

Universidade Federal de Campina Grande
Departamento de Engenharia Elétrica
Trabalho de Conclusão de Curso

Estudo Avançado sobre o Protocolo de Inicialização de Sessão SIP e sua Arquitetura

Daniel da Costa Uchôa
daniel@dee.ufcg.edu.br

Disciplina:
Projeto em Engenharia Elétrica

Orientador:
Angelo Perkusich

Campina Grande, Junho de 2005

Conteúdo

1	Introdução	1
2	Protocolos de Sinalização para Voz sobre IP	3
3	O Propósito do SIP	7
4	Elementos da Rede SIP	10
4.1	User Agents	10
4.2	Servidores <i>Proxy</i>	11
4.2.1	Utilização de Servidores <i>Proxy</i>	14
4.3	<i>Registrar</i>	15
4.4	Servidor de Redirecionamento	17
4.5	<i>Gateways</i>	19
5	Mensagens SIP	20
5.1	Pedidos	21
5.2	Respostas	23
6	Estrutura do Protocolo	26
6.1	Transações	26
6.2	Diálogos	27
6.3	Arquitetura da Aplicação	28
6.4	Arquitetura do Protocolo	29
7	Cenários Típicos do SIP	32

<i>CONTEÚDO</i>	2
7.1 Registro	32
7.2 Convite	32
7.3 Finalização de Sessão	33
8 Conclusão	35

Lista de Figuras

4.1	Atuação das partes constituintes de um <i>user agent</i> no estabelecimento e finalização de sessão (redesenhado de [7]).	12
4.2	Convite de sessão entre <i>user agents</i> de duas redes diferentes.	15
4.3	Registro de um usuário no servidor <i>Registrar</i>	17
4.4	Servidor de Redirecionamento SIP.	18
5.1	Convite para abertura de sessão: pedido INVITE [15].	22
5.2	Resposta 401 SIP. O pedido do cliente não foi autorizado pelo servidor.	24
5.3	Respostas SIP com seu código de <i>status</i> e a classe pertencente.	25
6.1	Troca de mensagens entre dois <i>user agents</i> durante uma conversação. São ilustrados os conceito de transação e diálogo.	27
6.2	Pilha de protocolos de uma aplicação SIP.	29
6.3	Arquitetura do protocolo SIP.	31
7.1	Fluxo da mensagem REGISTER.	33
7.2	Fluxo da mensagem INVITE.	34
7.3	Fluxo da mensagem BYE.	34

Resumo

Um novo paradigma para desenvolvimento de aplicativos está surgindo. Estes não mais serão entendidos como aplicações isoladas, e sim em profunda interação com diversos tipos de dispositivos que utilizam plataformas variadas. As ferramentas e bibliotecas de desenvolvimento de aplicativos SIP ainda não atingiram um nível de abstração adequado, exigindo dos engenheiros e desenvolvedores um profundo conhecimento sobre o protocolo, afim de implementarem uma comunicação fim-a-fim entre os dispositivos dos usuários, um requisito essencial para novas aplicações. Assim, este trabalho tem como objetivo oferecer informações importantes sobre este novo protocolo, como sua estrutura de mensagens e sua arquitetura, conhecimentos extremamente necessários à desenvolvedores que desejam criar aplicações capazes de oferecer um superconjunto de novos serviços.

Capítulo 1

Introdução

O atual mundo dinâmico que vivenciamos não mais tolera a prestação de serviços isolados e limitados. Os usuários de serviços modernos querem facilidade e flexibilidade entre os diversos serviços oferecidos pelas diversas redes existentes. Temos os serviços de voz em telefones fixos e móveis, serviços de Internet e dados, a transmissão de rádio e televisão, cada qual utilizando uma arquitetura de rede separada. Este conceito de redes específicas e isoladas está com seus dias contados, pois não possibilita troca de informação entre as redes.

Os usuários de hoje querem mobilidade. Uma sessão de videoconferência, por exemplo, poderia ser iniciada num PC do escritório, continuada em um terminal móvel, e concluída na tela de TV de uma residência. A *convergência IP*¹, portanto, tem um papel crucial neste contexto, já que este protocolo vem se tornando o padrão para implementação da camada de rede dos mais variados tipos de rede.

Convergência IP é a fusão de serviços de voz e dados com o domínio da Internet, utilizando redes baseadas em comutação de pacotes como uma infraestrutura comum, de forma a enriquecer a comunicação entre indivíduos. Com os protocolos Internet (IP) é possível seduzir o usuário final pela combinação de voz, vídeo, texto, e etc. Esta convergência entre diferentes redes reduz custos, pois elimina a redundância de *hardware* que existia anteriormente, onde cada tipo de serviço prestado possuía uma rede isolada. Permite também

¹IP é o protocolo Internet (*Internet Protocol*) da camada de rede do modelo OSI para interligação de redes heterogêneas. Sua principal tarefa é o roteamento de pacotes, não permitindo congestionamento [1].

a implantação, integração e interação de novos serviços, que possibilitam novas formas de comunicação e condução de negócios.

No contexto da convergência IP, serviços e aplicações são redefinidos. Aplicações não são mais entidades isoladas trocando informações apenas com a interface do usuário. A próxima geração de aplicações envolverá conectividade fim-a-fim entre terminais IP. Esta capacidade de estabelecimento de conexões fim-a-fim é o ingrediente chave para permitir uma comunicação mais rica.

Para a iniciação desta conexão fim-a-fim o protocolo de abertura de sessão SIP² pode ser utilizado. Este é um protocolo usado para criar, modificar e terminar sessões com um ou mais participantes, além de negociar os codificadores/decodificadores (*codecs*), localização de usuários, redirecionamento de mensagens, entre outras funções [2].

Neste trabalho, vamos entender o protocolo SIP e sua arquitetura no contexto da *telefonia IP*. O termo telefonia IP é comumente utilizado para especificar a entrega de um superconjunto de serviços avançados da rede telefônica pública comutada PSTN³ usando telefones ou acessos IP, transporte e controle de redes baseadas no protocolo IP [7]. A transmissão dos sinais de voz em tempo real neste tipo de rede é conhecida como *voz sobre IP* (VoIP⁴).

As principais vantagens da implementação VoIP podem ser resumidas em uma economia significativa na manutenção da rede e nos custos de operação, e a rápida integração de novos serviços.

²*Session Initiation Protocol* - definido na RFC 3261 [2]

³*Public Switched Telephone Network*

⁴Voice over IP

Capítulo 2

Protocolos de Sinalização para Voz sobre IP

As informações de controle para configuração, supervisão e finalização de chamadas, bem como para o tratamento de situações anormais em sistemas de comunicação são basicamente um conjunto de mensagens designadas por *sinalização*. A sinalização em redes de telecomunicações remonta aos primórdios da telefonia, onde uma telefonista operava o sistema, atendendo as chamadas para ela direcionadas e repassando-as para os locais de destino. Desde então, muito se avançou nos protocolos de sinalização, que passaram a operar automaticamente nas centrais comutadoras Strowger [13], que evoluíram para centrais digitais, e chegando agora a serem executados em computadores especializados chamados *roteadores*. Hoje, com a telefonia IP, o trabalho de comutação que era realizado por enormes PABXs¹ pode ser facilmente executado por computadores pessoais de uso geral (PCs), bastando para isto que este seja equipado com uma placa de telefonia PCM/CIA para comunicação com a rede analógica de telefonia fixa (o que, dependendo do caso, quando se deseja apenas utilizar a rede de computadores local, não é nem necessária), e um software para o interfaceamento

¹PABX se refere a um PBX automaticamente controlado. PBX é uma acrônimo para *Private Branch eXchange*. Este termo refere-se genericamente a qualquer sistema de comutação possuído ou arrendado por empresas ou organizações para prover tanto funções de chaveamento internas quanto acesso à rede PSTN [13].

entre esta rede analógica e a rede local IP e para o roteamento de chamadas na rede.

Na rede telefônica pública comutada, PSTN, um protocolo de sinalização bastante conhecido utilizado para controle de chamada e interligação de redes é o SS7, o qual é um protocolo de sinalização por canal comum, onde a sinalização é fornecida em comum para um conjunto de circuitos (canais dedicados)[11]. As vantagens de utilização deste tipo de abordagem de sinalização está no fato de se separar a informação propriamente dita da informação de controle (sinalização), o que diminui a sobrecarga de processamento, pois processadores especializados para comutação podem ser utilizados em separado, ou seja, aliviando os processadores de tráfego desta tarefa. Outras vantagens podem ser assinaladas: possibilidade de implementação de mecanismos de detecção e correção de erros, e segurança contra interrupções.

A primeira geração de protocolos de sinalização e controle do meio para VoIP foram lançados em 1996 pela ITU-T² com a sigla H.225/H.245, definidos sob a “tutela” do protocolo H.323 [14]. Tais protocolos pretendiam oferecer serviços de voz sobre IP (VoIP) em tempo real baseados em redes locais (LANs³). Porém, segundo Khasnabish [9], os provedores de serviços de telecomunicações (conhecidos pela abreviação *telcos*) encontraram os dois seguintes problemas com a versão inicial do protocolo H.323 (versão 1):

1. A maioria das propriedades de chamada avançadas e serviços desejados no domínio PSTN não puderam ser facilmente implementados usando o H.323 pela falta de “abertura” (no sentido de procedimentos internamente definidos) por parte dos fabricantes de equipamentos;
2. Implementações escaláveis não eram nem viáveis e nem de custo efetivo, pelo fato de precisarem utilizar o que se chama de *statefull proxys*, que são servidores que armazenam todas as informações das chamadas para finalidade de acompanhamento do estado da transação.

Estes problemas motivaram a ITU-T a lançar a segunda versão do H.323 em 1998. Mas em 1999 a IETF⁴ lançou a primeira versão de um novo paradigma de protocolo baseado

²*International Telecommunication Union - Telecommunication*

³*Local Area Network*

⁴*Internet Engineering Task Force*

em Web⁵ com semântica bem definida para inicialização sessão: o SIP⁶. Este protocolo foi criado para controle de chamadas VoIP e criação e gerenciamento de serviços (um super-conjunto de serviços de domínio PSTN). O SIP apresentou a solução para os problemas acima enumerados: ele suporta o que se chama *stateless proxies*, ou seja, servidores que não precisam armazenar informações sobre o estado de uma chamada. Eles apenas redirecionam as mensagens, ficando a lógica de acompanhamento das transações por parte do dispositivo final. Isto tornou a implementação escalável de VoIP muito mais viável do que era possível utilizando-se o H.323.

A força do SIP pode ser resumida em três características básicas: escalabilidade, extensibilidade e flexibilidade. Primeiramente, o SIP baseia-se no modelo Internet para escalabilidade, o que significa que o centro da rede é rápido e não-complexo, enquanto que a parte de fora da rede (terminais) é mais complexa e lenta. Em segundo lugar, o SIP é extensível sem perder a compatibilidade com versões anteriores. Os clientes SIP podem possuir muitos tipos de extensões. Os clientes então discutirão capacidades comuns quando iniciando uma sessão. Portanto, a sessão sempre será completada entre dois clientes, independente de suas versões do protocolo. Por último, o SIP é flexível pelo fato de não apresentar limitações quanto a arquitetura da rede de suporte. Isto dá aos *telcos* liberdade de determinar como empregar o SIP em suas redes.

A corrida para auto-afirmação como protocolo de sinalização VoIP padrão continuou. Tanto a ITU-T como a IETF lançaram novas versões de seus protocolos. Muitos especialistas sugerem que a implementação do SIP seja bem mais simples, o que é realmente verdade pelo fato deste se basear em mensagens baseadas em texto, tal como o HTTP⁷, mas não só por isto.

Podemos citar algumas diferenças fundamentais entre estes dois protocolos. Primeiro, a arquitetura do H.323 é baseada em servidores centralizados, enquanto que o SIP pode funcionar sem servidores. Segundo, o H.323 inclui tanto sinalização quanto tratamento de mídia, ao invés de ser modular como o SIP. Terceiro, o H.323 é especificado pela ITU-T,

⁵Web refere-se a um *framework* arquiteturado para acesso de documentos “linkados” (unidos) espalhados por milhares de máquinas conectadas à Internet conhecido como *World Wide Web*(WWW).

⁶Session Initiation Protocol

⁷*HyperText Transfer Protocol*. É o protocolo de transferência padrão da Web [12]

e é baseado em outros padrões específicos da ITU-T. Por outro lado, o SIP é um padrão aberto, e utiliza outros padrões abertos amplamente empregados. Portanto, é mais fácil de implementar. Além disso, o SIP se adapta melhor para plataformas múltiplas pela sua modularidade.

Optamos então pelo estudo do protocolo SIP por este apresentar uma enorme facilidade para adição de novos serviços, como os mensagem instantânea, vídeo conferência, vídeo sob demanda, e etc. , bem como ser de fácil aplicação e implementação. Um argumento importante também é a necessidade de convergência entre as redes, sendo a *suite* de protocolos IP o padrão universal atual. Portanto, é quase que mandatório a utilização do protocolo SIP, o qual foi projetado de acordo com o modelo Internet para ser executado sobre esta *suite* IP.

Capítulo 3

O Propósito do SIP

Há muitas aplicações na Internet que necessitam da criação e gerenciamento de uma *sessão*, onde uma sessão é considerada uma troca de dados entre uma associação de participantes [2]. A implementação de tais aplicações se torna complicada devido ao comportamento destes participantes: usuários podem se movimentar entre terminais, podem ser endereçados por múltiplos nomes, e eles podem se comunicar através da utilização de diferentes meios, algumas vezes simultaneamente.

Diversos protocolos para transporte de várias formas de dados de sessão multimídia em tempo real, tal como voz, vídeo, ou mensagens de texto, foram criados. O protocolo de inicialização de sessão SIP¹ trabalha em conjunto com estes protocolos, possibilitando que terminais de Internet se descubram e concordem com uma caracterização de sessão que eles desejem compartilhar.

O SIP é uma ferramenta de propósito geral ágil, capaz de estabelecer, modificar, e finalizar sessões multimídias (conferências). O SIP trabalha independentemente dos protocolos de transporte usados na camada mais a baixo, e sem depender do tipo de sessão que está sendo estabelecido.

O SIP é um protocolo de controle de camada de aplicação projetado para fácil implementação, com boa escalabilidade e flexibilidade. Como dito na Seção 2, o SIP foi desen-

¹*Session Initiation Protocol*

volvido pela IETF, e sua especificação é disponível sob forma de diversas RFCs², sendo a RFC 3261 [2] a mais importante, pois contém a especificação do núcleo do protocolo.

O propósito deste protocolo é apenas fazer com que a comunicação seja possível, mas a comunicação propriamente dita precisa ser alcançada por outros meios. Isto quer dizer que o SIP não é um sistema de comunicação verticalmente integrado, ao invés disto, ele é um componente que pode ser usado com outros protocolos IETF para construir uma arquitetura multimídia completa.

Tipicamente, tais arquiteturas irão incluir protocolos como o RTP³ [3] para transporte de dados em tempo real e provisionamento de uma realimentação sobre qualidade de serviço, QoS⁴, o RTSP⁵ [4] para controle de entrega do fluxo (*streaming*) de mídia, o MEGACO⁶ para controle dos *gateways* (definido mais a frente) para a rede PSTN, e o SDP⁷ para descrever sessões multimídias. Todos estes protocolos fazem parte do ambiente IP, que chamamos de *suíte* IP. Portanto, o SIP pode ser usado conjuntamente com outros protocolos com a finalidade de prover serviços completos aos usuários. Entretanto, a funcionalidade básica e operação do SIP não dependem de nenhum destes protocolos.

O SIP dá suporte a cinco propriedades importantes no estabelecimento e finalização de comunicações multimídia:

- **situação do usuário:** determinação do sistema terminal que será usado para comunicação;
- **disponibilidade do usuário:** determinação da disposição da parte chamada para entrar em comunicação;
- **configuração de sessão:** estabelecimento de parâmetros de sessão em ambas as partes destino e fonte;
- **gerenciamento de sessão:** inclui transferência e finalização de sessões, modificação

²*Request For Comments*

³*Real-Time Transport Protocol*

⁴*Quality of Service*

⁵*Real-Time Streaming Protocol*

⁶*Media Gateway Control Protocol*

⁷*Session Description Protocol*

dos parâmetros de sessão, e a invocação de serviços.

Ao invés de oferecer serviços, o SIP oferece primitivas as quais podem ser utilizadas para a implementação de diferentes serviços. Como exemplo do uso das primitivas do SIP para a implementação de serviços, podemos citar que ele pode localizar um usuário e entregar um objeto opaco (pois para o SIP não interessa o que há no objeto) na localização corrente do usuário desejado (exemplo retirado de [2]). Se esta primitiva for usada para a entrega de um descritor de sessão escrito em SDP, por exemplo, os terminais poderão concordar nos parâmetros da sessão. Se a mesma primitiva for usada para entregar além da descrição SDP também uma foto da pessoa que disca, um serviço de identificador de chamadas poderá ser facilmente implementado. Ou seja, como os exemplos mostram, uma única primitiva é tipicamente usada para prover diferentes tipos de serviços.

O SDP é um protocolo de descrição de sessão utilizado para descrever e codificar características dos participantes da sessão, de forma que todos possam concordar com o *codec* ou o protocolo de transporte utilizados, permitindo que a conversação seja estabelecida.

Capítulo 4

Elementos da Rede SIP

Embora, na sua forma mais simples, seja possível o uso de apenas dois *user agents* enviando mensagens diretamente de um para o outro, uma rede típica SIP irá possuir mais de um tipo de elemento SIP.

Basicamente, os elementos que constituem o mundo SIP são os *user agents*, os *proxies*, os *registrars*, os servidores de redirecionamento (*redirect servers*), e os *gateways* de sinalização para interoperação com outros protocolos de sinalização. Vamos discuti-los brevemente nesta seção.

4.1 User Agents

Os *user agents* (UA) são terminais da Internet que utilizam o SIP para localizar um ao outro e para negociar características de sessão [7]. Usualmente, são aplicativos residentes nos dispositivos do usuário. Estes dispositivos podem ser computadores, aparelhos celulares, *gateways* PSTN¹, *handhelds* (PDAs), e etc.

Os *user agents* dividem-se em:

1. *User Agent Client (UAC)*: é uma entidade lógica do UA que cria novos pedidos, e então utiliza a maquinaria de transação de estados para enviá-los. O papel do UAC

¹*Public Switched Telephone Network*

se prolonga enquanto a duração da transação². Em outras palavras, se uma parte do código de um software inicia um pedido, ele atua como um UAC enquanto durar a transação. Se ele recebeu um pedido antes, ele atuará como um *user agent server* até o fim da transação;

2. *User Agent Server (UAS)*: faz o papel inverso do UAC. É uma entidade local que gera respostas à pedidos SIP. A resposta aceita, rejeita, ou redireciona pedidos. Ou seja, se uma parte do código de um software SIP responde a pedidos, ele age com um UAS.

Deve-se notar que o UA ora atua como um UAC ora com UAS, ou seja, a parte do *user agent* que executa uma chamada age com um UAC quando envia um pedido INVITE, convidando uma outra parte para abertura de sessão, e recebendo a resposta ao pedido. Quando uma parte do código deste UA recebe um pedido e envia uma resposta, ele age como um UAS.

Porém, a situação se inverte quando a parte chamada decide finalizar a sessão enviando um pedido BYE. Neste caso, a parte chamada do UA (enviando um BYE) se comporta como um UAC, e a parte que está chamando, no outro UA, se comporta como um UAS.

Um exemplo retirado de [7] ilustra bem esta troca de papéis das partes que constituem um UA: na Figura 4.1 é mostrado três *user agents* e um *stateful forking proxy*. Cada *user agent* contém um UAC e UAS. A parte do *proxy* que recebe o INVITE a partir da quem realizou a chamada, de fato, atua como um UAS. Quando este está repassando o pedido e armazenado o estado da chamada, o *proxy* cria dois UACs, cada um responsável por um braço da chamada. Quando a parte B chamada decide terminar a sessão, ele envia um BYE. Neste instante, o *user agent* que previamente se comportava como UAS torna-se um UAC e vice-versa.

4.2 Servidores *Proxy*

O SIP permite a criação de *hosts* de rede chamados servidores *proxy*. Estes são entidades

²Vamos definir transações e diálogos mais à frente.

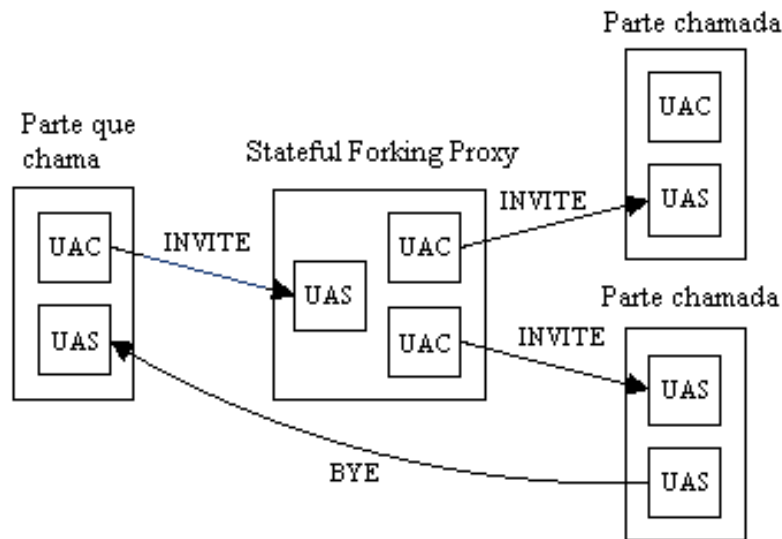


Figura 4.1: Atuação das partes constituintes de um *user agent* no estabelecimento e finalização de sessão (redesenhado de [7]).

intermediárias que atuam como servidor e cliente para o propósito de se fazer pedidos a outros clientes fora do alcance da parte que realiza a chamada.

Os servidores *proxy* primeiramente executam ações de roteamento de convite de sessão, de acordo com a localização corrente do convidado, o que significa que seu trabalho é garantir que um pedido seja enviado de uma entidade para outra mais próxima possível do usuário alvo desejado. Dessa forma, usualmente, o convite para sessão irá atravessar um conjunto de *proxies* até encontrar um que conheça a localização atual da parte chamada. Tal *proxy* irá repassar o convite para sessão diretamente para a parte chamada, e esta por sua vez irá aceitar ou recusar o convite para abertura de sessão.

Proxies são também úteis na implantação de políticas de uso, certificando-se de que um certo usuário tem permissão para realizar uma chamada, por exemplo. Um *proxy* interpreta, e se necessário, reescreve partes específicas de uma mensagem de pedido antes de passá-la à frente. Estes servidores também realizam outras funções importantes, como autenticação e bilhetagem.

Um servidor *proxy* tem a opção de passar uma mensagem adiante sem alteração, ou ele pode alterar alguns campos do cabeçalho da mensagem. Assim, ele age ora com servidor,

recebendo mensagens, e ora como cliente, enviando mensagens, sendo que ele pode gerar uma resposta localmente e enviar para a parte que realizou o pedido.

Esta alteração nos campos de cabeçalho da mensagem realizada pelos servidores pode ser feita quando deseja-se que a mensagem, na volta, siga o mesmo caminho de ida. Isto pode acontecer por motivos de tarifação (ou bilhetagem) ou para controle de *firewall*. Assim, a informação é acrescentada nos cabeçalhos para que a mensagem de resposta passe pelos mesmos roteadores até a atingir a origem do pedido. Existem campos no cabeçalho das mensagens SIP que podem ser utilizados pelos *proxies* para controle de *loops* (laços) e para quando se deseja que os roteadores encaminhem não só as mensagens SIP de sinalização, mas todas as mensagens no caminho, ou seja, o *stream* de mídia não é mais estabelecido diretamente entre dois user agents SIP, e sim roteada também pelos *proxies*. Dessa forma, a comunicação deixa de ser *peer-to-peer* (fim-a-fim). As funções destes cabeçalhos serão explicadas mais à frente, quando estivermos falando sobre as mensagens SIP.

Existem dois tipos básicos de servidores *proxy* SIP:

1. **Servidores *Stateless***: apenas repassam as mensagens independentemente umas das outras. São entidades lógicas que não mantêm a máquina de estados de transação do cliente ou servidor quando processa um pedido, ou seja, eles não armazenam o estado da chamada, não se importando com transações. Um servidor *stateless proxy* passa adiante cada pedido que recebe no *downstream* e cada resposta que recebe no *upstream*;
2. **Servidores *Stateful***: São entidades lógicas que mantêm a máquina de estados de transação do cliente e servidor quando processa um pedido, ou seja, dado a recepção de um pedido, *stateful proxies* criam um estado e o acompanham até o fim da transação.

Os servidores *stateless* são mais simples e rápidos que os servidores *stateful*, porém são incapazes de descartar retransmissões de mensagens ou de executar roteamento avançado, como *forking* ou cruzamento recursivo.

O fato de os servidores *stateful* guardarem o estado da chamada lhes garante propriedades interessantes. Ele pode realizar uma busca paralela de localidades, ou seja, dado que ele recebeu apenas um convite de abertura de sessão ele pode enviar um INVITE para várias localidades ao mesmo tempo. Esta característica é denominada *forking*.

Servidores *proxy* podem descartar retransmissões, pois sabendo-se o estado da transação eles sabem se já receberam tal mensagem. Eles podem executar métodos mais avançados para busca de usuário. Um usuário pode ser chamado no escritório, e se não atender a chamada poderá ser redirecionada para seu telefone celular, por exemplo. A desvantagem é que possuem performance limitada por serem mais complexos e precisarem de memória suficiente para armazenar o estado de todas as chamadas que por ele estão passando.

As mensagens roteadas pelos servidores SIP contém informações suficientes para permitir que os servidores sejam *stateless*, porém a maioria dos proxies SIP hoje são *stateful*, pois freqüentemente ele precisam executar operações tipo *forking*, bilhetagem e algum tipo de cruzamento de NAT, além de todas as outras propriedades adicionais atribuídas a servidores *stateful proxy* acima descritas.

4.2.1 Utilização de Servidores *Proxy*

Em uma configuração típica, cada entidade centralmente administrada, como uma companhia, por exemplo, possui seu próprio servidor *proxy* SIP, o qual é utilizado por todos os *user agents* dentro desta localidade. Pode-se ilustrar o fluxo de mensagens trocadas por entidades SIP e a utilização destes *proxies* através da Figura 4.2 (baseado em exemplo contido em [7]). Mostra-se na figura duas companhias, A e B, cada uma delas com seu próprio servidor *proxy*. Na figura está representado o fluxo de informação gerado quando um funcionário da Companhia A, Joe, convida Bob, um empregado da Companhia B, para abertura de sessão.

Joe digita o endereço URI³ que deseja chamar (ou seleciona um nome de contato da sua agenda de endereços) a partir de um *user agent* residente em seu *handheld*. Ele deseja falar com Bob, cuja URI é `sip:bob@b.com`. O *user agent* de Joe não sabe como rotear o convite por ele mesmo, mas está configurado para enviar todo tráfego de saída que não seja para sua própria rede para o servidor *proxy* de sua companhia, cuja URI é `proxy.a.com`. O servidor percebe que o usuário `sip:bob@b.com` está em uma rede que não a sua, de uma outra companhia, e então procurará o servidor *proxy* da Companhia B para repassar o convite. O servidor *proxy* da Companhia A pode estar pré-configurado com o endereço de B, ou

³O endereço SIP é sob a forma de URI, *Universal Resource Identifier*.

proxy.a.com irá usar um servidor de DNS para resolver o nome b.com e descobrir o endereço proxy.b.com. O convite então alcança o *proxy* da Companhia B, que sabe que o usuário Bob está atualmente fora de seu escritório e redireciona a chamada para o aparelho celular de Bob, que decide se aceita ou não a chamada.

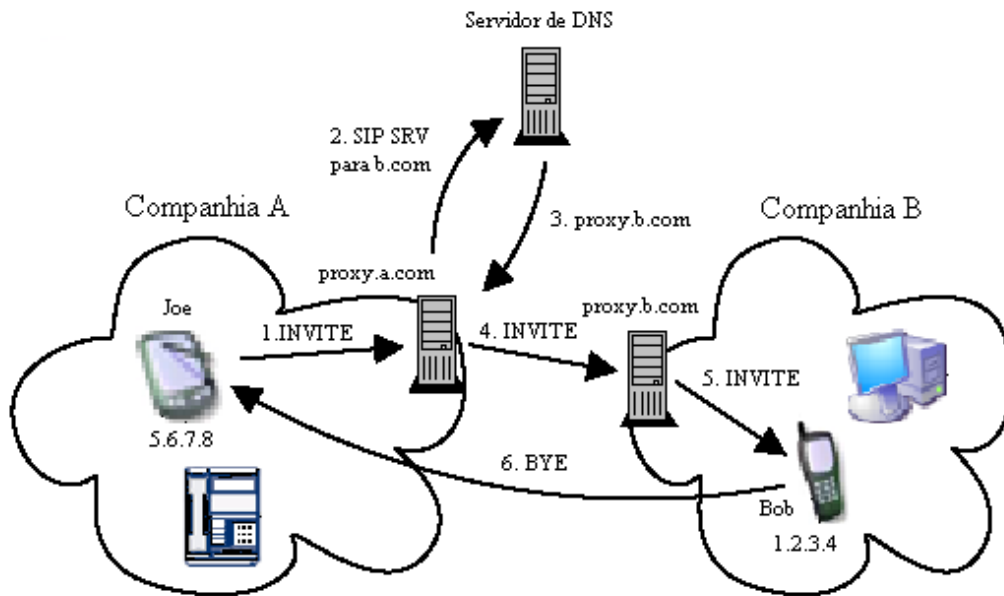


Figura 4.2: Convite de sessão entre *user agents* de duas redes diferentes.

4.3 Registrar

Para que a localização de um usuário seja conhecida por todos os usuários da rede, utiliza-se o *Registrar*. Esta é uma entidade SIP especial que recebe registros dos usuários, extrai informação sobre sua localização corrente, e armazena a informação dentro de uma base de dados. Um *Registrar* é um servidor que aceita pedidos de registro REGISTER e coloca a informação que recebe nestes pedidos dentro do serviço de localização (base de dados) para o domínio que ele trata.

O IP do usuário pode não ser fixo, ou seja, ele pode mudar devido à várias circunstâncias. Uma delas, que apesar de antiga ainda é uma opção bastante utilizada, é o usuário utilizar

uma conexão discada, onde este recebe de seu Provedor de Serviços (ISP⁴) um endereço IP dinâmico. O usuário pode fazer parte de uma rede local que também utilize o serviço DHCP, onde um servidor DHCP⁵ lhe atribui um endereço IP dinâmico atualmente disponível quando o *host* é inicializado (*boot*). Mesmo se tratando de usuários móveis estes terão um endereço *home* fixo, além do endereço local atual. Assim, é necessário manter o mapeamento dos endereços SIP com os endereços IP, e para isto utiliza-se o *Registrar*.

Voltando ao exemplo da Subseção 4.2, foi mencionado que o *SIP Proxy* em *proxy.b.com* saberia a localização corrente de Bob, mas não foi explicado como ele teria tal informação, ou seja, como o servidor *proxy* na Companhia B pode aprender a localização atual de um usuário. Vamos agora elucidar este problema.

O *user agent* residente no aparelho celular de Bob (*SIP phone*) precisa inicialmente se registrar em um *Registrar* antes de receber chamadas. Ele assim o faz através de uma solicitação REGISTER. O *Registrar* recebe o pedido, extrai a informação sobre o endereço IP da máquina, a porta (TCP ou UDP) na qual o *softphone* está esperando chamadas, e o nome do usuário que está atualmente utilizando o dispositivo.

O propósito da base de dados de localização é mapear a URI *sip:bob@b.com* com alguma coisa do tipo *sip:bob@1.2.3.4:5060*. A base de dados de localização é então usada pelo servidor *proxy* da Companhia B para quando chegar um convite para inicialização de sessão para *sip:bob@b.com* ele procurar a localidade atual de Bob. O servidor encontra o endereço *sip:bob@1.2.3.4:5060* e envia o convite para lá. Isto pode ser visto na Figura 4.3.

Apresenta-se nesta figura um registro típico de usuário. Uma mensagem contendo o endereço público do usuário *sip:bob.b.com* (chamado *Address of Record (AOR)*) que será gravado no *Registrar*, associando este endereço a uma outra URI de contato onde o usuário possa estar disponível, *sip:bob@1.2.3.4:5060*. Nesta última URI, o número 1.2.3.4 é o endereço IP do dispositivo (neste caso, um aparelho celular) que está pedindo registro. O *Registrar* extrai esta informação e a armazena na base de dados. Se tudo ocorrer bem, o *Registrar* envia um 200 OK como resposta, indicando que o pedido foi atendido com sucesso, e o processo de registro é finalizado com sucesso, repetindo-se periodicamente para se manter o *binding*

⁴Internet Service Provider

⁵Dynamic Host Control Protocol

(ligação).

O usuário deve enviar uma mensagem REGISTER periodicamente, afim de atualizar sua localização, pois o registro tem uma validade definida no campo do cabeçalho “Expires”. Assim, o *user agent* precisa “dar um *refresh*” no registro dentro de um tempo de vida pré-determinado. A ligação ou mapeamento entre a URI SIP (endereço) e o endereço de rede do usuário é denominada *binding*.

As mensagens SIP possuem um cabeçalho “Via” para controle do caminho seguido, para que evite-se *loops* (laços) de roteamento, o que faz com que as mensagens fiquem indo e voltando na rede, sem alcançar o destino desejado. Assim, em cada *hop* (salto) realizado pela mensagem na rede, ou seja, a cada *proxy* que ela alcança, verifica-se se o endereço desta máquina está na lista de endereços contida no campo “Via” do cabeçalho da mensagem (ver Capítulo 5).

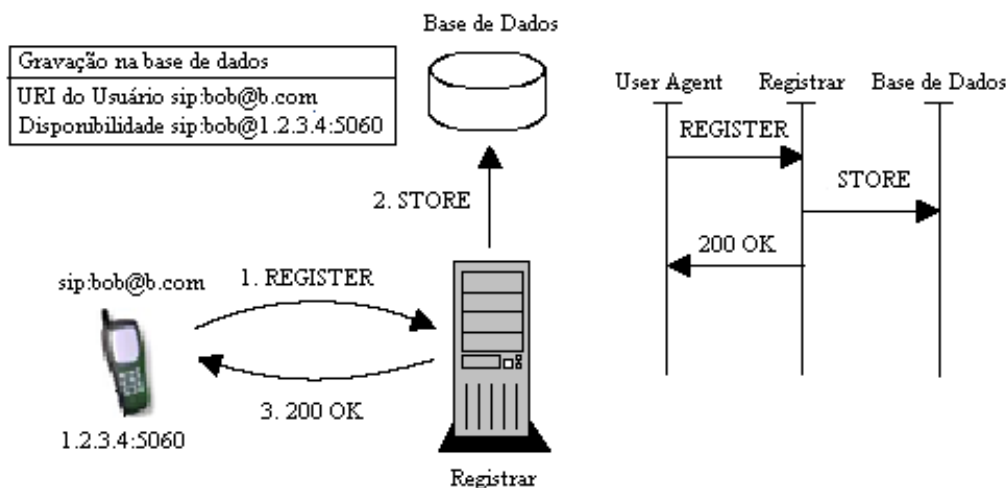


Figura 4.3: Registro de um usuário no servidor *Registrar*.

4.4 Servidor de Redirecionamento

A entidade que recebe um pedido e envia de volta uma resposta contendo uma lista da

localização corrente de um usuário particular é chamada de *Servidor de Redirecionamento*. Um Servidor de Redirecionamento é um *user agent server* (UAS) que gera respostas 3XX⁶ para os pedidos que recebe, direcionando o cliente a contatar um conjunto alternativo de URIs.

Um Servidor de Redirecionamento recebe pedidos e analisa o recipiente destes pedidos na base de dados de localização, criada pelo *Registrar*. Ele então cria uma lista de localizações correntes do usuário e envia esta lista para o originador do pedido em uma resposta dentro das respostas SIP 3xx de redirecionamento.

O originador do pedido então extrai a lista de destinos e envia um outro pedido diretamente para eles. Na Figura 4.4 é mostrado um redirecionamento típico.

Com este tipo de servidor pode-se distribuir chamadas entre grupos de servidores secundários, permitindo assim um melhor equilíbrio do tráfego. Dessa forma, um servidor de redirecionamento é uma ferramenta útil para melhorar o escalonamento dos servidores de distribuição de chamadas.

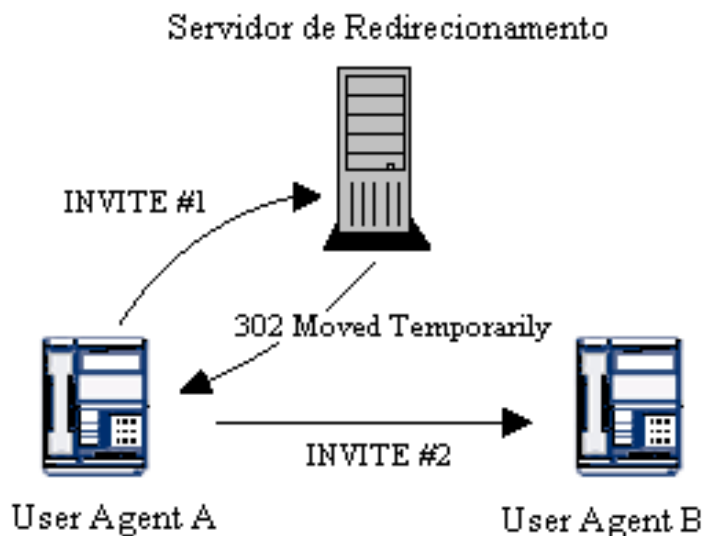


Figura 4.4: Servidor de Redirecionamento SIP.

⁶O SIP utiliza um código de três dígitos para identificar as respostas. O primeiro dígito define a classe da resposta. A classe 3XX começa em 300 e vai até 399. Uma resposta deste tipo pertence a classe de Redirecionamento, e indica que ações adicionais precisam ser tomadas para se completar o pedido. Vamos falar mais sobre esta e outras classes de resposta mais à frente.

4.5 Gateways

Os *Gateways* ficam localizados no limite (fronteira) da rede de sinalização do protocolo, e é necessário para interoperação através de diferentes domínios. Eles atuam como uma interface entre diferentes redes de sinalização usando mapeamento de protocolos. São tradutores de protocolos. Um exemplo de um protocolo que é executado por um *gateway* entre uma rede SIP e uma rede PSTN é o *Media Access Gateway Controller* (MEGACO) [5].

Capítulo 5

Mensagens SIP

A comunicação utilizando SIP, ou seja, a sinalização, é realizada pela troca de uma série de mensagens. As mensagens SIP são enviadas entre entidades SIP para que o protocolo execute sua função. As mensagens podem ser transportadas de forma independente pela rede. Usualmente, elas são transportadas em um datagrama UDP separado.

As mensagens podem ser classificadas de duas formas: *pedidos (requests)*, se iniciadas pelo cliente para o servidor; ou *respostas (responses)*, se enviadas do servidor para o cliente. Pedidos são usualmente utilizados para inicializar algum tipo de ação ou informar o conteúdo do pedido de alguma coisa. Respostas são usadas para confirmar que um pedido foi recebido e processado, e contém o *status* do processamento.

As mensagens SIP são baseadas em texto usando a sintaxe ABNF¹[6], com codificação de oito bits, utilizando um conjunto de caracteres UTF-8, e fim de linha CRLF². As mensagens consistem de uma linha de início, um ou mais cabeçalhos, uma linha em branco indicando o fim dos campos de cabeçalho, e um corpo de mensagem opcional [2]. Exceto pela diferença no conjunto de caracteres, muito da sintaxe das mensagens SIP e campos de cabeçalho é idêntica às mensagens HTTP.

Uma mensagem SIP típica tem a forma representada pela Figura 5.1 [15]. A primeira linha indica que esta mensagem é um pedido INVITE, usada para estabelecer uma sessão. IN-

¹*Augmented Backus-Naur Format*

²*Carriage Return Line Feed*

VITE é o nome do método desejado. A URI na primeira linha, `sip:7170@iptel.org` é chamada *Request URI*, ou seja, a URI de pedido, e contém a URI do próximo salto (*hop*) da mensagem. Neste caso, será o *host* `iptel.org`.

Uma mensagem SIP pode conter um ou mais campos de cabeçalho “**Via**”, que são usados para armazenar o caminho do pedido, permitindo que servidores intermediários possa retransmitir as respostas pelo mesmo caminho. Este é o único campo onde os valores devem estar em ordem seqüencial, já que será reutilizado pela resposta, seguindo a ordem inversa de saltos ou nós da rede. Como esta mensagem possui apenas um campo “**Via**”, pode-se dizer que o *user agent* está sendo executado no *host* `195.37.77.100`, e que ele aguarda respostas na porta `5060`, a porta padrão do SIP.

Os campos “**From**” e “**To**” identificam o originador (parte que chama) e o destino do convite (parte chamada). O campo “**From**” possui um nome opcional para o originador do convite, e este campo deve estar em todos os pedidos e respostas.

“**Call-ID**” contém um identificador globalmente único para este diálogo e seu propósito é identificar mensagens pertencentes a mesma chamada. “**CSeq**” é utilizado para manter a ordem de seqüência dos pedidos, pois como sabemos, as mensagens podem chegar fora de ordem. Dessa forma, este campo é usado para manter o controle de fluxo e controlar retransmissões. Assumindo-se “**CSeq**” = 1 para o INVITE inicial, pode-se supor que esta mensagem seja uma retransmissão.

O campo de cabeçalho “**Contact**” contém o endereço IP e a porta na qual o originador do pedido está esperando futuros pedidos enviados pela parte chamada.

O cabeçalho da mensagem é separado do corpo principal por uma linha em branco. O corpo de uma mensagem INVITE contém uma descrição do tipo de mídia aceito pelo originador do pedido, e é codificado em SDP (ver fim da Seção 3).

5.1 Pedidos

Os pedidos SIP são distinguidos por terem uma Linha de Pedido (*Request-Line*) para a linha de início. A linha de pedido contém o nome do método, a URI do pedido, e a versão

```

INVITE sip:7170@iptel.org SIP/2.0
Via: SIP/2.0/UDP 195.37.77.100:5040;rport
Max-Forwards: 10
From: "jiri" <sip:jiri@iptel.org>;tag=76ff7a07-c091-4192-84a0-
d56e91fe104f
To: <sip:jiri@bat.iptel.org>
Call-ID: d10815e0-bf17-4afa-8412-d9130a793d96@213.20.128.35
CSeq: 2 INVITE
Contact: <sip:213.20.128.35:9315>
User-Agent: Windows RTC/1.0
Proxy-Authorization: Digest username="jiri", realm="iptel.org",
algorithm="MD5", uri="sip:jiri@bat.iptel.org",
nonce="3cef75390000001771328f5ae1b8b7f0d742da1feb5753c",
response="53fe98db10e1074
b03b3e06438bda70f"
Content-Type: application/sdp
Content-Length: 451

v=0
o=jku2 0 0 IN IP4 213.20.128.35
s=session
c=IN IP4 213.20.128.35
b=CT:1000
t=0 0
m=audio 54742 RTP/AVP 97 111 112 6 0 8 4 5 3 101
a=rtpmap:97 red/8000

```

Figura 5.1: Convite para abertura de sessão: pedido INVITE [15].

do protocolo separados por um caracter de espaçamento simples. A linha de pedido termina com CRLF.

A especificação SIP contida na RFC 3261 [2] define seis métodos:

- **ACK**: um pedido ACK é enviado pelo cliente para confirmar que ele recebeu uma resposta do servidor;
- **BYE**: um pedido BYE é enviado para um agente de origem, ou a partir de um agente de destino, para interromper uma chamada;
- **CANCEL**: esse pedido deve ser enviado quando se quer interromper a transmissão antes de receber uma resposta do servidor;
- **INVITE**: é usado para iniciar uma chamada;
- **OPTIONS**: é um pedido enviado a um servidor para saber as capacidades, sendo que o servidor pode enviar de volta uma lista de métodos. Em alguns casos, o servidor pode

também enviar as capacidades de algum usuário requisitado;

- **REGISTER**: um cliente pode registrar um ou mais endereços de sua localização.

Os pedidos SIP possuem alguns campos adicionais no cabeçalho das mensagens para indicar características específicas, tais como:

- **Accept**: este cabeçalho opcional indica quais tipos de mídia são aceitáveis na resposta;
- **Accept-Language**: indica as linguagens preferidas pelo originador da chamada;
- **Expires**: para uma mensagem REGISTER esse campo indica quanto tempo o registro será válido. Para uma mensagem INVITE pode ser usado para limitar a duração de buscas;
- **Priority**: indica a prioridade do pedido;
- **Record-Route**: é usado por alguns proxies para adicionar ou atualizar o campo de cabeçalho se este desejar fazer parte do caminho de todas as mensagens trocas pelos *user agents*;
- **Subject**: campo de texto livre que deve fornecer alguma informação sobre a natureza da chamada.

5.2 Respostas

As respostas SIP são distinguidas dos pedidos por terem uma linha de status como suas linhas iniciais. Uma Linha de *Status* (*Status-Line*) consiste da versão do protocolo seguida por códigos numéricos de *status* e sua frase textual associada, com cada elemento separado por um caracter de espaçamento simples.

Na Figura 5.2 ilustra-se uma resposta típica.

O SIP utiliza um código de *status* de três dígitos para identificar as respostas. O primeiro dígito define a classe da resposta. Os outros dois dígitos não possuem nenhum

```
SIP/2.0 401 Proxy-Authenticate
Via: SIP/2.0/UDP 192.168.15.230:5060
From: sip:muda@colorado.edu
To: sip:doug@fcc.org;tag=124jahs09123h24h12j4h1j2
Call-ID: 12488221@192.168.1.30
CSeq: 22 INVITE
Contact: <sip:doug@236.87.48.68:5060;transport=udp>;q=0.00
Server: Winip (proxy)
Content-Length: 0
Proxy-Authenticate: Digest realm="att.com",
                    domain="sip:fcc.gov", qop="auth",
                    nonce="f84flceec41e6cbe5aea9c8e88d359",
                    opaque="", stale=FALSE, algorithm=MD5
```

Figura 5.2: Resposta 401 SIP. O pedido do cliente não foi autorizado pelo servidor.

papel categórico. Assim, qualquer resposta cujo código de *status* está entre 100 e 199 é referenciada como resposta classe 1XX, entre 200 e 299 como classe 2XX, e assim por diante. A versão 2.0 do SIP [2] permite seis valores diferentes para o primeiro dígito do código de *status*:

- **1XX: Provisional**– pedido recebido, continuando o processamento do pedido;
- **2XX: Success**– a ação foi recebida com sucesso, entendida e aceita;
- **3XX: Redirection**– ações adicionais precisam ser tomadas para completar o pedido;
- **4XX: Client Error**– o pedido contém erro de sintaxe ou não pode ser preenchido neste servidor;
- **5XX: Server Error**– o servidor falhou no preenchimento de um pedido aparentemente válido;
- **6XX: Global Failure**– o pedido não pode ser preenchido em qualquer servidor

Um quadro-resumo de todas as respostas separadas em suas classes é mostrado na Figura 5.3.

Cada pedido precisa ser respondido, exceto um ACK, que não pede resposta, senão cada reconhecimento precisaria ser também reconhecido, o que geraria um tráfego de controle

1xx – Informativo		Pedido continuando a processar o pedido
	100	Tentando
	180	Chamando
	181	A chamada está sendo retransmitida
	182	Colocado na fila
2xx – Sucesso		A ação foi recebida, atendida aceita com sucesso
	200	OK
3xx – Redirecionamento		Uma ação adicional deve ser tomada para completar o pedido
	300	Múltiplas escolhas
	301	Movido permanentemente
	302	Movido temporariamente
	380	Serviço alternativo
4xx – Erro de Cliente		O pedido contém sintaxe inválida ou não pode ser efetuado neste servidor
	400	Pedido inválido
	401	Não autorizado
	402	Necessário pagamento
	403	Proibido
	404	Não encontrado
	405	Método não permitido
	406	Não aceitável
	407	Necessária autenticação do proxy
	408	Tempo para o pedido esgotado
	409	Conflito
	410	Não mais presente
	411	Necessário fornecer o comprimento
	413	Corpo da mensagem de pedido muito grande
	414	URI do pedido muito grande
	415	Tipo de mídia não suportado
	420	Extensão inválida
	480	Temporariamente não disponível
	481	Transação ou leg de chamada não existe
	482	Loop (laço) detectado
	483	Excesso de hops (segmento)
	484	Endereço incompleto
	485	Ambíguo
5xx		Erro de servidor
	500	Erro interno ao servidor
	501	Não implementado
	502	Gateway inválido
	503	Serviço não disponível
	504	Tempo esgotado no gateway
	505	Versão SIP não suportada
6xx		Falha global
	600	Ocupado em todos os lugares
	603	Declínio
	604	Não existe em lugar nenhum
	606	Não aceitável

Figura 5.3: Respostas SIP com seu código de *status* e a classe pertencente.

exponencialmente crescente, esgotando rapidamente a capacidade de processamento da rede. A maior parte do cabeçalho de resposta é copiada do cabeçalho do pedido.

Capítulo 6

Estrutura do Protocolo

6.1 Transações

Embora foi dito que mensagens SIP sejam enviadas independentemente pela rede, elas usualmente são agrupadas dentro do que se chama de *transações* pelos *user agents* e certos tipos de *proxies*. Portanto, é dito que o SIP é um protocolo transacional.

Uma transação é uma seqüência de mensagens SIP trocadas entre os elementos da rede. Uma transação consiste de um pedido e todas as respostas àquele pedido.

As entidades SIP que possuem a noção de transação são ditas *stateful*. Tais entidades usualmente criam um estado associado com a transação e o mantêm na memória pela duração da transação. Quando um pedido ou uma resposta chega, entidades *stateful* tentam associar o pedido (ou resposta) com transações existentes. Para serem capazes de executar esta tarefa ele extraem um identificador único de transações da mensagem e comparam com os identificadores de todas as transações que eles acompanham. Se tal transação existir, seu estado é então atualizado.

Na Figura 6.1 são mostradas duas transações e um diálogo durante a conversação entre dois *user agents*. Os diálogos serão discutidos na próxima seção.

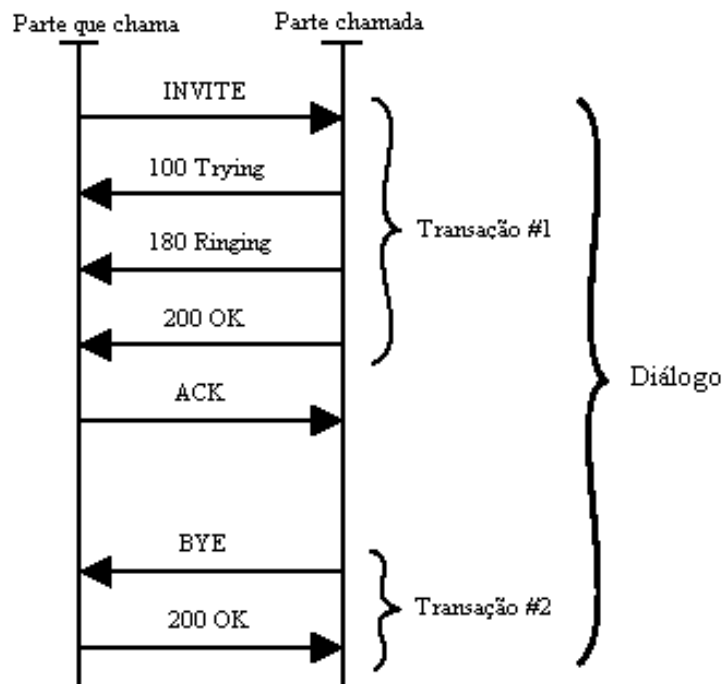


Figura 6.1: Troca de mensagens entre dois *user agents* durante uma conversa. São ilustrados os conceitos de transação e diálogo.

6.2 Diálogos

Foi mostrado uma transação que inclui o pedido INVITE e suas respostas, quando um *user agent* deseja convidar outro *user agent* para abertura de sessão, e uma outra transação formada pelo pedido BYE e suas respostas, quando se deseja finalizar uma sessão (ver Figura 6.1). Estas duas transações podem estar relacionadas de alguma forma, e assim ambas irão pertencer a um mesmo *diálogo*. Um diálogo é uma relação fim-a-fim que persiste por algum tempo entre dois *user agents*. Poderia-se dizer que um diálogo é uma seqüência de transações. Na Figura 6.1 está ilustrado este conceito.

Diálogos são identificados por um identificador de chamadas, o campo de cabeçalho “Call-ID”, um *tag* local e outro remoto, ou seja, os campos “From” e “To”. Mensagens pertencentes ao mesmo diálogo precisam ter estes campos iguais.

Diálogos facilitam o seqüencialmente e roteamento apropriados das mensagens entre

terminais SIP. Mensagens dentro de um diálogo são enviadas diretamente de um *user agent* para outro. Isto resulta em um importante melhoramento de performance, desde que *proxies* não precisam tomar conta de todas as mensagens dentro de um diálogo. Eles são utilizados apenas para rotear o primeiro pedido que estabelece um diálogo.

6.3 Arquitetura da Aplicação

Desde que o SIP é apenas um protocolo de sinalização, ele não se preocupa com a mídia utilizada. Portanto, outros protocolos são necessários. Na Figura 6.2 está ilustrada a pilha de protocolos necessários para uma aplicação SIP.

É importante destacar que o SIP não é entendido como um protocolo de uso geral. Seu propósito é apenas promover a comunicação, mas a comunicação em si precisa ser suportada por outros protocolos da arquitetura IP (o que chamamos de *suite IP*). Para uma comunicação em tempo real seria necessária a utilização do protocolo RTP¹ sobre o UDP²sobre IP (ver Figura 6.2). Dessa forma, os protocolos adicionais necessários para a transmissão propriamente dita serão oferecidos pelo núcleo (*kernel*) do sistema operacional escolhido, sobre o qual a aplicação SIP residirá.

Dois importantes protocolos para o SIP são o SDP e o RTP. O SDP é utilizado para ajustar a mídia da sessão, e o RTP é usada para o transporte de mídia.

Um problema que deve-se ter em mente com respeito à transporte é a falta de largura de banda para a transmissão de informação. Portanto, a codificação e decodificação desempenham um importante papel em chamadas VoIP. Para isto, são utilizados os *codecs* de áudio e vídeo. A própria natureza da Internet levanta a questão sobre qualidade de serviço (QoS), por fazer uso de uma conexão orientada à pacotes. Cada pacote segue um caminho independente, podendo chegar ao destino fora de ordem e com atrasos variáveis entre chegadas de

¹O RTP (*Real-Time Protocol*) é um protocolo da arquitetura IP usado para *stream* de mídia (áudio e vídeo), onde uma pronta entrega torna-se necessária [3].

²O UDP (*User Datagram Protocol*) é a versão datagrama para a camada de transporte da arquitetura IP. Esta camada é projetada de forma a permitir que pares de entidades dos servidores fonte e destino mantenham uma conversação.

cada pacote. SIP e RTP não implementam QoS, e um outro protocolo deve ser usado para isto. Um protocolo para esta finalidade, que utiliza a mesma pilha e trabalha conjuntamente como o RTP é o RTCP³. Um outro protocolo para esta finalidade é o RSVP⁴.

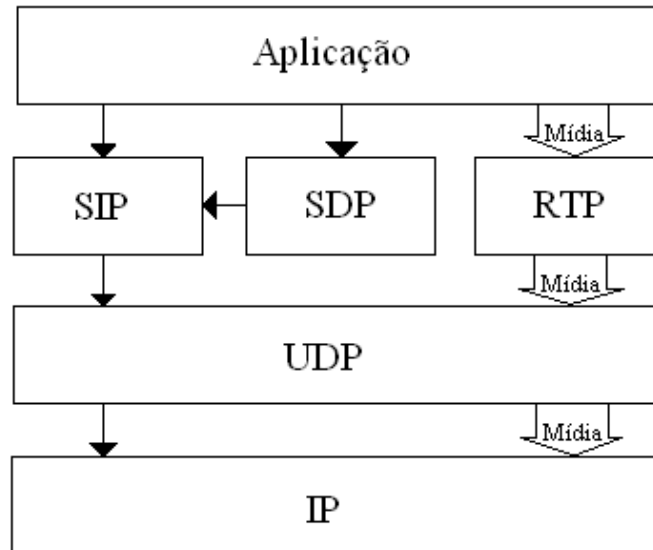


Figura 6.2: Pilha de protocolos de uma aplicação SIP.

6.4 Arquitetura do Protocolo

O SIP, por sua vez, é também estruturado como um protocolo organizado em camadas, o que significa que seu comportamento é descrito em termos de um conjunto de estágios de processamento idealmente independentes, com apenas um fraco acoplamento entre cada estágio. Estas camadas podem ser visualizadas na Figura 6.3. O comportamento do protocolo é descrito como camadas para o propósito de apresentação, permitindo a descrição de funções comuns através de elementos em uma única sessão. Ele não dita nenhuma forma de

³O *Real Time Control Protocol* é um protocolo usado para monitorar a QoS e levar informação sobre participantes de uma sessão formada. Pertence à *suite* IP.

⁴*Reservation Protocol*. É utilizado para reservar recursos da rede, para uma conexão de voz sobre IP, por exemplo.

implementação. Quando dizemos que um elemento contém camadas, entendemos que ele é complacente com um conjunto de regras definidas por aquela camada.

Nem todo elemento especificado pelo protocolo possui todas as camadas. Além disso, os elementos especificados pelo SIP são entidades lógicas, e não físicas.

A camada mais baixa do SIP é sua *camada de sintaxe e codificação*. Sua codificação é especificada utilizando a gramática ABNF⁵[6]. No Capítulo 5 apresenta-se resumidamente a estrutura de mensagens do SIP.

A segunda camada é a *camada de transporte*. Ela define como um cliente envia pedidos e recebe respostas, e como um servidor recebe pedidos e envia respostas sobre a rede. Todos os elementos SIP possuem esta camada. Esta camada se preocupa com a determinação da conexão a ser usada para um pedido ou resposta no caso de transporte orientado à conexão. Todo elemento SIP precisa implementar os protocolos de transporte UDP e TCP.

A terceira camada é a *camada de transação*. Transações são componentes fundamentais do SIP. Uma transação é um pedido enviado pelo cliente transação, utilizando a camada de transporte, para um servidor transação, ao longo da qual todas as respostas para aquele pedido enviadas a partir do servidor transação voltam ao cliente. A camada de transação trata de retransmissões da camada de aplicação, comparando respostas com pedidos, e fim do tempo de vida das mensagens (*timeouts*) no nível de aplicação.

Qualquer tarefa que um *user agent client* (UAC) pode executar é realizada fazendo-se uso de uma série de transações. *User agents* (UA) possuem a camada de transação assim como os servidores *proxy stateful*. Servidores *stateless proxy* não possuem esta camada.

A camada de transação possui um componente cliente, referenciado com cliente transação (*client transaction*), e um componente servidor, chamado servidor transação (*server transaction*). Cada componente é representado por uma máquina de estados finitos que é construída com a finalidade de processar um pedido em particular.

A camada acima da camada de transação é chamada *camada de transação usuário* (*Transaction User* (TU)). Cada uma das entidades SIP, exceto *stateless proxies*, é um transação usuário.

Quando um TU deseja enviar um pedido, ele cria uma instância transação cliente e passa

⁵*Augmented Backus-Naur Format*

para ela o pedido, junto com o endereço IP, a porta, e transporte para o qual enviar o pedido. Um TU que cria um transação cliente pode também cancelá-lo. Quando um cliente cancela uma transação, ele pede que o servidor pare processamentos adicionais, retorna para o estado que existia antes da transação ser iniciada, e gera uma resposta de erro específica para àquela transação. Tudo isto é feito com o pedido CANCEL, o qual constitui uma transação própria, mas referencia a transação a ser cancelada.

Os elementos SIP possuem um núcleo que os diferenciam uns dos outros., Núcleos, exceto para *stateless proxies*, são TUs. Enquanto que o comportamento do núcleo de um UAC e UAS depende do método, existem algumas regras comuns à todos os métodos. Para UAC, estas regras governam a construção de pedidos; para UAS, elas governam o processamento de um pedido e a geração de uma resposta.

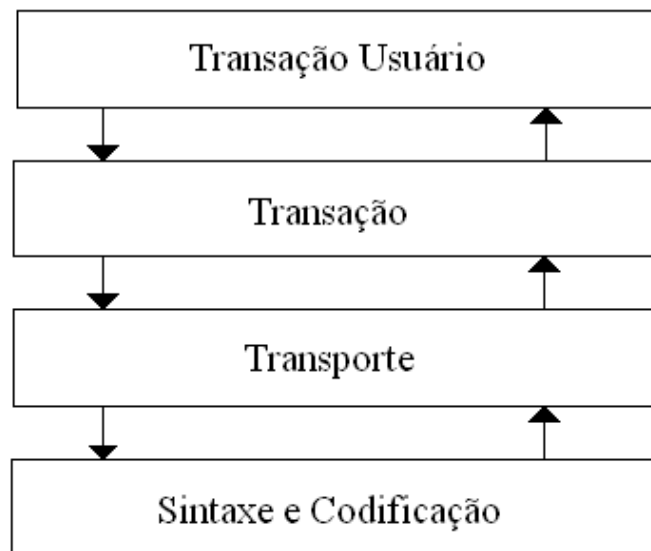


Figura 6.3: Arquitetura do protocolo SIP.

Capítulo 7

Cenários Típicos do SIP

Neste capítulo vamos exemplificar resumidamente três cenários básicos que tipicamente se configuram em um tráfego de mensagens SIP

7.1 Registro

Usuários precisam se cadastrar em um servidor *Registrar* para que possam receber chamadas de outros usuários. Um registro compreende um pedido REGISTER seguido de uma resposta 200 OK enviada pelo *registrar* para o *user agent* que originou o pedido informado que o registro foi realizado com sucesso. Geralmente, registros devem ser autorizados, podendo surgir uma resposta 407 se o usuário não enviar as credenciais válidas, indicando que é necessária a autenticação no servidor para este serviço. Um exemplo de registro é ilustrado na Figura 7.1

7.2 Convite

Um convite de sessão inicializa-se com um pedido INVITE enviado por um *user agent* para um servidor *proxy*. O *proxy* envia imediatamente uma resposta 100 Trying para parar

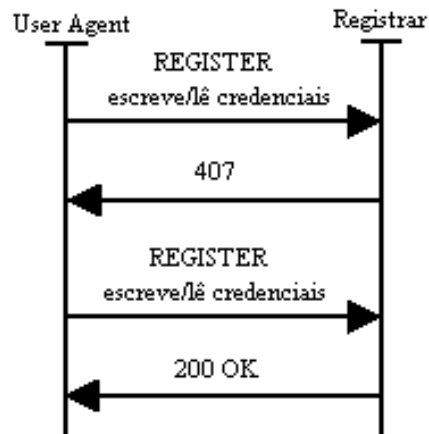


Figura 7.1: Fluxo da mensagem REGISTER.

retransmissões e passar adiante o pedido.

Todas as respostas provisionais enviadas pelo destino final da chamada devem ser enviadas de volta para o originador da chamada. Repare a resposta 180 Ringing no fluxo de chamada da Figura 7.2. Esta resposta é gerada quando o telefone da parte chamada começa a tocar.

Um 200 OK é gerado uma vez que a parte chamada atende o telefone e é retransmitida pelo *user agent* da parte chamada até que esta receba um confirmação ACK do originador da chamada. A sessão é estabelecida neste ponto, e um fluxo de mídia é estabelecido entre os dois *user agents* (comunicação fim-a-fim).

7.3 Finalização de Sessão

A finalização de uma sessão estabelecida é realizada pelo envio do pedido BYE dentro de um diálogo estabelecido pela mensagem INVITE. Mensagens BYE são enviadas diretamente de um *user agent* para outro, ao menos que algum *proxy* no caminho do pedido INVITE tenha indicado que deseja intermediar a conversação fazendo uso do cabeçalho “**Record-Route**”, gravando informações de roteamento.

A parte interessada em finalizar a sessão envia um BYE para a outra parte envolvida

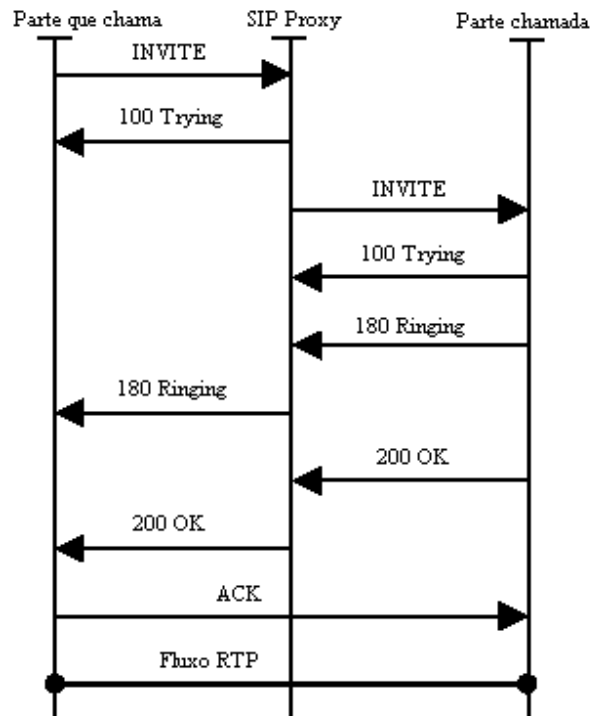


Figura 7.2: Fluxo da mensagem INVITE.

na sessão. A outra parte envia uma resposta 200 OK para confirmar o BYE e a sessão é terminada (ver Figura 7.3).

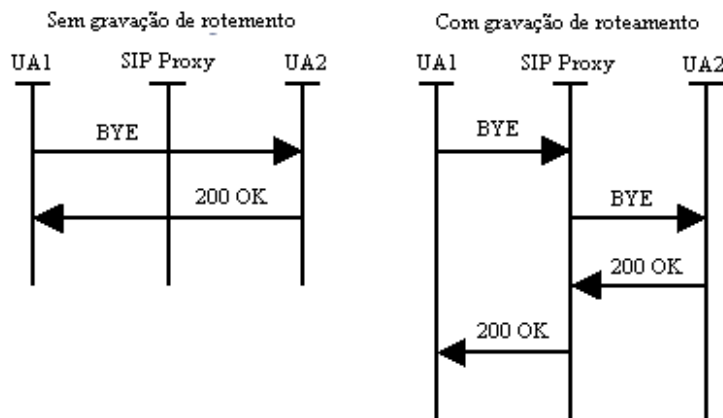


Figura 7.3: Fluxo da mensagem BYE.

Capítulo 8

Conclusão

O SIP é um protocolo de sinalização para estabelecer, modificar e finalizar uma sessão multimídia. Ele é modular, pois não se preocupa com o tratamento da mídia que será utilizada. Ele apenas faz com que os usuários possam se encontrar, estabelecer uma sessão, e utilizar outros protocolos para troca de objetos opacos para o SIP.

A força do SIP pode ser resumida em três características básicas: escalabilidade, extensibilidade e flexibilidade. Estas características estão fazendo com que o SIP ganhe cada vez mais espaço em empresas de desenvolvimento e na indústria, alcançando seu concorrente direto: o protocolo de sinalização H.323 da ITU-T.

A convergência entre os diversos tipos de rede é uma tendência emergente. O protocolo de camada de rede já firmado como padrão universal é o protocolo IP, o que favorece ainda mais o uso do SIP para soluções multimídias fim-a-fim e *multicast*, por ser este facilmente integrado ao modelo Internet e já desenvolvido para trabalhar com a *suite* de protocolos IP.

Um novo paradigma para desenvolvimento de aplicativos está surgindo. Estes não mais serão entendidos como aplicações isoladas, e sim em profunda interação com diversos tipos de dispositivos que utilizam plataformas variadas. Engenheiros e desenvolvedores deverão ter algum conhecimento sobre o protocolo SIP afim de atender esta requisição que está sendo imposta pelos novos usuários de super-serviços.

Aplicações como jogos de RPG, vídeo ao-vivo ou vídeo sob demanda, por exemplo, estarão sempre em contato com aplicações semelhantes de outros usuários compartilhando

a mesma sessão. Assim, os jogos em rede ou as aplicações de vídeo oferecerão conectividade fim-a-fim com um grupo de usuários desejados, e a utilização do SIP será então inevitável.

Propõe-se como trabalhos futuros o levantamento dos aplicativos e implementações SIP, proprietários ou livres, bem como a avaliação de ferramentas e bibliotecas de desenvolvimento voltadas para o SIP. Isto nos dará uma clara noção do mercado hoje existente em torno do SIP, servindo também como preparação para futuras implementações e/ou desenvolvimento de aplicações SIP, tal como uma rede de voz sobre IP, que está em evidência atualmente.

Bibliografia

- [1] RFC 971: “*IP: Internet Protocol*”, Internet Engineering Task Force’s - IETF, Network Working Group, disponível em <http://www.ietf.org.html> .
- [2] RFC 3261: “*SIP: Session Initiation Protocol*”, Internet Engineering Task Force’s - IETF, Network Working Group, disponível em <http://www.ietf.org.html> .
- [3] RFC 1889: “*RTP: A Transport Protocol for Real-Time Applications*”, Internet Engineering Task Force’s - IETF, Network Working Group, disponível em <http://www.ietf.org.html> .
- [4] RFC 2326: “*RTSP: Real-Time Streaming Protocol (RTSP)*”, Internet Engineering Task Force’s - IETF, Network Working Group, disponível em <http://www.ietf.org.html> .
- [5] RFC 3015: “*Megaco Protocol Version 1.0*”, Internet Engineering Task Force’s - IETF, Network Working Group, disponível em <http://www.ietf.org.html> .
- [6] RFC 2234: “*Augmented BNF for Syntax Specifications: ABNF*”, Internet Engineering Task Force’s - IETF, Network Working Group, disponível em <http://www.ietf.org.html> .
- [7] TERENA (2004): “*IP Telephony Cookbook*”, TERENA Report.
- [8] Edwards, L., Barker, R. (2004) : “*Nokia Mobile Developer Series - Developing Series 60 Applications: A Guide for Symbian OS C++ Developers*”, Addison-Wesley, New York, USA.
- [9] Khasnabish, B. (2003): “*Implementing Voice over IP*”, John Wiley & Sons, Inc., New Jersey, USA.

- [10] Hollabaugh, C. (2002): “*Embedded Linux*”, Addison-Wesley, New York, USA.
- [11] Alencar, M. S.(1998): “*Telefonia Digital*”, Érica, São Paulo, Brasil.
- [12] Tanenbaum, A. S.(1996): “*Computer Networks*”, Prentice-Hall, Inc., New Jersey, USA.
- [13] Bellamy, J.(1991): “*Digital Telephony*”, John Wiley & Sons, Inc., New York, USA.
- [14] H.323 Recommendation, Packet-Based Multimedia Communications Systems, ITU-T, Geneva, 1996, 1998, 2000.
- [15] Jan Janak, “*SIP Introduction*”, disponível em http://www.iptel.org/ser/doc/sip_introduction.pdf .