4
Desenvolvimento de uma Solução de Gerenciamento de Mobilidade na Camada de Aplicação com SIP

Este capítulo apresenta a implementação da solução de Gerenciamento de Mobilidade na camada de aplicação através do protocolo SIP, tal como descrita na Seção 3.5. É chamada de SIP User Agent (ou SIP UA), uma API reutilizável que implementa um User Agent SIP que provê os aspectos necessários para esta solução de Gerenciamento de Mobilidade. Esta API esconde os detalhes e a complexidade contidos na manipulação do protocolo SIP, tornando fácil sua utilização por qualquer aplicação que necessite de uma solução como esta para suportar a mobilidade de dispositivos. As principais entidades e módulos da API SIP User Agent são apresentados a seguir.

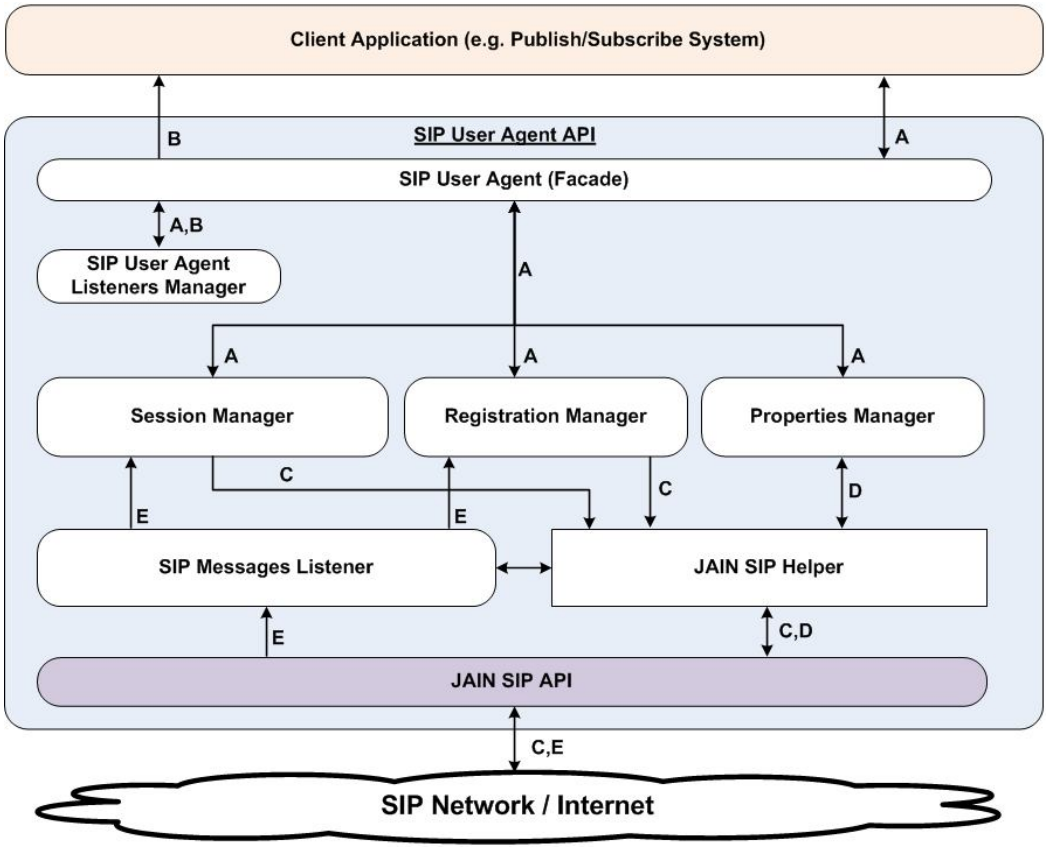

Figura 17 – Módulos da API SIP User Agent

SIP User Agent. É a fachada principal da API. Ela oferece para a aplicação acesso às principais funcionalidades do SIP User Agent. De acordo com o método

chamado pela aplicação (mostrado na Figura 18), a fachada chama o módulo responsável por seu processamento (fluxos A na Figura 17).

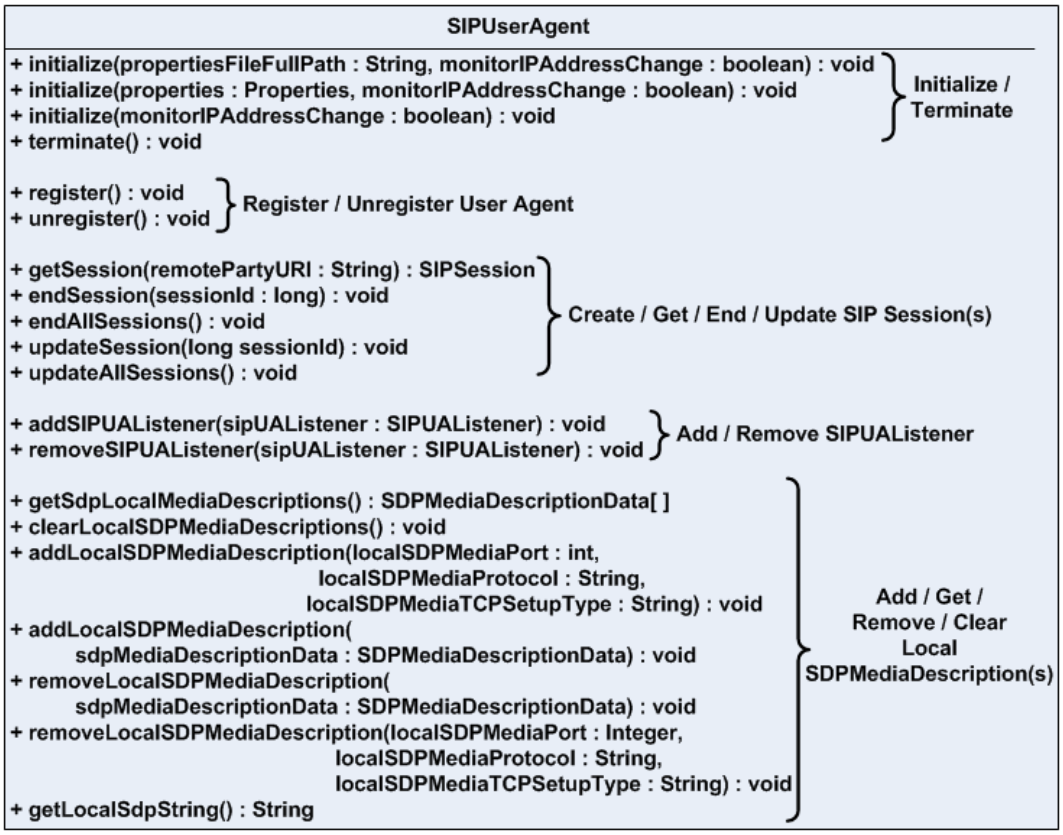

Figura 18 – Principais métodos da fachada SIPUserAgent

A fachada contém métodos para:

- Inicialização e término da API, através dos quais a aplicação passa todos os parâmetros necessários para a execução do SIP UA. As chamadas a estes métodos são encaminhadas para o módulo PropertiesManager.
- Adição ou remoção de um registro de um usuário em um Registrar. As chamadas a estes métodos são encaminhadas para o módulo RegistrationManager.
- Criação, término ou atualização de Sessões SIP com outros User Agents (a classe SIPSession que representa uma Sessão SIP é explicada mais adiante nesta Seção). As chamadas a estes métodos são encaminhadas para o módulo SessionManager.
- Adição ou remoção de objetos Listeners da aplicação (que são explicados a seguir). As chamadas a estes métodos são encaminhadas para o módulo SIPUAListenerManager.

- Adição, obtenção e remoção de descrições locais de mídia, especificadas em SDP (as classes que representam descrições de mídia são explicadas mais adiante nesta Seção). As chamadas a estes métodos são encaminhadas para o módulo PropertiesManager.

SIPUAListenerManager. Como o SIP User Agent se trata de uma API que funciona com um protocolo que envia e recebe mensagens assincronamente, o módulo SIPUAListenerManager implementa o Padrão de Projeto Observador (Gamma, Helm et al., 1995), oferecendo métodos para o registro de objetos observadores (também conhecidos como Listeners) que são utilizados para enviar para a aplicação os principais eventos de execução2. A aplicação deve registrar objetos que implementem a interface SIPUAListener (mostrada na Figura 19), que as torna capazes de receber eventos de execução do SIP User Agent. O SIPUAListenerManager é o responsável por gerenciar todos os objetos Listeners que foram registrados pela aplicação. Quando algum módulo do SIP User Agent deseja notificar a aplicação sobre a ocorrência de algum evento de execução, ele chama este módulo para que ele notifique todos os Listeners registrados (fluxos B na Figura 18).

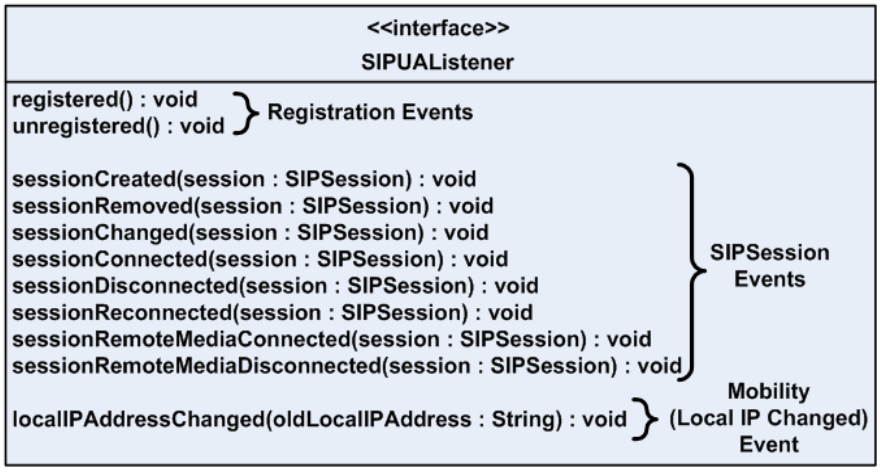

Figura 19 – Principais métodos da interface SIPUAListener

Desta forma a aplicação pode ser notificada quando um dos seguintes eventos de execução ocorre dentro do SIP User Agent:

- Registro ou remoção de um usuário em um Registrar.

[2] O termo *evento de execução* se refere à ocorrência de alguma situação na execução definida programaticamente como relevante para a aplicação. Este termo é utilizado para a diferenciação do termo *evento* utilizado nos sistemas publish/subscribe.

- Criação, remoção ou mudança de uma Sessão SIP.
- Mudanças no estado da comunicação de uma Sessão SIP, tais como quando uma Sessão é conectada, desconectada, re-conectada, ou quando ocorre uma conexão ou desconexão de mídia de comunicação.
- Quando o endereço IP local do dispositivo muda, possivelmente devido à mobilidade.

SessionManager. É o responsável por gerenciar as Sessões SIP. Ele envia requisições e recebe respostas SIP para o estabelecimento, atualização e término destas Sessões. Por exemplo, quando a aplicação solicita o estabelecimento de uma Sessão com outro User Agent remoto, este módulo realiza o envio e recebimento de todas as mensagens necessárias, tais como INVITE, OK, ACK e etc. Para a troca de mensagens SIP, este módulo utiliza o módulo JAINSIPHelper, que é explicado a seguir nesta Seção.

RegistrationManager. O RegistrationManager é responsável por gerenciar o registro do usuário em um Registrar. Ele envia mensagens REGISTER para um Registrar com o identificador independente de localização do User Agent e o endereço IP corrente do dispositivo. Ele contém também uma thread que envia periodicamente atualizações do registro para o Registrar com o endereço IP corrente do dispositivo. Estas atualizações são enviadas com uma taxa de um terço do tempo especificado da expiração do registro no Servidor SIP, ou então, são imediatamente disparadas no caso da detecção de uma mudança no endereço IP do dispositivo.

PropertiesManager. É o módulo responsável por gerenciar todas as propriedades necessárias para instanciar o SIP User Agent e sua configuração. Este módulo também armazena propriedades relacionadas à utilização do protocolo SIP e especificamente para a JAIN SIP API (dentro do módulo JAINSIPHelper). O módulo oferece também facilidades para carregar propriedades de um arquivo de propriedades, de um objeto Properties (da linguagem Java) ou diretamente das propriedades do sistema, definidas anteriormente.

JAIN SIP API. A JAIN SIP (Java API for SIP Signalling) é uma API Java projetada pelo NIST (National Institute of Standards and Technology) para oferecer aos desenvolvedores de aplicações Java a capacidade de utilizar o protocolo SIP. Ela oferece a manipulação de cada detalhe deste protocolo,

fornecendo meios para a construção de User Agents SIP de forma totalmente personalizada. Por exemplo, com esta API, pode-se definir o formato das mensagens, métodos de envio e tratadores para mensagens recebidas em um User Agent, assim como as respostas a serem enviadas para cada tipo de mensagem recebida. Dentre os principais componentes da JAIN SIP API podemos destacar (O'doherty e Ranganathan, 2003):

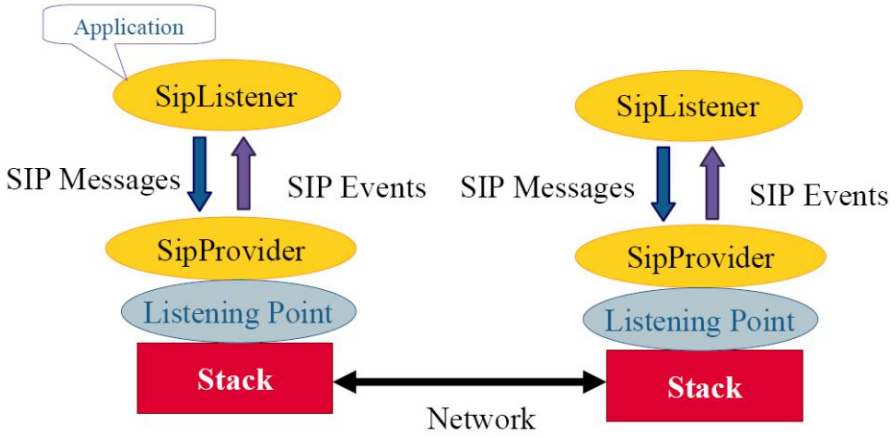

Figura 20 – Principais componentes da API JAIN SIP (O'doherty e Ranganathan, 2003)

- SipFactory, AddressFactory, HeaderFactory e MessageFactory. São fábricas para a criação dos diversos objetos da API. Elas retornam objetos que implementam interfaces padrão para as principais entidades e mecanismos do protocolo SIP.
- SipStack. É a responsável final por enviar e receber mensagens SIP pela rede. É utilizada para criar objetos ListeningPoints e SipProviders.
- ListeningPoint. Representa um ponto de recepção de mensagens, encapsulando um endereço e porta para escutar mensagens, com um determinado tipo de protocolo, por exemplo, 139.82.24.239 UDP/5060.
- SipProvider. Contém métodos para enviar mensagens SIP. Através dela, pode-se registrar um objeto que Listener (i.e. que implementa a interface SIPListener, explicada a seguir) para escutar mensagens SIP recebidas.
- SipListener. Um objeto que implementa esta interface e se registra em um SipProvider é notificado quando mensagens SIP são recebidas.

JAINSIPHelper. O nível de abstração da API JAIN SIP não é suficiente para tornar rápida e fácil sua utilização. Para facilitar a utilização da JAIN SIP, foi desenvolvida uma abstração das principais operações que podem ser realizadas com a API, permitindo que os principais parâmetros de cada operação sejam

especificados e a JAINSIPHelper interaja com a API JAIN SIP de forma transparente. Desta forma, as dificuldades de criar mensagens SIP e realizar operações específicas da JAIN SIP são encapsuladas, facilitando o desenvolvimento do sistema e sua manutenção. Portanto, todos os módulos do SIP User Agent utilizam a JAINSIPHelper para realizar suas funções. Por exemplo, quando o SessionManager precisa enviar uma requisição (e.g. INVITE) para estabelecer uma sessão com outro User Agent, ou quando o RegistrationManager precisa enviar uma requisição REGISTER, para registrar o User Agent em um Registrar (fluxos C na **Figura 17**), estes módulos podem chamar os métodos do JAINSIPHelper passando os parâmetros necessários e este módulo se encarrega de construir a mensagem SIP e enviá-la utilizando a API JAIN SIP. O PropertiesManager também utiliza o JAINSIPHelper (fluxos D na **Figura 17**) para a conversão e validação de dados (e.g. URIs de usuários), também realizados em última instancia pela API JAIN SIP.

SIPMessageListener. Este módulo do sistema implementa a interface SIPListener da JAIN SIP API, e é adicionado como um Listener a um SIPProvider para receber mensagens SIP vindas da camada inferior. Ele processa mensagens SIP que chegam e as repassa para o módulo apropriado para seu processamento. Por exemplo, requisições ou respostas para estabelecimento de Sessões (e.g. INVITE, OK, ACK e etc) são passadas para o módulo SessionManager, e requisições ou respostas relacionadas ao registro do usuário em um Registrar (e.g. REGISTER, OK e etc) são passadas para o RegistrationManager (fluxos E na **Figura 17**).

Servidores SIP. Além de um User Agent, para a utilização do protocolo SIP, são necessários servidores que façam o papel das principais entidades da rede SIP, tais como SIP Proxy, Registrar e Location Service. O SER (SIP Express Router) é uma implementação de código aberto de um servidor SIP que faz papel de todas as entidades necessárias, e que inclusive oferece diversas outras funcionalidades tais como autenticação de usuários. Esta implementação é bem consolidada e difundida pelos usuários do protocolo SIP. Outra implementação de servidor SIP é o (SIP Presence Proxy), desenvolvido pelos mesmos criadores da API JAIN SIP. Esta implementação oferece uma ferramenta para visualização das mensagens SIP que trafegam pelo Proxy e que é muito útil para a depuração do protocolo.

Descrição SDP Local. Como foi explicado na Seção 2.1, o protocolo SIP é utilizado para sinalização, e é independente da mídia (troca de dados, voz, etc) propriamente dita sendo utilizada pela aplicação. Quando uma Sessão SIP é estabelecida, cada participante utiliza a especificação SDP do outro participante para enviar dados. Portanto, todo User Agent deve ser configurado com uma descrição local em SDP das mídias de comunicação que deve utilizar, para que possa apresentar esta descrição para os outros participantes das Sessões SIP que vier a estabelecer. O SIP User Agent oferece, portanto, a capacidade da aplicação configurar a descrição SDP local do SIP User Agent com qualquer número desejável mídias a serem utilizadas. Além disso, para que a aplicação não necessite manipular o protocolo SDP, o SIP User Agent oferece facilidades para que a aplicação só tenha que especificar os parâmetros de cada mídia (tais como endereços IP, números de portas e protocolos comunicação), e a descrição SDP local do User Agent seja automaticamente gerada dentro das Sessões que são estabelecidas. Estes parâmetros podem ser especificados em uma das propriedades de inicialização do SIP User Agent. Entretanto, pode ser necessário para a aplicação alterar a descrição SDP local do User Agent após a inicialização. Por exemplo, caso deseje alterar o número de uma porta de comunicação. Para isto, o SIP User Agent oferece também métodos que permitem a adição, obtenção ou remoção de dados de mídia (objetos do tipo SDPMediaDescriptionData) na descrição SDP local utilizada pelo SIP UA (último grupo de métodos na **Figura 18**).

Sessões de Comunicação. A principal entidade utilizada para o gerenciamento da interação entre os User Agents é chamada de SIPSession (Sessão SIP), que representa uma Sessão de Comunicação SIP, que é um vínculo lógico entre User Agents. Cada Sessão contém os seguintes dados:

- O identificador independente de localização (URI) do usuário a que ela pertence.
- Dados referentes à comunicação SIP, tais como IP, porta e protocolo de transporte (TCP ou UDP) do SIP User Agent do usuário utiliza para enviar e receber mensagens SIP.
- Dados referentes à troca de dados propriamente dita, tais como IP, porta e protocolos de transporte (TCP ou UDP) que o dispositivo do usuário está

utilizando para enviar e receber dados (o termo mídia é utilizado para se referir à troca de dados efetiva entre os dois participantes de uma Sessão).

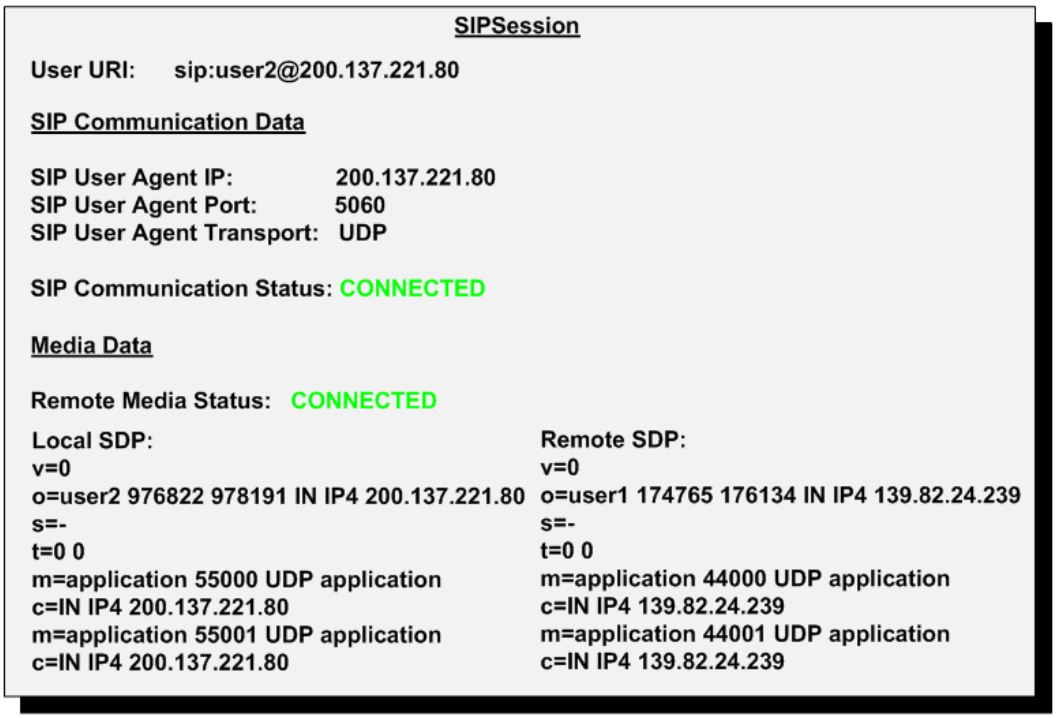

Figura 21 – Exemplo de dados de comunicação SIP (SIP Communication Data) e dados de mídia (Media Data) de uma Sessão SIP

Tal distinção entre dados de comunicação SIP com os dados de mídia é necessária, pois tal como foi explicado na Seção 2.1, o protocolo SIP é apenas um protocolo de sinalização em que a mídia de comunicação é independente da troca de mensagens de sinalização do protocolo. Os dados de mídia deverão conter a descrição em SDP pertence ao User Agent local e a descrição SDP remota (i.e. pertencente ao outro participante da Sessão).

Um dispositivo pode estar conectado para o protocolo SIP, mas não ter uma mídia conectada para trocar dados no endereço especificado dentro da descrição da Sessão em SDP. Logo, para monitorar o estado das Sessões, são necessários dois conjuntos de estados, um indicando se a Sessão SIP está válida e outro indicando se a mídia para troca de dados está válida.

A tabela abaixo mostra os possíveis estados para os dois indicadores contidos dentro de uma SIPSession:

SIP Communication Status	**UNDEFINED, DISCONNECTED, INVITING, RINGING_THERE, BUSY_THERE, RINGING_HERE, REINVITING, CONNECTED**
Remote Media Status	**DISCONNECTED, CONNECTED**

Tabela 1 – Possíveis estados para os dois indicadores contidos dentro de uma Sessão SIP

O indicador SIP Communication Status contém os estados da Sessão SIP estabelecida pelo SIP User Agent. Por exemplo, quando o SIP User Agent envia uma mensagem INVITE para o SIP User Agent do outro lado da comunicação convidando o outro participante para o estabelecimento da Sessão (veja **Figura 22**), o estado da Sessão permanece INVITING até que algum tipo de resposta seja recebido.

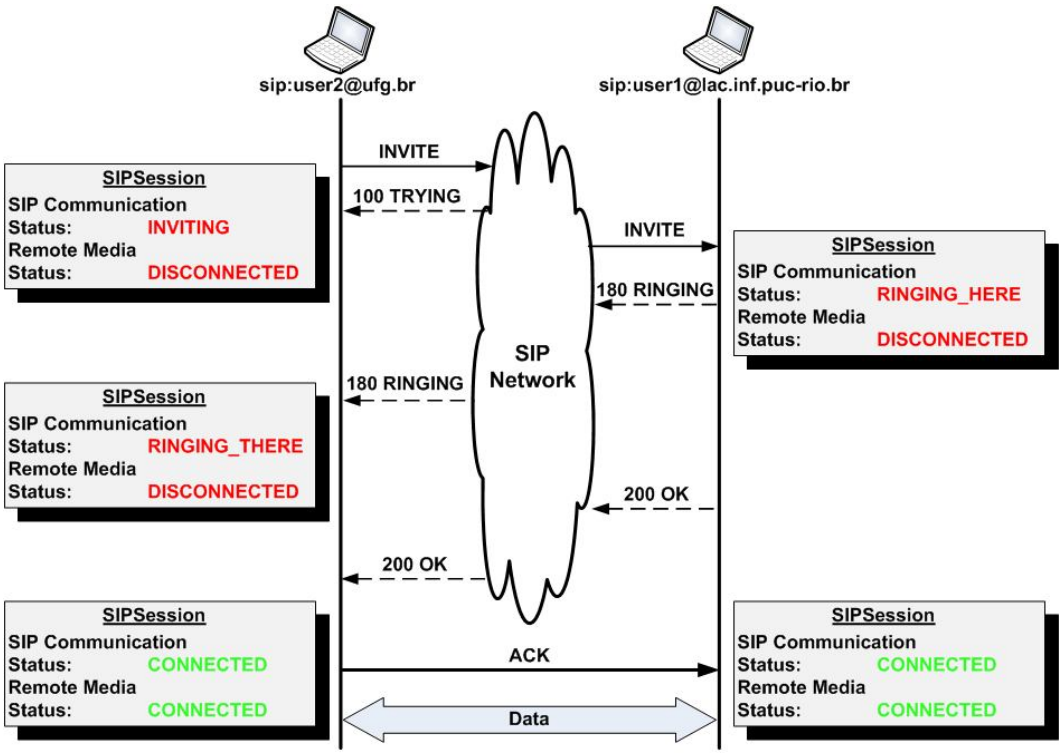

Figura 22 – Estados de uma Sessão SIP durante o estabelecimento da mesma.

Por exemplo, uma resposta RINGING mudaria o estado da Sessão para RINGING_THERE ou uma resposta BUSY mudaria o estado para BUSY_THERE. O User Agent do outro participante, ao receber o INVITE, tem uma nova Sessão SIP criada com o estado UNDEFINED e assim que envia a resposta RINGING seu estado passa para RINGING_HERE. Como o SIP User Agent tem como comportamento padrão automaticamente aceitar as sessões de comunicação de outros participantes, logo após de enviar a mensagem RINGING o SIP User Agent aceita a Sessão enviando também uma mensagem OK para o User Agent que iniciou a interação. Quando este receber uma mensagem ACK para o OK enviado, as Sessões SIP em ambos os lados estarão com o estado CONNECTED.

O indicador Remote Media Status contém o estados da comunicação propriamente dita, ou seja, se dados podem ou não ser enviados para o outro

participante da comunicação. Este indicador serve como auxiliar para o monitoramento do comportamento das Sessões. Por exemplo, se a aplicação não é bem sucedida na transmissão de dados através de uma Sessão (e.g. no caso de uma conexão TCP, ocorre uma falha no envio, ou no caso de uma conexão UDP, uma mensagem de confirmação enviada pelo outro participante da sessão não é recebida). Para indicar tal erro, a aplicação atribuiria a Media Status o estado DISCONNECTED, informando que não foi possível realizar o envio de dados. Não faria sentido atribuir o estado DISCONNECTED ao indicador SIP Communication Status, pois a comunicação SIP poderia ainda estar funcionando normalmente através de uma Sessão SIP estabelecida. Isto indicaria que apesar da Sessão SIP ter sido estabelecida, algum erro na outra ponta da comunicação está impedindo o recebimento de mensagens no IP e na porta especificada na descrição SDP da Sessão.

Sessões SIP podem ser atualizadas por ambos os participantes da comunicação. Por exemplo, quando um dispositivo muda algum dado em sua descrição SDP local, ele deve atualizar as Sessões que possui para que os outros participantes recebam estas atualizações (os demais participantes terão a descrição SDP remota atualizada). Para atualizar uma Sessão, o User Agent envia um RE-INVITE para o outro participante atualizando a Sessão e seus estados. Um RE-INVITE, contém o mesmo identificador da sessão indicando que a Sessão deverá ser atualizada. Este mecanismo é utilizado para o suporte à mobilidade de terminais tal como descrito na Seção 3.5, e como ilustrado na Seção 5.2, para suportar a mobilidade de terminais no sistema publish/subscribe utilizado como estudo de caso para a validação desta solução.