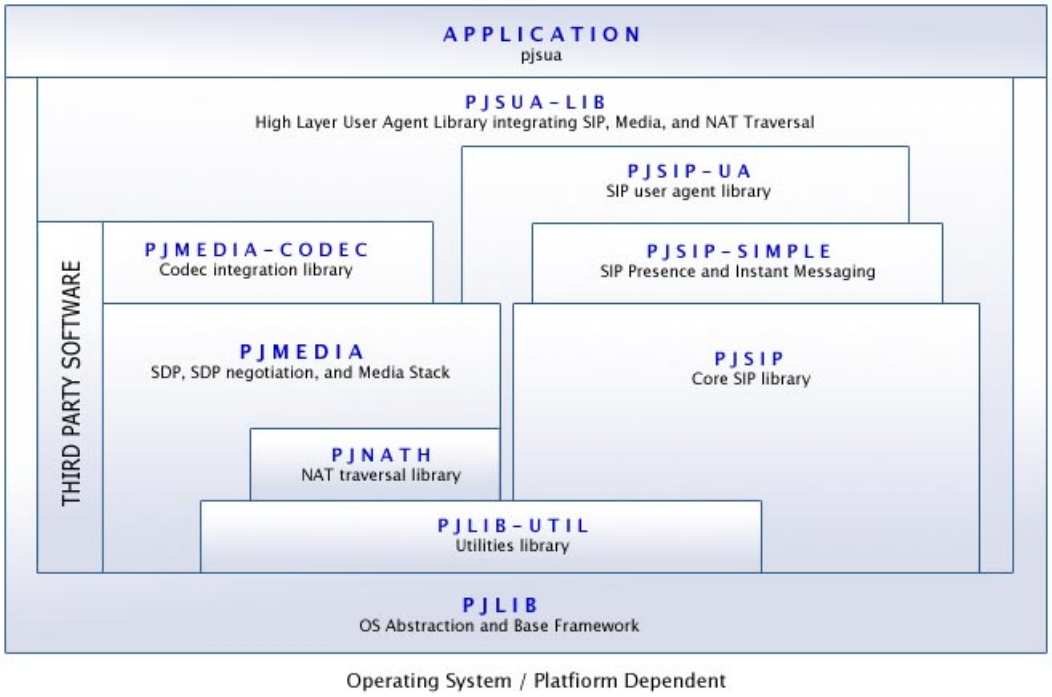

Figura 4.1: Diagrama de Arquitetura PJSUA (PRIJONO, 2007).

Pode ser observado que o PJSIP apresenta diferentes níveis de camadas e de *Application Programming Interface* (API), descritas brevemente a seguir.

- ***PJLIB*** - Pequena biblioteca, posicionada no nível mais baixo da pilha. Ela é apropri-

ada para aplicações embarcadas, criando abstrações para características que normalmente não são portáveis entre SOs. Entre outras, pode-se citar abstrações para manipulação de *threads*, semáforos, *sockets* e para gerenciamento de tempo;

- ***PJLIB-UTIL*** - É uma biblioteca auxiliar a biblioteca PJLIB, fornecendo operações tais como análisadores de XML, resolvedores de DNS , clientes STUN e algoritmos de criptografia;

- ***PJNATH*** - É uma biblioteca de código aberto que fornece funcionalidades para atravessar NATs. Ela depende apenas da PJLIB e PJLIB-UTIL;

- ***PJMEDIA*** - Pilha com funcionalidades de mídia. Apresenta dimensões reduzidas, boa extensibilidade e portabilidade excelente. Contém todos os principais ; componentes de mídia, tais como: algoritmos de manipulação de áudio, protocolos SDP, RTP e RTCP,

- ***PJMEDIA-CODEC*** - É a biblioteca de integração de CODECs. Possui várias estruturas e funções, tais como o gerenciador de CODECs, registrador de um novo CODEC, inicialização de CODEC, entre outras;

- ***PJSIP*** - É a biblioteca responsável pela implementação do SIP propriamente dita. Ela foi projetada para ter alto desempenho, ter tamanho reduzido (própria para sistemas embarcados) e ser flexível;

- ***PJSIP-SIMPLE*** - biblioteca que implementa a presença de eventos SIP e mensagens instantâneas;

- ***PJSIP-UA*** - Esta é a biblioteca que implementa os UAs do SIP. Permite gerenciar sessões INVITE, registro de clientes e transferências de chamadas;

- ***PJSUA-LIB*** - É uma biblioteca de alto nível e que serve para a construção de aplicações multimídia SIP de forma simples. Ela fornece abstrações que "empacotam" as funcionalidades de sinalização e de mídia, fornece gerenciamento de contas, mensagens instantâneas, além de recursos multimídia como videoconferência, transmissão de arquivos, e outros;

- ***APPLICATION*** - São modelos de aplicações que podem ser usados como referência para desenvolvimento.

  Note que dependendo das necessidades do sistema em desenvolvimento, é possível desenvolver aplicações diretamente sobre as camadas PJMEDIA e PJSIP.

4.2.3 Exemplo de Aplicação: A Aplicação *simple_pjsua*

Juntamente com a biblioteca acompanham vários exemplos de aplicação. Uma delas é a *simple_pjsua*, mostrada em anexo (Anexo B). Ela é escrita em linguagem C, e possui menos de 200 linhas de código. Mesmo sendo curta, é bastante completa. Contém opções de registro, de autenticação, negociação SDP, e meios de comunicação inteiramente caracterizados.

Examinando o código pode-se observar a seguinte sequência de operações:

1. Inicialmente é chamada a função *pjsua_create*, que ao ser executada deve criar um agente usuário (UA). Neste processo, entre outras coisas, é inicializada a biblioteca PJLIB, fundamental antes de qualquer função PJLIB ser executada.

2. Após ser criada, a aplicação inicializa o PJSUA através da função *pjsua_init*. Esta possui várias configurações, que podem ser feitas pela aplicação.

3. Na sequência, a aplicação precisa realizar tarefas como: criar transportes SIP com "pjsua_transport_create", criar contas SIP com "pjsua_acc_add" ou "pjsua_acc_add_local", e opcionalmente configurar o dispositivo de som, configurações de CODEC, e outras mídias.

4. Para dar início ao PJSUA é utilizada a função *pjsua_start*. Ela verifica se as configurações foram realizadas corretamente e aplica valores padrão quando não foram passadas. Várias modificações podem ser feitas durante a execução, como adicionar, alterar ou excluir contas.

   Todo o processo de criação, inicialização e outras funções auxiliares são controlados pela API PJSUA.

5. Finalmente o código entra em um loop infinito esperando a ocorrência de eventos aos quais foram associadas as funções de *callback*, para finalizar todas as chamadas ativas no momento, ou o término do programa, pressionando as teclas 'h' ou 'q' respectivamente.

Note que o nível de API que está sendo usado é o mais alto (PJSIP-LIB) e o código se utiliza do conceito de funções *callback*. Estas funções são cadastradas para serem chamadas na ocorrência de um evento, sendo usadas em paradigmas de programação dirigidos a eventos.

A aplicação *simple_pjsua* foi utilizada como base para os testes de mobilidade que serão descritos. Para um teste inicial de registro com um servidor *Asterisk* o código sofreu algumas alterações para adaptar-se ao cenário (ver Figura 4.2). As linhas 3 e 4 sofreram mudanças

nos parâmetros passados em SIP_DOMAIN e SIP_USER, referentes ao número IP do servidor *Asterisk* e nome da conta de usuário, respectivamente. A linha 10 mostra o código "cfg.port" com o número da porta utilizada para o correto funcionamento da aplicação, 5061. Na linha 15 é utilizada a opção "*" como parâmetro para a função "pj_str", para permitir que a aplicação receba um valor diferente do passado (SIP_DOMAIN) para o parâmetro realm (KANGKUNG, 2007).

```
1  ...
2
3  #define SIP_DOMAIN "10.1.1.101"
4  #define SIP_USER "6001"
5
6  ...
7
8      /* Add UDP transport. */
9
10    cfg.port = 5061;
11 ...
12
13     /* Register to SIP server by creating SIP account. */
14
15    cfg.cred_info[0].realm = pj_str("*");
16 ...
17     }
```

Figura 4.2: Linhas Modificadas no *simple_pjsua*

4.3 O PJSIP e a Mobilidade

No código da aplicação *simple_pjsua* foram feitas alterações buscando um registro dinâmico que permitisse de alguma maneira tratar um novo registro após troca de redes. As modificações foram realizadas no sentido de associar um sinal (SIGUSR1) a uma função (tratador) de forma que um agente externo, por exemplo, o uDHCPc, pudesse informar à aplicação SIP de que uma nova rede foi acessada e que uma atualização de registro deve ser encaminhada para o servidor de registros SIP. Deve ser observado que o PJSIP não realiza esta detecção de forma imediata.

A Figura 4.3 ilustra as linhas de comando adicionadas ao arquivo principal (simple_pjsua). A inserção das linhas de código não alteraram o funcionamento da aplicação. Note que a referida função permite realizar um novo registro junto ao servidor de registros.

A associação de uma função (*handler*) com um sinal é realizada pela função signal (linha 15). O sinal SIGUSR1 é associado a função *USR1handler*. Na ocorrência do sinal, o fluxo de execução do programa é desviado para a função, tal como mostra a Figura 4.3 por meio das

setas. O fluxo principal é interrompido no ponto em que estiver, mesmo que aguardando por um evento (primitiva do SO).

```
 1
 2 ...
 3 jmp_buf  JumpBuffer;
 4
 5 void  USR1handler(int  sig)
 6 {
 7          signal(SIGUSR1, USR1handler);
 8          siglongjmp(JumpBuffer, 1);
 9 }
10
11
12 ...
13 /* main() */
14        :
15        signal(SIGUSR1, USR1handler);
16        :
17
18 /* Wait until user press "q" to quit. */
19        for (;;) {
20                /* loop principal */
21
22        }
23
24        if (sigsetjmp(JumpBuffer, 1) == 1) {
25                pjsua_acc_set_registration(acc_id,PJ_TRUE);
26        }
27
```

Momento de recebimento do sinal USR1

Figura 4.3: Esquema Recepção do Sinal.

A função *USR1handler* reassocia o sinal USR1 a ela mesma, garantindo que um novo sinal seja tratado novamente por ela (linha 7). Ne sequência, ela invoca a função *siglongjmp*. Esta função da biblioteca permite a realização de um salto para um ponto do programa determinado por *sigsetjmp*. Esta última função salva um contexto necessário a realização deste salto. Neste ponto é executada a função *pjsua_acc_set_registration* que realiza um novo registro com o servidor de registros. O fluxo principal é retomado normalmente, dentro do *loop* infinito.

4.4 O Experimento de Mobilidade com o SIP

4.4.1 Objetivos do Experimento

O objetivo deste experimento é avaliar a capacidade da aplicação simple_pjsip, modificada segundo a Seção 4.3, de se registar no servidor de registros SIP após uma mudança de sub-rede. As mensagens de registro SIP serão analisadas e o tempo T_{reg} será medido. Este tempo é

definido por:

$$T_{reg} = T_{E_{ok}} - T_{E_{up}} \tag{4.1}$$

A medição de tempo é feita tendo como início o reconhecimento de uma nova rede (evento E_{up}), por parte do cliente e tendo como fim o momento em que o cliente recebeu a mensagem 200 OK do servidor *Asterisk* (evento E_{ok}), confirmando o registro da conta SIP.

4.4.2 Descrição do Cenário

O cenário é similar ao da Seção 3.3. Nos testes realizados foram utilizados dois APs e dois microcomputadores: um com o papel de servidor (rodando *Asterisk*), e o outro como cliente (sofrendo a mobilidade). A Figura 4.4 ilustra o cenário.

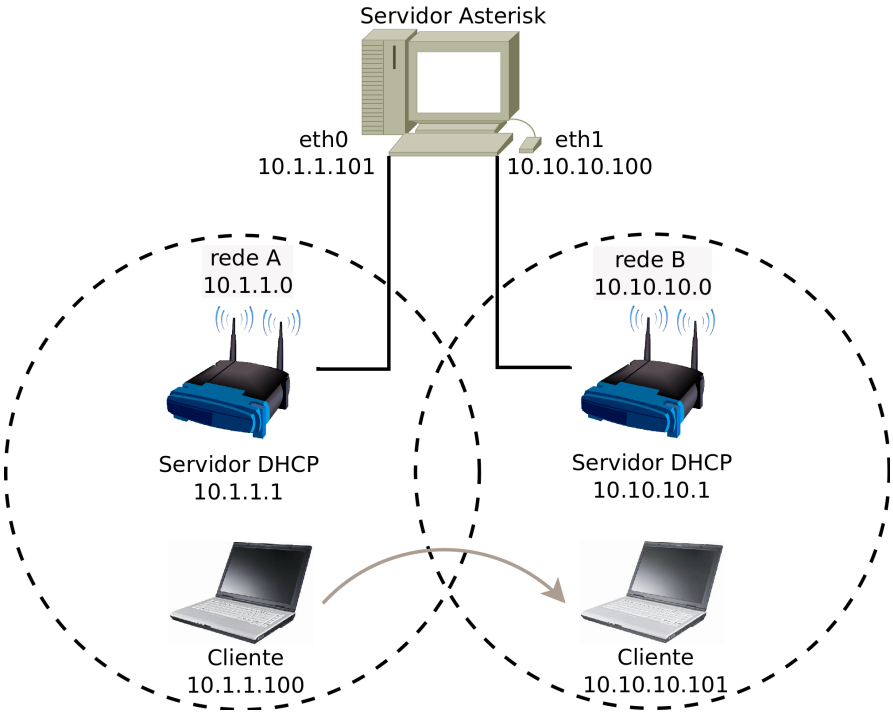

Figura 4.4: Cenário de Testes Wireless.

Os APs estavam no mesmo ambiente, porém trabalhando em canais diferentes: o AP1 no canal 1 e o AP2 no canal 2. Eles possuiam configuração para trabalhar como servidores DHCP, fornecendo a faixa de IP ".100" até ".105". Outra configuração necessária nos APs foi acrescentar na tabela de roteamento estático o endereço da rede "vizinha", até então desconhecida.

A simulação da mobilidade do cliente foi efetuada através do desligamento da transmissão sem fio através da página de configuração de determinado AP, que podia ser acessada através do micro servidor, pois estava conectado via *ethernet*.

Os microcomputadores possuiam características diferentes. O servidor foi equipado com

duas placas de rede, assim cada uma das interfaces possuia um endereço de IP ativo pertencente à rede de determinado AP (10.1.1.101 e 10.10.10.100). Já o computador utilizado como cliente, possuia uma configuração física simples sendo utilizada somente a interface para redes sem fio.

Configuração do *Asterisk*

O *Asterisk* foi configurado mantendo seus arquivos *default* praticamente intactos. Porém, deve dar-se ênfase à algumas modificações necessárias para o pleno funcionamento e execução do experimento. No caso do arquivo *sip.conf* (que possui as contas SIP), foi feita inserção de linhas de comando para inserir uma conta SIP, necessária para os testes. A conta utilizada foi configurada com o parâmetro "insecure=very" (detalhes do código na Figura 4.5) para que o servidor *Asterisk* ao receber uma requisição de registro, aceitasse respondendo-a diretamente, sem realizar o processo de autorização.

```
...

[6001]
type=friend
context=projeto
username=6001
host=dynamic
insecure=very
```

Figura 4.5: Código de Configuração de Conta SIP

Outro arquivo modificado foi o *extensions.conf*, nele foram inseridas linhas de código para um mínimo funcionamento. Na Figura 4.6 a seguir, pode ser visualizado o código em questão, ele define a qual contexto pertence determinada conta de usuário SIP e o que deve ser feito quando ela é evocada.

```
...

[projeto]
exten=>6001,1,Dial(SIP/6001,30)
exten=>6001,n,Hangup
```

Figura 4.6: Código de configuração de contexto

O Software no Terminal: A Integração *ip monitor*, uDHCPc e PJSIP

A viabilidade de junção das aplicações tornou-se mais clara após entender como se dá o funcionamento de cada uma delas. O fato de a aplicação uDHCPc aceitar o recebimento de

sinais ajudou e muito no desempenho e sequência do trabalho.

Como mostrado na Figura 3.8 do capítulo anterior, o *script* envia os sinais USR2 e USR1 para o processo uDHCPc, com a finalidade de forçar a aplicação a limpar as configurações de rede, realizando em seguida a tentativa de uma nova configuração.

Na Figura 4.7 são ilustrados os vários componentes que integram o software no terminal, bem como a sequência das etapas percorridas até chegar-se a um novo registro junto ao servidor *Asterisk*. São destacados os principais eventos executados desde a percepção de nova rede até pedido efetivo de registro.

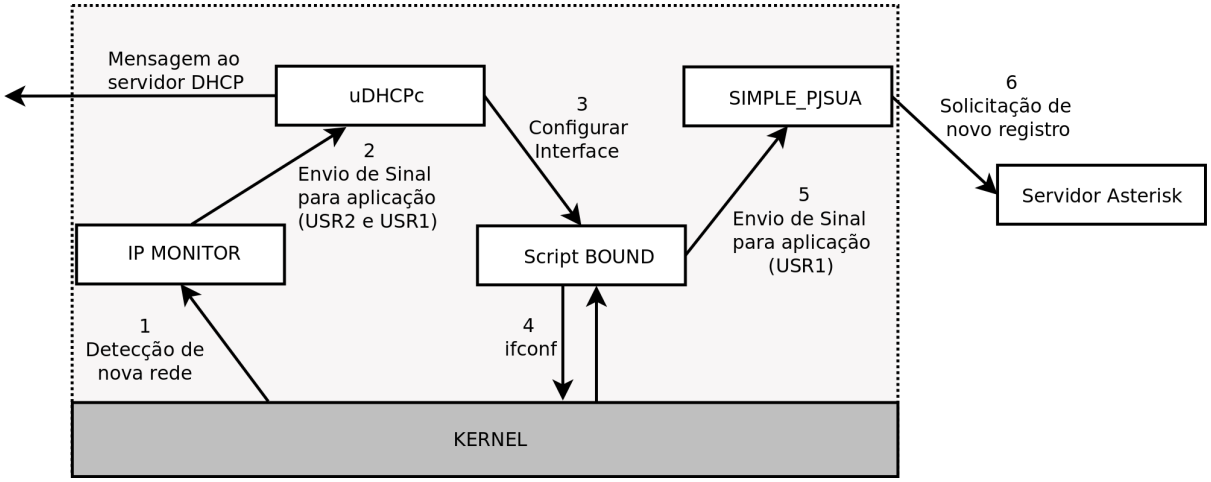

Figura 4.7: Etapas para um Novo Registro.

O primeiro evento é a detecção de nova rede. Ela é sentida através da ferramenta *ip monitor*, que busca esta informação diretamente no *kernel*. Após esta detecção são realizados os envios de sinais para a aplicação uDHCPc. O sinal USR2 com a função de zerar as informações da rede, e o USR1 de realizar a concessão de um novo endereçamento.

Recebendo esses sinais, o uDHCPc realiza a nova configuração da interface (3) com auxílio do script BOUND. Em seguida, repassa as informações ao *kernel* (4). No fim da execução do BOUND foi inserida linha que faz o envio do sinal "USR1" à aplicação PJSIP. Estando a aplicação preparada para receber tal sinal e realizar o pedido de novo registro ao servidor *Asterisk*, encerra-se o ciclo de re-registro.

4.4.3 Procedimento do Experimento

Alguns procedimentos, indicados a seguir, foram seguidos para correta execução deste experimento:

1. De início foi executado o *script* principal, *monitoramento_wireless* (Anexo C). Obtendo, primeiramente associação à rede projeto1 e aquisição de endereçamento IP;

2. Com endereço válido na rede foi realizado o primeiro registro junto ao servidor *Asterisk*;

3. Na sequência foi feito o processo de mobilidade. Desativando o AP da rede A. Após o estado DOWN ser filtrado, com *awk* e *grep* junto ao *ip monitor*, partiu-se para a associação com a nova rede, a rede B.

4. Após associação, foram enviados através do script, os sinais ao processo uDHCPc, para reinicialização deste serviço. Assim que confirmada a configuração da interface, já foi enviado sinal à aplicação *simple_pjsua* com intenção de realizar um novo pedido de registro junto ao *Asterisk*. Este sinal foi incluido no fim do *script* de configuração de endereçamento de rede (*default.bound*).

5. Por fim, a associação com a rede B foi então feita, e o pedido de novo registro realizado. Com isso era esperado somente o recebimento de confirmação, que veio com a mensagem 200 OK.

4.4.4 Análise de Mensagens SIP

As mensagens trocadas entre cliente e servidor foram capturadas com a ajuda do aplicativo Wireshark. Com elas, além de obter medidas de tempo, puderam ser feitas análises mais profundas, sendo possível apontar os problemas detectados durante os estudos, e encontrar soluções, possibilitando prosseguir.

No início dos testes foi constatado que o cliente ao enviar uma requisição de registro ao servidor (REGISTER), recebia como resposta a mensagem Unauthorized, contendo campos para tratar de autenticação, Figura 4.8. Com essa necessidade de autenticação não era possível realizar o novo registro desejado, pois o servidor esperava a volta dos parâmetros do cliente, e isto não acontecia.

Para contornar esse problema foi utilizada a opção "insecure=very" na configuração de contas do *Asterisk*, como já destacado no item 4.4.2. Assim, quando solicitado a realizar um registro não aguardaria mensagens de autenticação. Após esta configuração o servidor passou a responder diretamente com a mensagem 200 OK.

A Figura 4.9 mostra detalhes da mensagem SIP encaminhada do cliente ao servidor, ela representa a primeira requisição realizada, o REGISTER.

```
<--- Transmitting (no NAT) to 10.10.10.10:5061 --->
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 10.10.10.10:5061;
     branch=z9hG4bKPjw0rtoSYertzTfVyzcbDSjDboU9m-73CN;
     received=10.10.10.10;
     rport=5061
From: <sip:6001@172.18.22.149>;tag=.ddVx6xJLpBQKsH1uTMJKPdfqnFJ9lB4
To: <sip:6001@172.18.22.149>;tag=as400803b7
Call-ID: YDpOe1LbiT--59P09oR4UNjNg4HffVUx
CSeq: 10555 REGISTER
User-Agent: Asterisk PBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
Supported: replaces
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="5bc33666"
Content-Length: 0
```

Figura 4.8: Mensagem SIP: Unauthorized.

No campo Request-Line pode ser visto o tipo de requisição, neste exemplo é feita solicitação de registro (REGISTER). Na sequência é listado a quem está sendo endereçado o pacote (sip: 10.1.1.101); ainda na mesma linha o protocolo e versão utilizados (SIP/2.0) são informados.

```
▽ Session Initiation Protocol
  ▽ Request-Line: REGISTER sip:10.1.1.101 SIP/2.0
      Method: REGISTER
    ▽ Request-URI: sip:10.1.1.101
        Request-URI Host Part: 10.1.1.101
      [Resent Packet: False]
  ▽ Message Header
    ▽ Via: SIP/2.0/UDP 10.1.1.100:5061;rport;branch=z9hG4bKPj0pAWM0WEZv3AJnH5QP9dqMjfIF3CZYH2
        Transport: UDP
        Sent-by Address: 10.1.1.100
        Sent-by port: 5061
        RPort: rport
        Branch: z9hG4bKPj0pAWM0WEZv3AJnH5QP9dqMjfIF3CZYH2
      Max-Forwards: 70
    ▽ From: <sip:6001@10.1.1.101>;tag=ssGf78S7zNyhjQHl7V6MFBjRaMLmF-Wd
      ▷ SIP from address: sip:6001@10.1.1.101
        SIP tag: ssGf78S7zNyhjQHl7V6MFBjRaMLmF-Wd
    ▷ To: <sip:6001@10.1.1.101>
      Call-ID: v6CulLeoZu37TP3yxMoBuAQzXxkVTUNv
    ▷ CSeq: 19951 REGISTER
    ▷ Contact: <sip:6001@10.1.1.100:5061>
      Expires: 300
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS
      Content-Length:  0
```

Figura 4.9: Mensagem SIP: Request - REGISTER.

Na Figura 4.10 são apresentados os campos da mensagem 200 OK. Esta mensagem possui um campo que indica o status (Status-Line), descrito em código de status (Status-Code), e o tempo de resposta da mensagem (Response Time). Os campos do cabeçalho da mensagem possuem as mesmas funções dos descritos anteriormente na mensagem de requisição.

```
▽ Session Initiation Protocol
  ▽ Status-Line: SIP/2.0 200 OK
      Status-Code: 200
      [Resent Packet: False]
      [Request Frame: 2]
      [Response Time (ms): 2]
  ▽ Message Header
    ▽ Via: SIP/2.0/UDP 10.1.1.100:5061;branch=z9hG4bKPj0pAWM0WEZv3AJnH5QP9dqMjfIF3CZYH2;received=10.1.1.100;rport=5061
        Transport: UDP
        Sent-by Address: 10.1.1.100
        Sent-by port: 5061
        Branch: z9hG4bKPj0pAWM0WEZv3AJnH5QP9dqMjfIF3CZYH2
        Received: 10.1.1.100
        RPort: 5061
    ▽ From: <sip:6001@10.1.1.101>;tag=ssGf78S7zNyhjQHl7V6MFBjRaMLmF-Wd
      ▷ SIP from address: sip:6001@10.1.1.101
        SIP tag: ssGf78S7zNyhjQHl7V6MFBjRaMLmF-Wd
    ▷ To: <sip:6001@10.1.1.101>;tag=as2bd3284e
      Call-ID: v6CulLeoZu37TP3yxMoBuAQzXxkVTUNv
    ▷ CSeq: 19951 REGISTER
      Server: Asterisk PBX 1.6.2.0~rc2-0ubuntu1.2
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO
      Supported: replaces, timer
      Expires: 300
    ▷ Contact: sip:6001@10.1.1.100:5061;expires=300
      Date: Fri, 26 Nov 2010 22:30:32 GMT
      Content-Length: 0
```

Figura 4.10: Mensagem SIP: 200 OK.

Outro problema encontrado foi que o encaminhamento das mensagens de retorno do servidor estavam sendo endereçadas ao IP antigo do cliente. Isto foi percebido através do campo Via da mensagem SIP. Para este problema já existia um estudo realizado pelos próprios criadores do protocolo SIP, tratando do *Network Address Translation* (NAT). Após essa descoberta bastava ativar uma linha de comando no arquivo de configuração de contas SIP.

A opção NAT foi então ativada no arquivo, descomentando a linha “nat= yes”. Assim, o servidor passa a considerar o uso do NAT, e o IP usado pelo servidor para envio de respostas passa a não ser mais o endereço contido no campo Via. Com esse procedimento os pacotes passaram a ser enviados ao destino de rede correto.

A Figura 4.11 representa a mensagem de requisição de registro enviada após ocorrer a mobilidade. Com ela pode ser observado o campo VIA.

4.4.5 Análise dos Resultados Obtidos

Após o término das modificações e análise das trocas de mensagens foram realizadas medidas de tempo para mensurar o desempenho obtido nesta etapa.

O *script* Tempo UP/ DOWN, que captura os eventos DOWN e UP, foi novamente utilizado. A Figura 4.12 mostra esses eventos, bem como o horário de captura.

```
▽ Session Initiation Protocol
  ▽ Request-Line: REGISTER sip:10.1.1.101 SIP/2.0
      Method: REGISTER
    ▽ Request-URI: sip:10.1.1.101
        Request-URI Host Part: 10.1.1.101
      [Resent Packet: False]
  ▽ Message Header
    ▽ Via: SIP/2.0/UDP 10.10.10.101:5061;rport;branch=z9hG4bKPj67g0sjhaEGtButldh.vksrvJSrXEiusB
        Transport: UDP
        Sent-by Address: 10.10.10.101
        Sent-by port: 5061
        RPort: rport
        Branch: z9hG4bKPj67g0sjhaEGtButldh.vksrvJSrXEiusB
      Max-Forwards: 70
    ▽ From: <sip:6001@10.1.1.101>;tag=EUgaLZSMhMZ9IPjswabDM7KrtOf-wUG8
      ▷ SIP from address: sip:6001@10.1.1.101
        SIP tag: EUgaLZSMhMZ9IPjswabDM7KrtOf-wUG8
    ▷ To: <sip:6001@10.1.1.101>
      Call-ID: v6CulLeoZu37TP3yxMoBuAQzXxkVTUNv
    ▷ CSeq: 19953 REGISTER
    ▷ Contact: <sip:6001@10.10.10.101:5061;transport=UDP>
    ▷ Contact: <sip:6001@10.1.1.100:5061>;expires=0
      Expires: 300
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS
      Content-Length:  0
```

Figura 4.11: Mensagem SIP: reRequest - reREGISTER.

```
...
18:04:02-796746446
>> Paramentro DOWN encontrado <<
...
...
...
18:04:04-567137508
>> Paramentro UP encontrado <<
...
...
```

Figura 4.12: Saída do *Script* “Tempo UP/DOWN”.

A Figura 4.13 ilustra a mensagem SIP enviada pelo servidor, com destino ao cliente. Foi capturada no *log* de saída da aplicação *simple_pjsua*. Ela tem a função de confirmar o registro, e ainda neste experimento serviu para finalizar a contagem de tempo.

Os resultados de tempos obtidos são mostrados na Tabela 4.2. Nela são indicados os tempo obtidos em dez tomadas de tempo de T_{reg}, além da média desses tempos.

Tempo	01	02	03	04	05	06	07	08	09	10	Média
T_{reg}	1,735 s	1,814 s	1,743 s	1,755 s	1,794 s	1,745 s	1,764 s	1,801 s	1,751 s	1,740 s	1,764 s

Tabela 4.2: Tomadas de Tempo Novo Registro.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.10.10.101:5061;branch=z9hG4bKPjv2QFF4gmtMP2Prk4WRruZnLPhhRK4xRh;received=10.10.10.101;rport=5061
From: <sip:6001@10.1.1.101>;tag=hXRO9bRlfgyGipScf3jh5rAGfR3tb9Lg
To: <sip:6001@10.1.1.101>;tag=as0fec357a
Call-ID: R1D4MyS7yxiYM6ncO4JYkkyxD3Ks3R9F
CSeq: 41808 REGISTER
Server: Asterisk PBX 1.6.2.0~rc2-0ubuntu1.2
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO
Supported: replaces, timer
Expires: 300
Contact: sip:6001@10.10.10.101:5061;transport=UDP;expires=300
Date: Sat, 18 Dec 2010 20:04:24 GMT
Content-Length: 0
--end msg--
18:04:06.302    pjsua_acc.c  sip:6001@10.1.1.101: registration success, status=200 (OK), will re-register in 300 seconds
18:04:06.302    pjsua_acc.c  Keep-alive timer started for acc 0, destination:10.10.10.102:5060, interval:15s
```

Figura 4.13: Mensagem 200 OK - Confirmação de Registro.

4.5 Mobilidade de Sessões em Andamento: Discussão

O tratamento da mobilidade apresentando anteriormente não contempla sessões em andamento. Neste caso, além da atualização do registro com o servidor de registros, é necessário sinalizar o terminal (ou terminais) comunicante de que o fluxo de mídia deve ser encaminhado para um novo endereço. Neste sentido, além da chamada da função *pjsua_acc_set_registration*, a aplicação no terminal que realizou o movimento deve enviar um novo INVITE (reINVITE). A função *pjsua_acc_add* pode ser utilizada para esta operação. Entretanto, a aplicação deve estar com os novos endereços devidamente atualizados para que o protocolo SDP, usado no reINVITE, transporte corretamente o novo endereço de mídia. Este comportamento pode não estar sendo implementado pelo pjsip.

Do lado do terminal que recebe o reINVITE, a aplicação deve reconfigurar o protocolo RTP para que este redirecione o fluxo de mídia para o endereço fornecido no SDP. Esta reconfiguração deve ser realizada na função callback. A realização correta desta operação também deve ser confirmada.

4.6 Conclusão do Experimento

Neste capítulo foi inicialmente feita a escolha da implementação SIP a ser utilizada, apresentando a estrutura de suas bibliotecas e APIs. Foi também descrito um exemplo de aplicação, a *simple_pjsua*, que foi amplamente estudada e sofreu modificações para suportar a mobilidade de terminal.

Realizou-se análise da troca de mensagens entre cliente e servidor, dando maior ênfase a algumas delas, como INVITE, Unauthorized e 200 OK.

Por fim, foi feita uma análise de desempenho com os tempos obtidos no experimento *wire-*

less, e em seguida discutida a mobilidade de sessões em andamento, apontando os passos a serem executados pelas aplicações.

Tendo conhecimento do tempo gasto para obtenção de endereçamento IP pelo uDHCPc (obtidos no Capítulo 3), considera-se os valores alcançados neste experimento como dentro da normalidade. Visto que o tempo T_{reg} aqui contabilizado, praticamente somou o tempo gasto para realização de novo registro ao de configuração de interface T_{aqw}.

Além do tempo obtido deve dar-se ênfase neste capítulo para a realização do novo registro junto ao servidor SIP. Este processo tomou grande tempo deste estudo, pois envolveu vários eventos, análises e discussões. Ao fim, o sucesso do registro foi concretizado. Utilizando de configurações do NAT, inicialmente não utilizado, e desativando as solicitações de autenticação pelo servidor.

5 *Conclusões*

Os objetivos iniciais colocados para o projeto foram alcançados. Uma plataforma de testes para a realização de experimentos de mobilidade entre sub-redes, com o SIP, no contexto de sistemas embarcados foi definida. O uDHCPc e o PJSIP foram selecionados como base para esta plataforma. Em adição, conseguiu-se com êxito realizar um re-registro após uma mobilidade em redes sem fio através da modificação de uma aplicação PJSIP e uma integração via *scripts* de monitoramento da camada de enlace com o uDHCPc. O desempenho observado foi razoável embora acredita-se que possa ser melhorado, principalmente tratando-se de mobilidade terminal.

Da parte pessoal, pode-se citar conquistas na área de conhecimento do sistema Linux, de aplicações SIP, estruturas de programação em geral, além de conceitos de rede cabeada e sem fio. Vários conhecimentos adquiridos ao longo dos semestres do curso de tecnologia em Sistemas de Telecomunicações foram articulados e consolidados durante o desenvolvimento do Trabalho de Conclusão de Curso.

Muito trabalho ainda resta a ser realizado nesta área. Entre outras perspectivas pode ser colocado:

- A melhoria do desempenho do DHCP pode ainda ser aprofundada. O estabelecimento de endereçamento IP antes mesmo do ACK final do servidor DHCP pode ser investigado;
- A mobilidade de terminais com sessões em andamento deve ser implementada e testada em diferentes condições;
- A configuração do servidor PROXY SIP a partir do DHCP pode ser estudada e implementada. A mobilidade entre domínios com diferentes servidores PROXY SIP pode então ser investigada juntamente com a possibilidade de registro hierárquico.

ANEXO A – Tempo UP/ DOWN

```
ip monitor | while read linha
do

   # Filtra o 'DOWN' jogando na variavel var_down
   var_down=$(echo $linha | awk '$8 ~ "state" {print $9}' | grep 'DOWN')

   # Indica que ocorreu o evento DOWN
   if [ "$var_down" = 'DOWN' ]; then
      echo "..."
      echo `date +%T-%N`
      echo ">> Paramentro DOWN encontrado <<"
      echo "..."
      echo "..."
   fi

   # Filtra o 'UP' jogando na variavel var_up
   var_up=$(echo $linha | awk '$8 ~ "state" {print $9}' | grep 'UP')

   # Indica que ocorreu o evento UP
   if [ "$var_up" = 'UP' ]; then
      echo "..."
      echo `date +%T-%N`
      echo ">> Paramentro UP encontrado <<"
      echo "..."
      echo "..."
   fi

done
exit 0
```

Listagem A.1: Tempo UP/ DOWN

ANEXO B – Código original simple_pjsua

```
/* $Id: simple_pjsua.c 2408 2009-01-01 22:08:21Z bennylp $ */
/*
 * Copyright (C) 2008-2009 Teluu Inc. (http://www.teluu.com)
 * Copyright (C) 2003-2008 Benny Prijono <benny@prijono.org>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 */

/**
 * simple_pjsua.c
 *
 * This is a very simple but fully featured SIP user agent, with the
 * following capabilities:
 *  - SIP registration
 *  - Making and receiving call
 *  - Audio/media to sound device.
 *
 * Usage:
 *  - To make outgoing call, start simple_pjsua with the URL of remote
 *    destination to contact.
 *    E.g.:
```

```
34  *  simpleua sip:user@remote
35  *
36  *  - Incoming calls will automatically be answered with 200.
37  *
38  * This program will quit once it has completed a single call.
39  */
40
41 #include <pjsua-lib/pjsua.h>
42
43 #define THIS_FILE "APP"
44
45 #define SIP_DOMAIN "example.com"
46 #define SIP_USER "alice"
47 #define SIP_PASSWD "secret"
48
49
50 /* Callback called by the library upon receiving incoming call */
51 static void on_incoming_call(pjsua_acc_id acc_id, pjsua_call_id call_id,
52             pjsip_rx_data *rdata)
53 {
54     pjsua_call_info ci;
55
56     PJ_UNUSED_ARG(acc_id);
57     PJ_UNUSED_ARG(rdata);
58
59     pjsua_call_get_info(call_id, &ci);
60
61     PJ_LOG(3,(THIS_FILE, "Incoming call from %.*s!!",
62          (int)ci.remote_info.slen,
63          ci.remote_info.ptr));
64
65     /* Automatically answer incoming calls with 200/OK */
66     pjsua_call_answer(call_id, 200, NULL, NULL);
67 }
68
69 /* Callback called by the library when call's state has changed */
70 static void on_call_state(pjsua_call_id call_id, pjsip_event *e)
71 {
72     pjsua_call_info ci;
73
74     PJ_UNUSED_ARG(e);
75
76     pjsua_call_get_info(call_id, &ci);
```

```
77      PJ_LOG(3,(THIS_FILE, "Call %d state=%.*s", call_id,
78            (int)ci.state_text.slen,
79            ci.state_text.ptr));
80  }
81
82  /* Callback called by the library when call's media state has changed */
83  static void on_call_media_state(pjsua_call_id call_id)
84  {
85      pjsua_call_info ci;
86
87      pjsua_call_get_info(call_id, &ci);
88
89      if (ci.media_status == PJSUA_CALL_MEDIA_ACTIVE) {
90     // When media is active, connect call to sound device.
91     pjsua_conf_connect(ci.conf_slot, 0);
92     pjsua_conf_connect(0, ci.conf_slot);
93      }
94  }
95
96  /* Display error and exit application */
97  static void error_exit(const char *title, pj_status_t status)
98  {
99      pjsua_perror(THIS_FILE, title, status);
100     pjsua_destroy();
101     exit(1);
102 }
103
104 /*
105  * main()
106  *
107  * argv[1] may contain URL to call.
108  */
109 int main(int argc, char *argv[])
110 {
111     pjsua_acc_id acc_id;
112     pj_status_t status;
113
114     /* Create pjsua first! */
115     status = pjsua_create();
116     if (status != PJ_SUCCESS) error_exit("Error in pjsua_create()", status);
117
118     /* If argument is specified, it's got to be a valid SIP URL */
119     if (argc > 1) {
```

```
120     status = pjsua_verify_sip_url(argv[1]);
121     if (status != PJ_SUCCESS) error_exit("Invalid URL in argv", status);
122      }
123
124      /* Init pjsua */
125      {
126     pjsua_config cfg;
127     pjsua_logging_config log_cfg;
128
129     pjsua_config_default(&cfg);
130     cfg.cb.on_incoming_call = &on_incoming_call;
131     cfg.cb.on_call_media_state = &on_call_media_state;
132     cfg.cb.on_call_state = &on_call_state;
133
134     pjsua_logging_config_default(&log_cfg);
135     log_cfg.console_level = 4;
136
137     status = pjsua_init(&cfg, &log_cfg, NULL);
138     if (status != PJ_SUCCESS) error_exit("Error in pjsua_init()", status);
139      }
140
141      /* Add UDP transport. */
142      {
143     pjsua_transport_config cfg;
144
145     pjsua_transport_config_default(&cfg);
146     cfg.port = 5060;
147     status = pjsua_transport_create(PJSIP_TRANSPORT_UDP, &cfg, NULL);
148     if (status != PJ_SUCCESS) error_exit("Error creating transport", status);
149      }
150
151      /* Initialization is done, now start pjsua */
152      status = pjsua_start();
153      if (status != PJ_SUCCESS) error_exit("Error starting pjsua", status);
154
155      /* Register to SIP server by creating SIP account. */
156      {
157     pjsua_acc_config cfg;
158
159     pjsua_acc_config_default(&cfg);
160     cfg.id = pj_str("sip:" SIP_USER "@" SIP_DOMAIN);
161     cfg.reg_uri = pj_str("sip:" SIP_DOMAIN);
162     cfg.cred_count = 1;
```

```
163     cfg.cred_info[0].realm = pj_str(SIP_DOMAIN);
164     cfg.cred_info[0].scheme = pj_str("digest");
165     cfg.cred_info[0].username = pj_str(SIP_USER);
166     cfg.cred_info[0].data_type = PJSIP_CRED_DATA_PLAIN_PASSWD;
167     cfg.cred_info[0].data = pj_str(SIP_PASSWD);
168
169     status = pjsua_acc_add(&cfg, PJ_TRUE, &acc_id);
170     if (status != PJ_SUCCESS) error_exit("Error adding account", status);
171      }
172
173      /* If URL is specified, make call to the URL. */
174      if (argc > 1) {
175     pj_str_t uri = pj_str(argv[1]);
176     status = pjsua_call_make_call(acc_id, &uri, 0, NULL, NULL, NULL);
177     if (status != PJ_SUCCESS) error_exit("Error making call", status);
178      }
179
180      /* Wait until user press "q" to quit. */
181      for (;;) {
182     char option[10];
183
184     puts("Press 'h' to hangup all calls, 'q' to quit");
185     if (fgets(option, sizeof(option), stdin) == NULL) {
186         puts("EOF while reading stdin, will quit now..");
187         break;
188     }
189
190     if (option[0] == 'q')
191         break;
192
193     if (option[0] == 'h')
194         pjsua_call_hangup_all();
195      }
196
197      /* Destroy pjsua */
198      pjsua_destroy();
199
200      return 0;
201 }
```

Listagem B.1: Código original simple_pjsua

ANEXO C – Monitoramento Wireless

```
rede=$'projeto1'
canal=$'1'

parametro=$1

tamanho=$(expr length "$parametro")

if [ "$tamanho" -le 3 ] || [ "$tamanho" -ge 7 ]; then
   echo'';
   echo "Parametro nao e valido! Informe o nome de uma interface existente!";
   echo "Utilize: monitoramento <interface>";
   echo "Exemplo: monitoramento eth0";
   echo '';
   exit 0;
fi

filtro=''
filtro=$(sed -n '/'$parametro'/ p' /proc/net/dev)
echo
echo

if [ -n "$filtro" ]; then
   echo "O parametro passado e valido!";
   echo "A interface existe.";
   echo
else
   echo
   echo "Parametro nao e valido! Informe o nome de uma interface existente!"
   echo "Utilize: monitoramento <interface>"
   echo "Exemplo: monitoramento eth0"
   echo
   exit 0
fi
```

```
34
35 NM=/etc/init.d/network-manager
36 SVC=service
37 SNM=network-manager
38
39 echo '#########################################################'
40 echo '#######         Servico NetworkManager         #######'
41 echo '#########################################################'
42 echo '### Status do Servico NetworkManager ###'
43 echo "`$SVC $SNM status`"
44 echo ''
45 sleep '1'
46 echo '### Parando o NetworkManager ###'
47 echo "`$SVC $SNM stop`"
48 sleep '2'
49 echo ''
50 echo ''
51 echo '#########################################################'
52 echo '#######            Processos dhclient          #######'
53 echo '#########################################################'
54
55 echo '### Lista os numeros PID dos processos dhclient ###'
56 echo "`ps aux |grep dhclient |cut -c11-15`"
57 echo "### Finaliza processos dhclient ###"
58 echo "`killall dhclient; killall dhclient3`"
59 echo ''
60 echo ''
61
62 echo '#########################################################'
63 echo '#######              Processo udhcp             #######'
64 echo '#########################################################'
65
66 killall udhcpc
67
68 iwconfig $parametro mode managed
69 iwconfig $parametro channel $canal key off essid $rede
70
71 udhcpc -i $parametro
72
73 pid_udhcpc=''
74 pid_udhcpc=$(pidof udhcpc)
75
76 if [ -n $pid_udhcpc ] && [ $pid_udhcpc != "" ]; then
```

```
77    echo "O numero PID do processo udhcpc e: $pid_udhcpc"
78    else
79    echo "Erro: Processo nao foi iniciado corretamente."
80    echo "PID do processo udhcpc nao foi encontrado."
81    exit 0
82  fi
83
84  echo "..."
85
86  ip monitor | while read linha
87  do
88
89  var_down=$(echo $linha | awk '$8 ~ "state" {print $9}' | grep 'DOWN')
90
91    if [ "$var_down" = 'DOWN' ]; then
92
93       if [ "$rede" = 'projeto1' ]; then
94          iwconfig wlan0 mode managed
95          iwconfig wlan0 channel 2 key off essid projeto2
96          rede=$'projeto2'
97          echo "Rede atual: $rede"
98          kill -USR2 $pid_udhcpc
99          kill -USR1 $pid_udhcpc
100
101      elif [ "$rede" = 'projeto2' ]; then
102         iwconfig wlan0 mode managed
103         iwconfig wlan0 channel 1 key off essid projeto1
104         rede=$'projeto1'
105         echo "Rede atual: $rede"
106         kill -USR2 $pid_udhcpc
107         kill -USR1 $pid_udhcpc
108      fi
109
110 done
```

Listagem C.1: *Script* Monitoramento *Wireless*

Lista de Abreviaturas

AP *Access Point*

API *Application Programming Interface*

CH *Correspondent Host*

CoA *Care-of-Address*

DHCP *Dynamic Host Configuration Protocol*

GPL *General Public License*

IETF *Internet Engineering Task Force*

HoA *Home Address*

HTTP *HyperText Transfer Protocol*

IP *Internet Protocol*

MEGACO *Media Gateway Control Protocol*

MH *Mobile Host*

MIPv6 *Mobile Internet Protocol version 6*

MSCTP *Mobile Stream Control Transmission Protocol*

NAT *Network Address Translation*

PC *Personal Computer*

RFC *Request for Comments*

RPTC Rede Pública de Telefonia Comutada

RTP *Real-Time Transport Protocol*

RTCP *Real-Time Transport Control Protocol*

RTSP *Real Time Streaming Protocol*

SCTP *Stream Control Transmission Protocol*

SDP *Session Description Protocol*

SIP *Session Initiation Protocol*

SMTP *Simple Mail Transfer Protocol*

SO Sistema Operacional

TCP/IP *Transmission Control Protocol/Internet Protocol*

uDHCPc *micro Dynamic Host Configuration Protocol client*

UA *User Agent*

UAC *User Agent Client*

UAS *User Agent Server*

UDP *User Datagram Protocol*

URI *Uniform Resource Identifier*

VoIP Voz sobre IP

Wi-Fi *Wireless Fidelity*

Referências Bibliográficas

ANDERSEN, E. *BUSYBOX*. Busy-Box, 2008. Disponível em: <http://www.busybox.net/>.

ANDERSEN, E. *uClibc*. uClibc, 2008. Disponível em: <http://www.uclibc.org/>.

DROMS, R. *Dynamic Host Configuration Protocol*. IETF, mar. 1997. RFC 2131 (Draft Standard). (Request for Comments, 2131). Updated by RFCs 3396, 4361. Disponível em: <http://www.ietf.org/rfc/rfc2131.txt>.

DUTTA, A. et al. *Comparative Analysis of Network Layer and Application Layer IP Mobility Protocols for IPv6 Networks*. Wireless Personal Multimedia Communications - WPMC, sep 2006. Disponível em: <http://www.cs.columbia.edu/ dutta/research/WP061002.PDF>.

KANGKUNG, W. *[pjsip] REALM problem*. pjsip.org, nov 2007. Disponível em: <http://lists.pjsip.org/pipermail/pjsip_lists.pjsip.org/2007-November/000671.html>.

KOZIEROK, C. M. *DHCP General Operation and Client Finite State Machine*. The TCP/IP Guide, sep 2005. Disponível em: <http://www.tcpipguide.com/free/t_DHCPGeneralOperationandClientFiniteStateMachine.htm>.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet - Uma Nova Abordagem*. [S.l.]: Pearson, 2003.

MORIMOTO, C. E. *Entendendo os sistemas embarcados*. Guia do Hardware, jun 2009. Disponível em: <http://www.guiadohardware.net/artigos/entendendo-sistemas-embarcados/>.

PERKINS, C. *IP Mobility Support*. IETF, out. 1996. (Request for Comments, 2002). Disponível em: <http://www.ietf.org/rfc/rfc2002.txt>.

PINHEIRO, R. J. Linux e sistemas embarcados. *SlideShare*, Rio de Janeiro, RJ, Brasil, nov 2007.

PISA, P. S. *SIP (Session Initiation Protocol)*. Universidade Federal do Rio de Janeiro - UFRJ, oct 2008. Disponível em: <http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/pisa/>.

PRIJONO, B. *pjsip.org*. sep 2007. Disponível em: <http://www.pjsip.org>.

QUINTERO, N. F. M. *Middleware para os protocolos SCTP e HIP*. Centro de Estudos das Telecomunicações – CETUC, jul 2007. Disponível em: <http://www-di.inf.puc-rio.br/ endler/courses/Mobile/Monografias/07/Mid.SCTP.HIP-FelipeMaya-mono.doc>.

RIBEIRO, G. *Como Funciona a telefonia 3G*. How Stuff Works, jan 2008. Disponível em: <http://informatica.hsw.uol.com.br/telefonia-3g1.htm>.

RIEGEL, M. *Mobile SCTP - Transport Layer Mobility Management for the Internet*. out. 2002. Disponível em: <www.max.franken.de/021010-mobile-sctp4softcom.pdf>.

ROSENBERG, J. et al. *SIP: Session Initiation Protocol*. IETF, jun. 2002. RFC 3261 (Proposed Standard). (Request for Comments, 3261). Updated by RFCs 3265, 3853, 4320. Disponível em: <http://www.ietf.org/rfc/rfc3261.txt>.

SCHULZRINNE, H. The session initiation protocol (sip). *Columbia University*, New York, NY, USA, 2001.

SCHULZRINNE, H.; WEDLUND, E. Application-layer mobility using sip. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM Press, New York, NY, USA, v. 4, n. 3, p. 47–57, 2000. ISSN 1559-1662.

SILVA, C. da; MEDEIROS, F. Plataforma para o estudo de mobilidade na camada de rede, monografia. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Campus São José, 2009.

STEWART, R. et al. *Stream Control Transmission Protocol*. IETF, out. 2000. (Request for Comments, 2960). Disponível em: <http://www.ietf.org/rfc/rfc2960.txt>.

UDUGAMA, A. *Manipulating the Networking Environment Using RTNETLINK*. Linux Journal, mar 2006. Disponível em: <http://www.linuxjournal.com/article/8498>.