2
Fundamentação Conceitual

Este capítulo apresenta alguns conceitos importantes que são utilizados ao longo do trabalho. Primeiramente, é apresentado o Session Initiation Protocol (SIP) (Rosenberg, Schulzrinne et al., 2002), que serviu de base para implementar a solução de Gerenciamento de Mobilidade na camada de aplicação (explicada em mais detalhes no Capítulo 3). Em seguida, os sistemas publish/subscribe são apresentados juntamente com seu paradigma de comunicação e seus principais elementos.

2.1.
Session Initiation Protocol (SIP)

O SIP é um protocolo de camada de aplicação utilizado para criar, modificar e terminar sessões entre um ou mais participantes. Ele foi projetado para ser independente do protocolo em nível de transporte utilizado, uma vez que ele é apenas um protocolo de sinalização para o estabelecimento e manutenção de sessões. Uma sessão nada mais é do que um vínculo lógico entre duas ou mais entidades envolvidas em uma interação. Uma vez estabelecida a sessão, ocorre então uma troca ou fluxo de dados entre os participantes, que é totalmente independente do SIP, ou seja, pode ser uma troca de dados via TCP, UDP ou qualquer outro protocolo desejado.

Para que os usuários do protocolo SIP possam se encontrar mutuamente e trocar mensagens para o estabelecimento de sessões de comunicação através da Internet, o SIP define como identificador de um usuário um URI (Uniform Resource Locator), um identificador único e independente de localização (i.e. independente do ponto de anexação com a rede), que é utilizado ao invés da utilização do endereço IP do dispositivo que o usuário está utilizando. O URI utilizado no protocolo SIP é semelhante a um endereço de e-mail, tal como “sip: usuario@dominio”.

Uma rede SIP possui cinco entidades lógicas, cada uma com suas funções específicas. Essas entidades em geral estão co-localizadas, ou seja, o Proxy Server, o Registrar, o Redirect Server e o Location Server são embutidos no mesmo programa servidor1.

User Agent: Um User Agent (UA) é o elemento que executa em cada um dos clientes participantes da interação e é onde uma sessão é iniciada e terminada. Um UA é uma entidade que contém tanto o User Agent Client (UAC) como o User Agent Server (UAS). O User Agent Client é um componente cliente que inicia requisições SIP. O User Agent Server (UAS) é um componente servidor que trata requisições SIP e retorna respostas para requisições recebidas. Um User Agent pode ser, por exemplo, uma aplicação de VoIP instalada no computador de um usuário final. Quando o usuário realiza uma ligação, o UAC desta aplicação é utilizado para iniciar a chamada enviando uma requisição INVITE (explicada mais adiante nesta Seção) para o destino especificado. Quando o dispositivo (por exemplo, um smartphone com VoIP) do usuário chamado recebe uma ligação (através de uma requisição INVITE, do SIP), o UAS desta aplicação retorna uma resposta RINGING para o chamador, indicando que o dispositivo do usuário chamado está tocando.

Location Service: É um serviço de localização que é utilizado para obter informações sobre a localização de usuários, i.e. de seus respectivos dispositivos. Ele contém uma base de dados com o mapeamento do URI de cada usuário registrado para os endereços de rede de um ou mais User Agents que um usuário pode estar utilizando. Estes mapeamentos de URIs para endereços de rede podem ser inseridos ou removidos de diferentes formas. Uma delas é através de mensagens SIP do tipo REGISTER que serão explicadas a seguir. Desta forma, um usuário pode ser localizado independente do endereço de rede do User Agent que ele está utilizando no momento.

Proxy Server: Um Proxy Server implementa um dos papéis do Servidor SIP, que pode atuar tanto como um servidor como um cliente de mensagens SIP. O Proxy Server recebe requisições de clientes e consulta o Location Service para determinar a localização dos User Agents dos usuários destinatários das

[1] Quando o termo Servidor SIP for utilizado no texto, ele estará se referindo a um servidor capaz de desempenhar todos os papéis acima relacionados.

requisições. Caso o usuário destinatário de uma requisição esteja registrado em seu Location Service, o Proxy Server repassa a requisição diretamente para o User Agent deste usuário. Caso contrário, o Proxy repassa a requisição adiante para o servidor SIP no domínio contido no URI do usuário destinatário (possivelmente utilizando o DNS, caso necessário). Um Proxy Server utilizado para repassar mensagens entre User Agents, ou entre Servidores SIP, participa de toda a interação até que os User Agents dos participantes estabeleçam uma sessão.

Redirect Server: Um Redirect Server é um servidor que aceita requisições SIP e utiliza o Location Service para mapear o URI contido na requisição para os endereços de rede dos usuários e os retornar para o cliente. Quando o usuário com o URI em questão não está registrado naquele servidor (i.e. não é encontrado um mapeamento para o URI recebido) o Redirect Server envia de volta para o cliente um mapeamento do URI com um endereço de rede vazio. Diferentemente dos Proxy Servers, os Redirect Servers não repassam a mensagem da requisição para outros Servidores SIP, mas apenas envia de volta para um UA requisitante. Isso permite a construção de servidores mais escaláveis, já que ao invés de participar de toda a transação, estes têm apenas que enviar de volta uma resposta com a localização. Ambos o Redirect Server e o Proxy Server aceitam registros de usuários (i.e. ambos possuem Registrars, explicados a seguir).

Registrar: O Registrar é um servidor que aceita mensagens do tipo REGISTER de User Agents. Tais mensagens contêm o identificador independente de localização do usuário (URI) em questão e o endereço de rede do User Agent que o usuário está utilizando. Quando o Registrar recebe a mensagem REGISTER, ele utiliza o Location Service para que este atualize o banco de dados de localizações com o URI e os endereços reais do usuário. Geralmente os servidores Registrar tratam também da autenticação e autorização de usuários.

No SIP uma interação entre participantes é considerada como uma transação, e um conjunto de transações forma uma sessão. As sessões possuem um identificador e as transações utilizam este identificador para indicar a qual sessão elas pertencem.

O SIP é composto por vários tipos de mensagens. Estes tipos de mensagens são divididos em requisições e respostas. Dentre os tipos de requisições podem ser destacados o REGISTER para o registro de um usuário no Location Service, o INVITE para o início de uma sessão de comunicação, o ACK para confirmação de

início de uma sessão, o CANCEL para o cancelamento de uma requisição e BYE para o término de uma sessão. As respostas são classificadas nas seguintes categorias:

- 1xx: Provisional – Indica que a requisição foi recebida, e seu processamento está em andamento. Por exemplo, quando uma requisição INVITE é recebida, uma resposta provisional 180 Ringing pode ser enviada para indicar que o User Agent do usuário chamado o está avisando sobre o recebimento da chamada (e.g. tocando um aviso sonoro).
- 2xx: Sucesso – Indica que a requisição foi recebida, entendida, aceita e processada com sucesso. Por exemplo, quando um usuário aceita uma chamada, o User Agent do mesmo envia uma resposta 200 OK para o User Agent do usuário chamador.
- 3xx: Redireção – Indica que ações adicionais necessitam ser tomadas para que o processamento da requisição possa ser completado. Por exemplo, quando um Redirect Server recebe uma requisição, mas verifica em seu Location Service que o usuário destinatário está registrado em outro Registrar, envia uma resposta 301 Moved Permanently ou 302 Moved Temporarily para o User Agent do usuário chamador.
- 4xx: Erro no Cliente – Indica que a requisição contém erro de sintaxe ou não pode ser atendida pelo servidor. Por exemplo, quando um servidor SIP recebe uma requisição destinada a um usuário que deveria estar registrado em seu Location Service (pois o URI do usuário corresponde ao domínio daquele servidor SIP), mas o usuário não é encontrado, uma resposta 404 Not Found é retornada para o User Agent chamador.
- 5xx: Erro no Servidor – Indica que o servidor falhou em completar uma requisição válida. Por exemplo, o servidor envia uma resposta 500 Server Internal Error, quando algum erro interno ocorre durante o processamento de uma requisição
- 6xx: Falha Global – Indica que a requisição não pode ser completada em qualquer servidor. Por exemplo, quando um usuário rejeita uma chamada, uma resposta 603 Decline é enviada para o User Agent do usuário chamador.

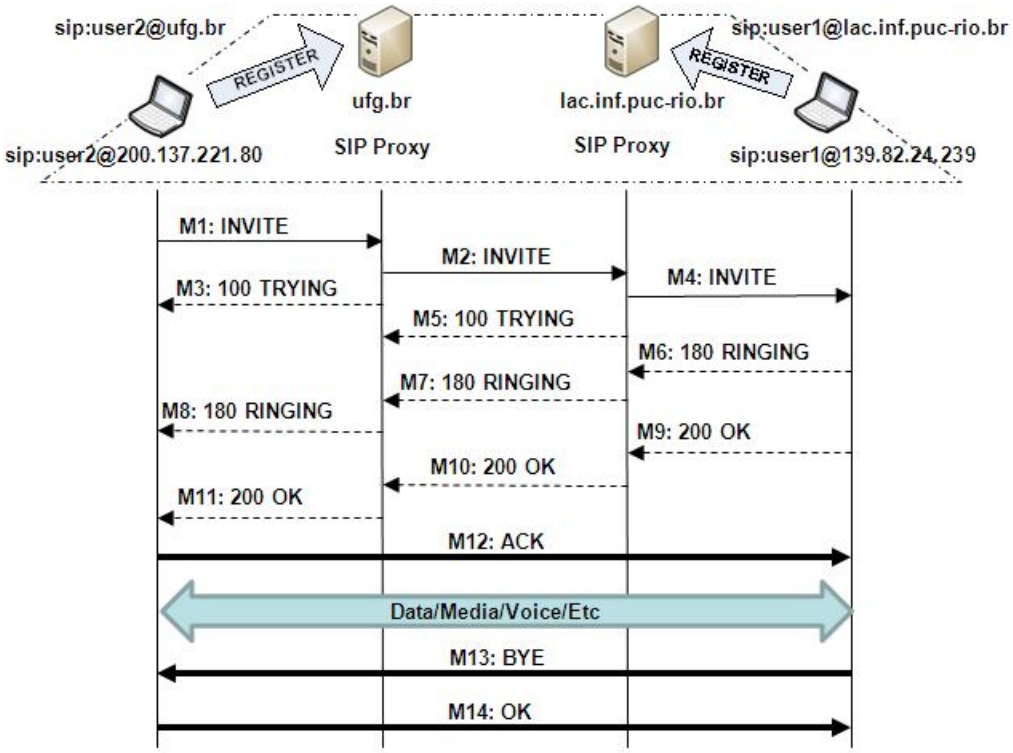

Figura 1 – Exemplo clássico de estabelecimento de sessão “Trapezóide SIP” (Rosenberg, Schulzrinne et al., 2002)

Na Figura 1, pode-se observar um exemplo clássico de estabelecimento de uma sessão SIP. Este exemplo básico também é conhecido com “Trapezóide SIP”, referindo-se às linhas pontilhadas em forma de trapezóide contido no desenho. Neste exemplo, um usuário User2 com o URI “sip:user2@ufg.br” se registra no Servidor SIP do domínio “ufg.br” indicando o endereço de rede do User Agent que está utilizando “sip:user2@200.137.221.80”. Outro usuário, User1, com o URI “sip:user1@lac.inf.puc-rio.br” se registra no Servidor SIP do domínio “lac.inf.puc-rio.br” indicando o endereço de rede do User Agent, “sip:user1@139.82.24.239”. No exemplo, assumimos que o usuário User2 deseja se comunicar com o usuário User1. As mensagens trocadas entre o UA do User2, o Proxy Server do domínio “ufg.br”, o Proxy Server do domínio “lac.inf.puc-rio.br” e o UA do usuário User1 são mostradas na Figura e contém uma numeração indicando sua ordem de execução (Rosenberg, Schulzrinne et al., 2002).

Neste exemplo, as requisições do usuário são enviadas para o Proxy Server de saída do UA, pertencente ao User2, que as encaminha utilizando DNS para o domínio contido no URI do usuário a ser chamado. O Proxy Server do domínio “lac.inf.puc-rio.br” recebe a requisição e verifica em seu Location Service o

endereço corrente onde o User Agent do usuário User1 está e encaminha a requisição para o UA do User1. O User1 poderia estar localizado em um outro endereço. Nesse caso, naturalmente ele teria atualizado seu registro no Location Service do Proxy Server e seria localizado da mesma forma.

Os User Agents dos usuários enviam dados embutidos nas mensagens SIP, expressos através do protocolo SDP (Session Description Protocol), que permitem uma negociação sobre as características da sessão a ser estabelecida. A requisição INVITE (enviada pelo UserAgent do usuário chamador) contém a descrição da sessão especificada pelo usuário chamador e a resposta OK (retornada pelo UserAgent do usuário chamado) contêm a descrição da sessão especificada pelo usuário chamado (Handley e Jacobson, 2006).

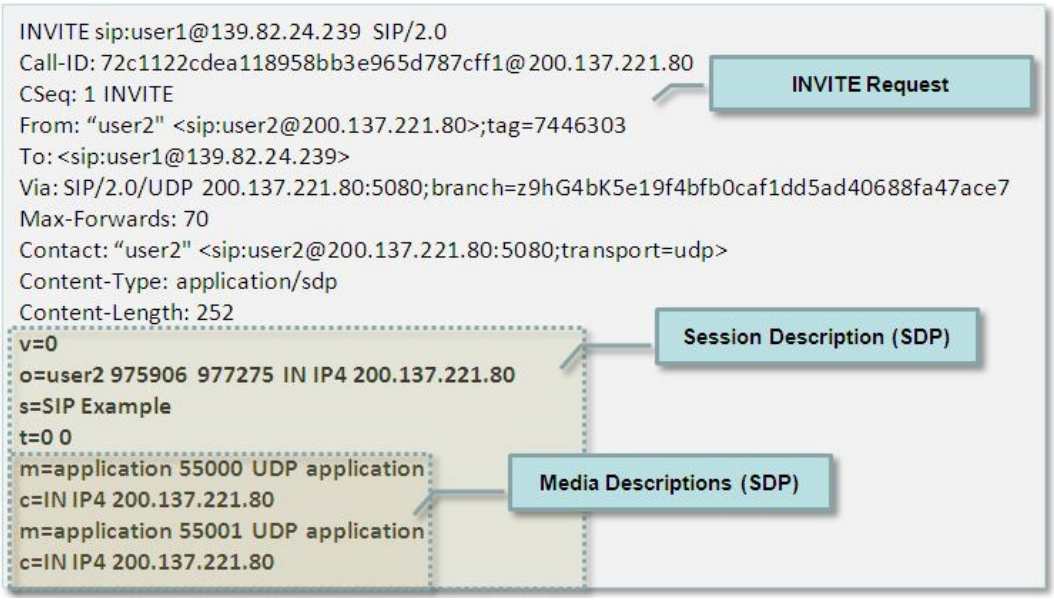

Figura 2 – Exemplo de Requisição INVITE com descrição SDP

Esta descrição permite que informações sobre a sessão, tais como os dados das mídias utilizadas (e.g. endereços IP, tipos de protocolo de comunicação, números de portas, dentre outros), e que as capacidades dos dispositivos sejam trocadas entre os participantes da sessão. Tais descrições SDP são transportadas em mensagens SIP, de uma forma análoga a como código HTML de uma página web é embutida em uma mensagem HTTP.

A Figura 2 mostra um exemplo representando a requisição INVITE enviada pelo User Agent do usuário User2 para o Servidor SIP do domínio "ufg.br" (mensagem M1 na **Figura 1**), com a descrição SDP embutida na mensagem. A resposta OK retornada pelo User Agent do usuário User1 (mensagem M9 na

Figura 1) carrega, no sentido oposto, a descrição SDP do usuário User1, que é encaminhada para o User Agent do usuário User2 (Wedlund e Schulzrinne, 1999).

Os endereços utilizados para a troca de dados entre os participantes da sessão são os endereços indicados na parte SDP da mensagem, e portanto são diferentes daqueles usados para o estabelecimento de sessões e envio/recebimento de mensagens SIP. Em uma mesma sessão podem ser especificadas várias mídias a serem utilizadas. Cada descrição de uma mídia inicia com um parâmetro do tipo "m=<tipo de mídia> <porta> <protocolo> <formato>" e pode ser complementado pelo parâmetro “c=<tipo da rede> <tipo do endereço> <endereço de conexão>" e atributos do tipo “a=<nome atributo>:<valor>”. Mais detalhes sobre cada campo e seus atributos podem ser encontrados em (Handley e Jacobson, 2006) e (Yon e Camarillo, 2005).

No exemplo da **Figura 2**, o usuário User2 está especificando que deseja utilizar o protocolo UDP para trocar dados no IP 200.137.221.80 e porta 55000 e também no IP 200.137.221.80 e porta 55001 (campos m e c na parte de descrições de mídias na figura).

2.2. Utilização do Protocolo SIP através de redes com NAT (Network Address Translation)

Hoje em dia, na Internet, uma grande quantidade de dispositivos não possui um endereço IP público (i.e. publicamente acessível por qualquer dispositivo) exclusivo para si, mas obtém um endereço IP privado através de um serviço NAT (Network Address Translation), realizado por um roteador na fronteira da sub-rede a qual está anexado.

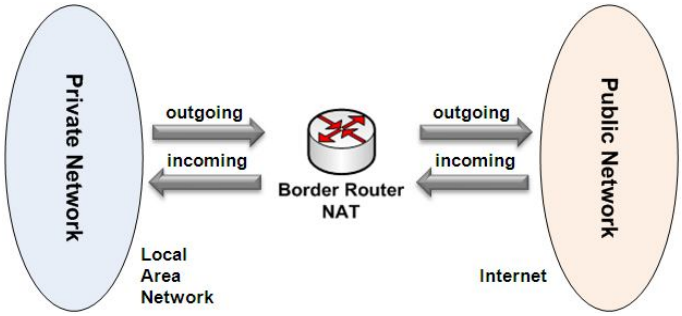

Figura 3 - Mecanismo de NAT entre uma rede privada e uma rede pública (Cisco, 2006)

O roteador NAT modifica as informações de endereço IP e porta contidas nos pacotes IP que atravessam uma sub-rede para outra. Este mecanismo é

utilizado para permitir que redes privadas interajam com redes publicas, realizando um mapeamento entre os endereços IP e portas privados para endereços IP e portas públicos e vice e versa, conforme mostrado na Fig 5. Esta associação é denominada NAT binding.

Os tipos existentes de NAT são (Cisco, 2006):

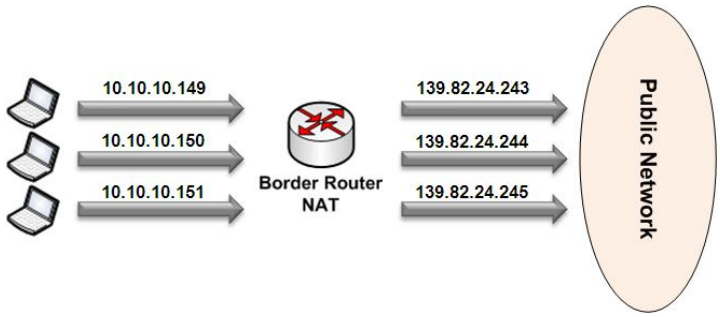

Figura 4 - Exemplo de NAT estático mapeando endereços IP privados para endereços IP públicos de forma estática e um para um (Cisco, 2006)

- NAT estático – Mapeia endereços IP privados para endereços IP públicos de forma estática (i.e. pré-configurada) e um para um. Este tipo de NAT é útil quando um dispositivo necessita ser acessível a partir do lado de fora da rede (i.e. a partir da parte externa à sub-rede a qual o NAT é realizado). Por exemplo, na Figura 4, o computador com o endereço IP 10.10.10.149 sempre terá seus pacotes enviados traduzidos para o endereço 139.82.24.243 e vice e versa. Este tipo de NAT requer uma pré-configuração de cada mapeamento, e, portanto não é muito utilizado para atender um grande e variável número de dispositivos.

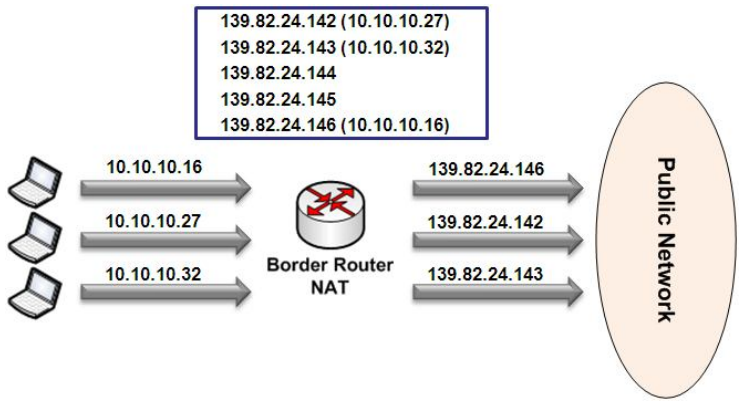

Figura 5 - Exemplo de NAT dinâmico utilizando um pool de endereços (Cisco, 2006)

- NAT dinâmico – Mapeia endereços IP privados para endereços IP públicos a partir de um conjunto (pool) de endereços publicos. O NAT dinâmico estabelece um mapeamento um-para-um entre endereços privados e públicos, mas o mapeamento pode variar dependendo do endereço público

disponível no pool, no momento da comunicação. Por exemplo, na Figura 5, cada computador com um endereço IP privado terá seus pacotes com o endereço traduzido para o primeiro endereço disponível no pool entre o intervalo 139.82.24.142 e 139.82.24.146.

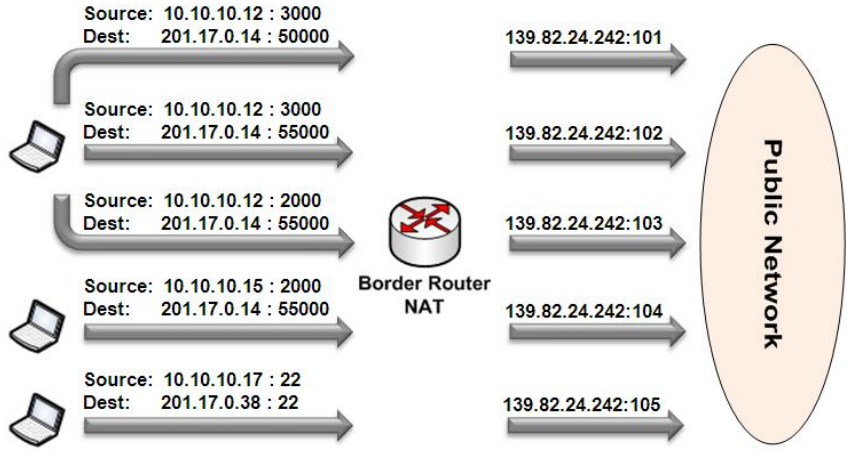

Figura 6 - Exemplo de NAT simétrico

- NAT simétrico – É a forma mais comum de NAT utilizada atualmente. Trata-se de um tipo de NAT dinâmico que mapeia múltiplos endereços IP privados para um único endereço IP público, mas utilizando diferentes portas. Por exemplo, na Figura 6, cada computador na rede privada terá seu endereço IP traduzido para o mesmo endereço IP 139.82.24.242, mas com um número de porta diferente. No NAT simétrico, um NAT binding (amarração NAT) é um mapeamento que é criado para cada combinação [endereço IP e porta privados origem, endereço IP e porta públicos destino], sempre que uma requisição feita a partir de um dispositivo interno à sub-rede com o NAT envia pacotes para qualquer lugar na rede externa a esta sub-rede. Este mapeamento (binding) é mantido temporariamente para que respostas retornadas a partir do dispositivo na rede externa, para o qual a requisição foi enviada, possam ser encaminhadas de volta para os dispositivos na rede interna (as respostas devem vir de um dispositivo externo com o endereço e porta públicos contidos no binding criado).

Como foi mencionado acima, o tipo de NAT mais utilizado hoje em dia é o NAT simétrico, devido à grande quantidade de dispositivos que em geral são atendidos em uma mesma sub-rede. Para os tipos mais simples de comunicação cliente/servidor, tais como acesso a Web, este tipo de NAT não apresenta problemas, pois quando uma aplicação cliente deseja se comunicar com um

servidor estabelecido em um dispositivo na rede externa ao NAT, os pacotes das requisições originados na rede interna ao NAT criam o binding, com a combinação [endereço IP e porta privados fonte, endereço IP e porta públicos destino], que é utilizado pelo roteador NAT para encaminhar os pacotes de respostas enviados pelo servidor de volta para os dispositivos.

Entretanto, a utilização de NAT simétrico na fronteira de uma sub-rede impede que dispositivos na rede externa ao NAT daquela sub-rede iniciem comunicações com dispositivos na rede privada. Por um lado, isto é vantajoso, pois protege os dispositivos da sub-rede interna contra ataques externos maliciosos. Por outro lado, isto significa que uma aplicação cliente executando em um dispositivo na rede externa ao NAT não pode iniciar uma comunicação com um servidor estabelecido em um dispositivo com um endereço na rede interna ao NAT, pois não existe NAT binding que faça o mapeamento entre o endereço IP e porta externos fonte dos pacotes enviados pela aplicação cliente para o endereço IP e porta internos do servidor na rede interna. Ou seja, o roteador NAT não tem como descobrir para qual dispositivo interno ao NAT os pacotes devem ser encaminhados. Esta questão gera problemas para a utilização de comunicações peer-to-peer, telefonia IP (VoIP) e do protocolo SIP (Georgescu, 2004).

O protocolo SIP, em sua forma mais básica, não pode ser utilizado com o mecanismo de NAT, pois os parâmetros com os endereços IP e portas utilizados tanto pela sinalização SIP quanto pelas mídias de comunicação (i.e. parte SDP da mensagem) são transmitidos dentro das próprias mensagens SIP. Um User Agent SIP executando em um dispositivo conectado a uma sub-rede que utiliza NAT não tem como saber com que endereço ele é alcançável a partir da rede externa ao NAT (i.e. a Internet), pois este só possui o conhecimento sobre o endereço IP e porta locais do dispositivo no qual ele está sendo executado. Logo, os endereços IP e portas incluídos na mensagem SIP ficam inválidos quando a mensagem é enviada através de um NAT, pois o User Agent receptor da mensagem (em um dispositivo externo ao NAT) não conseguirá enviar pacotes para os endereços IP contidos dentro da mensagem diretamente, para enviar mensagens SIP ou dados de comunicação.

O problema da utilização do protocolo SIP com NAT pode ser dividido em duas partes, a parte da sinalização SIP (i.e. envio e recebimento de mensagens

SIP) e a parte da descrição de mídias de comunicação (i.e. SDP) que permite a troca de dados propriamente dita entre os participantes da sessão:

Sinalização SIP – Quando mensagens do tipo REGISTER ou INVITE são enviadas, o cabeçalho <Contact> da mensagem contém o endereço IP e porta do User Agent que enviou a mensagem. No caso de uma mensagem REGISTER, o endereço IP e porta contidos no cabeçalho <Contact> são utilizados pelo Registrar para registrar um mapeamento do URI do usuário que enviou a mensagem para o endereço de rede do mesmo no Location Service. No caso de uma mensagem INVITE, o endereço e porta contidos no cabeçalho <Contact> indicam o endereço de rede em que o User Agent que enviou a mensagem deseja receber futuras mensagens SIP. Além do cabeçalho <Contact>, as mensagens SIP incluem o cabeçalho <Via>, que contém uma "pilha" de endereços IP e portas dos dispositivos (User Agent chamador, SIP Proxyes, e User Agent chamado) pelos quais a mensagem SIP já passou, representando a rota pela qual uma mensagem de resposta deve retornar (i.e. através dos mesmos dispositivos que foi enviada). Em ambos os casos, referentes ao cabeçalho <Contact> e ao cabeçalho <Via>, os endereços incluídos ficam inválidos quando a mensagem atravessa um mecanismo de NAT.

Este problema, em ambos os cabeçalhos, pode ser resolvido através de mecanismos dentro do próprio protocolo SIP. Para tal, deve ser configurado, tanto no Servidor SIP quanto nos User Agents, um parâmetro indicando que os mesmos devem utilizar o endereço IP e porta do pacote recebido (endereço e porta que foram traduzidos pelo NAT), e substituir o endereço e porta contidos no cabeçalho <Contact>. Além disso, é necessário acrescentar um parâmetro "received=address:port" ao lado do último endereço contido no cabeçalho <Via>. Esta substituição do endereço no cabeçalho <Contact> e a inclusão de parâmetro no cabeçalho <Via> corrigem a mensagem com os endereços traduzidos pelo NAT e pelos quais as respostas podem ser de fato enviadas, ou no caso do REGISTER, corrige o endereço de rede do User Agent que enviou a mensagem.

A Figura 7 mostra um exemplo de mensagem SIP enviada por um User Agent executando em um dispositivo anexado a uma sub-rede que utiliza NAT. O User Agent que envia a mensagem está no endereço IP e porta 192.168.0.123:5080 e a mensagem é recebida por um User Agent executando em um dispositivo com endereço IP e porta 139.82.24.239:5080 (O User Agent

receptor da mensagem poderia também estar atrás de um NAT, mas é apresentado com um IP público para fins de simplificação do exemplo). A figura mostra a substituição do endereço IP e porta do cabeçalho Contact, e inserção de parâmetro 'received=' no cabeçalho Via, pelo endereço IP e porta obtidos do pacote recebido, refletindo a tradução realizada pelo NAT.

Figura 7 – Exemplo de mensagem SIP enviada a partir de um User Agent executando em um dispositivo anexado a uma sub-rede com NAT.

Adicionalmente a estes mecanismos, como os bindings do NAT são temporários, o Servidor SIP ou os User Agents devem utilizar algum mecanismo que mantenha o “caminho aberto” através dos NATs para a sinalização SIP, enviando mensagens SIP ou pacotes IP auxiliares (também conhecidos como pacotes keep-alive) para manter estes bindings NAT ativos. O Servidor SIP pode ser configurado para enviar periodicamente tais pacotes IP para todos os endereços de rede registrados em seu Location Service. Entretanto, dependendo do tipo de NAT utilizado, pode não ser suficiente enviar pacotes a partir da rede externa ao NAT. Este problema pode ser resolvido configurando também os User Agents executando em dispositivos com endereços internos ao NAT que enviem mensagens REGISTER para o Servidor SIP com um intervalo de tempo menor do

que o tempo de expiração do NAT binding. Isto é possível porque hoje em dia os User Agents SIP utilizam uma sinalização simétrica (i.e. enviam e recebem mensagens SIP através da mesma porta de comunicação). Desta forma, tendo o caminho de sinalização SIP aberto através dos NATs é sempre possível alcançar os User Agents para negociar sessões de comunicação (Georgescu, 2004).

Descrição SDP e troca de dados – Conforme já foi explicado, a parte SDP da mensagem SIP contém os endereços IP e portas através dos quais o dispositivo que enviou a mensagem deseja trocar dados de comunicação. Quando a mensagem SIP atravessa um NAT, estes endereços tornam-se também inválidos e necessitam ser corrigidos. Caso contrário o dispositivo que recebeu a mensagem não poderá utilizá-los para enviar dados para o outro participante da sessão. Este problema não pode ser resolvido de forma tão simples como a solução apresentada acima para a sinalização SIP, pois os endereços IP e portas contidos na parte SDP são diferentes daqueles utilizados para enviar e receber mensagens SIP.

Dentre algumas soluções para este problema (Georgescu, 2004), podemos destacar a utilização de um elemento intermediador da comunicação, estabelecido na rede pública, conhecido como Media Proxy. Esta solução apresenta grande flexibilidade, pois o Media Proxy pode ser alcançado por ambos os participantes da comunicação, mesmo que ambos estejam anexados a diferentes sub-redes, e ambas as sub-redes utilizem NAT. Além disto, os User Agents não necessitam ser alterados. Na verdade, o redirecionamento da comunicação é transparente para os User Agents e os usuários. A desvantagem desta solução, entretanto, é que após o estabelecimento de uma sessão, ao invés de trocar dados diretamente, os dispositivos enviam e recebem dados através do Media Proxy, adicionando um overhead na comunicação, já que esta segue um caminho através de um terceiro elemento na rede.

A solução com o Media Proxy funciona da seguinte forma. Quando o SIP Proxy recebe uma requisição INVITE, ao invés de repassá-la diretamente adiante (para o User Agent chamado ou para outro SIP Proxy no domínio do URI do usuário chamado), este a intercepta, extraindo o identificador da sessão da mesma (i.e. cabeçalho Call-id contido na requisição SIP) e os dados da descrição SDP. Em seguida, o SIP Proxy repassa estes dados para o Media Proxy através de uma conexão direta com o mesmo através de sockets Unix.

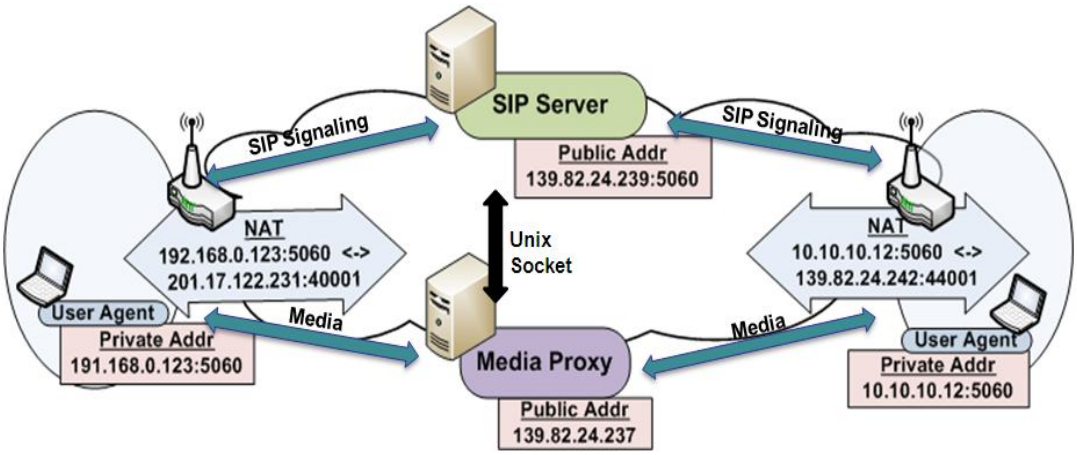

Figura 8 - Exemplo de cenário com utilização de um Media Proxy para travessia de NAT

O Media Proxy então verifica se já existem portas alocadas para aquela sessão. Caso não existam, o Media Proxy aloca portas correspondentes a aquelas contidas na descrição SDP da requisição SIP (aloca a mesma quantidade de portas de comunicação contidas na descrição SDP, não necessariamente com o mesmo número das portas de comunicação), começa a escutar nas portas alocadas, e retorna os números das portas alocadas para o SIP Proxy, que substitui os endereços IP e portas contidos na descrição SDP pelo endereço IP do Media Proxy e pelos números das portas de comunicação alocadas no mesmo. Caso já existissem portas alocadas para aquela sessão, o Media Proxy teria apenas retornado para o SIP Proxy os números das portas já alocadas. O SIP Proxy então repassa a requisição INVITE adiante normalmente.

No sentido oposto, quando o SIP Proxy recebe de volta uma resposta (e.g. OK) do User Agent para o qual encaminhou a requisição INVITE acima descrita, o SIP Proxy realiza o mesmo procedimento, repassando as informações da resposta SIP para o Media Proxy, que aloca portas correspondentes a parte SDP da resposta SIP recebida. Após substituir os endereços IP e as portas contidas na resposta SIP pelo endereço IP do Media Proxy e os números das portas alocadas no mesmo, o SIP Proxy repassa a resposta adiante para o User Agent que inicialmente havia enviado a requisição INVITE correspondente.

Quando a sessão é estabelecida, o Media Proxy escuta nas portas que alocou para a sessão e espera receber ao menos um pacote UDP originado de cada um dos participantes da sessão. Assim que este pacote é recebido, o Media Proxy preenche uma estrutura de dados que contém informações sobre o endereço IP e porta fonte do pacote recebido. Quando ambas as estruturas de dados das portas

que formam um par para a comunicação são preenchidas (i.e. um par de portas de comunicação alocadas para a troca de dados entre os participantes) o Media Proxy começa a encaminhar os pacotes trocados entre os participantes.

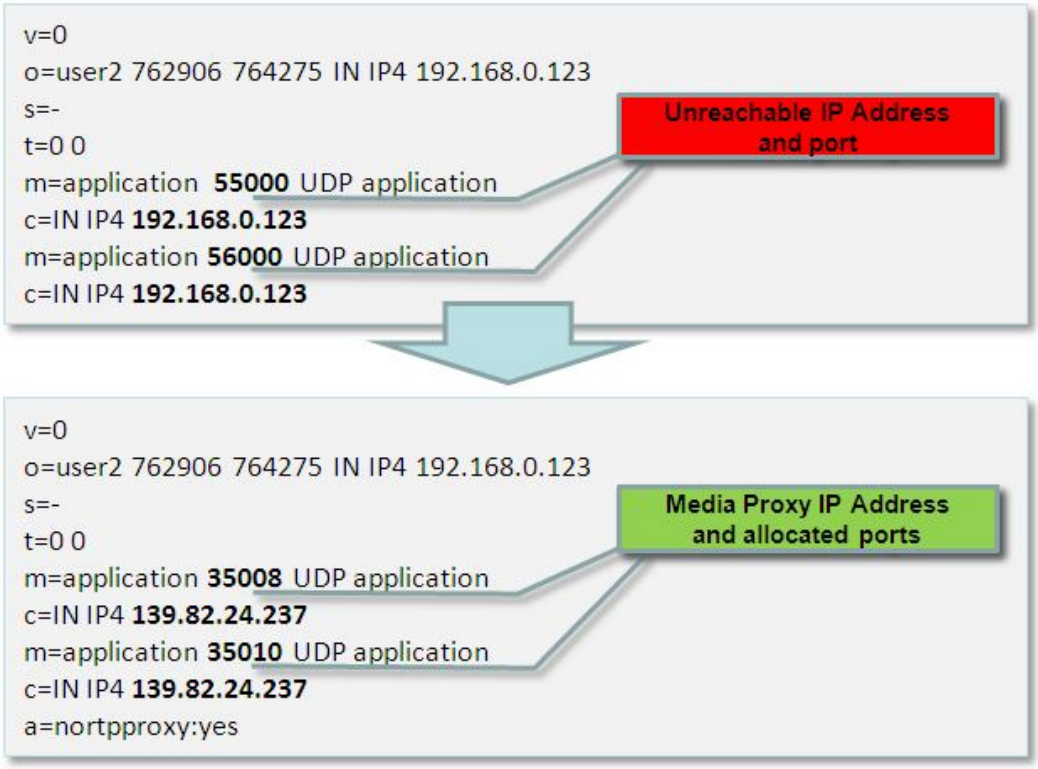

Figura 9 - Exemplo de descrição SDP de uma requisição SIP recebida pelo SIP Proxy, em que os endereços IP e portas contidos na descrição SDP são substituídos

Na literatura, existem outras abordagens para a travessia NAT, mas ao que tudo indica, nenhuma delas é tão geral e flexível como a solução usando o Media Proxy. Esta solução foi implementada neste trabalho juntamente com a API SIP User Agent, apresentada no Capítulo 4. Para isto, os SIP Proxies utilizados foram configurados para detectar quando os dispositivos estão conectados em redes protegidas for NAT e utilizar o Media Proxy quando necessário.

2.3. Sistemas Publish/Subscribe

Sistemas baseados em eventos, também conhecidos como publish/subscribe (publicação/subscrição de eventos), permitem que provedores de informações se comuniquem de forma assíncrona com consumidores interessados nestas através de uma infra-estrutura de servidores de eventos (ou Event Brokers). Para isto, os consumidores enviam para a infra-estrutura registros de interesse (também

chamados de subscrições) sobre os dados que desejam receber. Em geral, os dados publicados pelos produtores na infra-estrutura são chamados de eventos e a entrega destes dados pela infra-estrutura para consumidores interessados é chamada de notificação. Quando um evento é gerado e publicado no sistema, a infra-estrutura é responsável por: a) verificar a correspondência entre o evento e todas as subscrições existentes e b) entregar notificações da ocorrência do evento a todos os consumidores que fizeram subscrições correspondentes ao evento (Eugster, Felber et al., 2003).

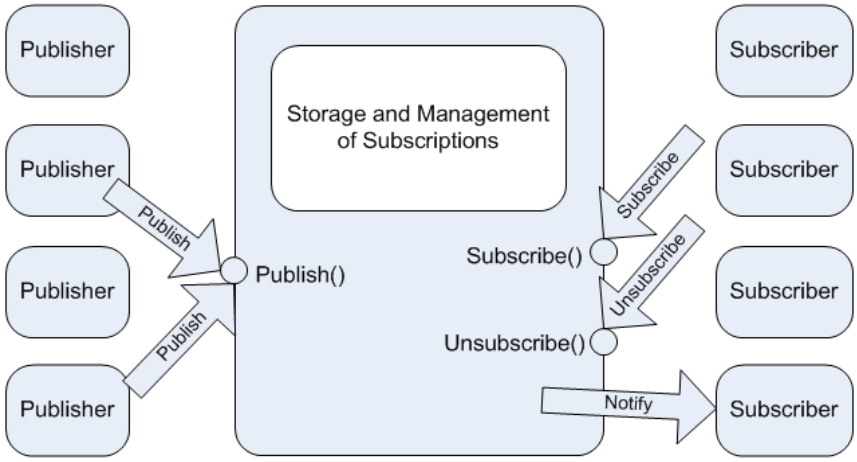

Figura 10 – Modelo de comunicação publish/subscribe (Eugster, Felber et al., 2003)

O paradigma de comunicação publish/subscribe (Figura 10) oferece aos participantes interações assíncronas, anônimas e em caráter de difusão (um-para-muitos). A assincronia vem do fato de que o consumidor (i.e. o assinante) não precisa consultar o produtor (remetente, ou publicador) para obter os dados, mas estes chegam a ele assincronamente, isto é, independente do seu estado de execução. A interação é anônima, pois os participantes da comunicação não precisam identificar as partes com quem vão se comunicar, já que a infra-estrutura se encarrega de fazer a correspondência entre os eventos publicados e as subscrições existentes. Um serviço de eventos possui uma natureza um-para-muitos, pois um mesmo evento pode gerar notificações para diversos consumidores com apenas uma operação de publicação (Eugster, Felber et al., 2003).

Dentre as principais características que podemos observar nos sistemas publish/subscribe existentes, pode-se destacar a forma como é feita a associação entre publicações de eventos e subscrições. Alguns sistemas publish/subscribe utilizam um encaminhamento baseado em grupos (ou canais), ou baseado em

assuntos (ou tópicos), tais como nas especificações CORBA Event Service (Omg) e Jini Distributed Events Specifications (Sun), ou no sistema TIBCO Rendezvous (Tibco). Neste tipo de sistema, cada evento é publicado para um destes canais pelo publicador ou é marcado com um canal ou assunto, e cada assinante define sua subscrição em termos do nome do canal ou assunto. Por exemplo, o método Subscribe(“PreçoAçãoX”) especificaria o interesse de um consumidor em receber notificações de eventos gerados para o canal “PreçoAçãoX”, um tópico hipotético que seria utilizado para a publicação de eventos sobre a ação X na bolsa de valores. Um produtor poderia então publicar um evento com o preço da ação X utilizando o método Publish(“PreçoAçãoX”,”27,05”). Neste tipo de comunicação, tipicamente existem vários consumidores, e na qual os consumidores ignoram completamente quem é, ou quais são, o(s) publicador(es) de eventos para um canal. Entretanto, permanece ainda um acoplamento implícito entre os publicadores e consumidores de eventos, através do nome do canal ou tópico (Huang e Garcia-Molina, 2004).

Outros sistemas publish/subscribe implementam um paradigma mais flexível, chamado de encaminhamento baseado em conteúdo, que permite que o consumidor inclua em sua subscrição uma consulta geral e arbitrária sobre o conteúdo dos eventos. Portanto, ao invés de confiar no publicador para classificar os eventos em canais ou em assuntos, o consumidor é capaz de definir subscrições complexas sobre o conteúdo (i.e. os dados associados) dos eventos através de uma linguagem específica, e que geralmente inclui expressões sobre valores de dados ou atributos, e operadores de comparação e operadores lógicos (!=,<,<=,>,>=, AND, OR e etc), que identificam os eventos de interesse. Em um sistema como esse, por exemplo, um publicador poderia gerar um evento de nome “PreçoAçãoX”, definir um atributo “Preço” para este evento e atribuir-lhe um valor, por exemplo [Preço = 27,05]. Assim, todos os consumidores que houvessem submetido subscrições cuja condição de interesse, por exemplo [Preço > 25], satisfizesse o valor do atributo publicado receberiam uma notificação, enquanto que os demais simplesmente não receberiam notificação alguma. Desta forma, os consumidores não recebem notificações de todos os eventos gerados em um canal específico, mas apenas eventos selecionados. Para tal, o servidor de eventos faz uma filtragem dos eventos para cada consumidor, de acordo com as condições de interesse indicadas. Alguns dos principais sistemas

publish/subscribe que se enquadram nesta categoria são JEDI (Cugola, Di Nitto et al., 2001), Elvin (Segall, Arnold et al., 2000), REBECA (Zeidler e Fiege, 2003) e Siena (Carzaniga, Rosenblum et al., 2001).

Outra característica que diferencia os sistemas publish/subscribe é a arquitetura da infra-estrutura utilizada, i.e. a topologia de servidores de eventos, conforme mostrado na Figura 11. A infra-estrutura publish/subscribe, em sua forma mais básica, consiste de um único servidor de eventos que é o componente responsável por receber subscrições e publicações, realizar a correspondência entre elas, e encaminhar apropriadamente as notificações aos consumidores. Entretanto, o sistema pode ser implementado com múltiplos servidores trabalhando em conjunto de uma forma distribuída aumentando assim a escalabilidade do sistema (Huang e Garcia-Molina, 2004). Como sistemas publish/subscribe com arquiteturas distribuídas podemos citar o REBECA (Zeidler e Fiege, 2003) e Siena (Carzaniga, Rosenblum et al., 2001), dentre outros.

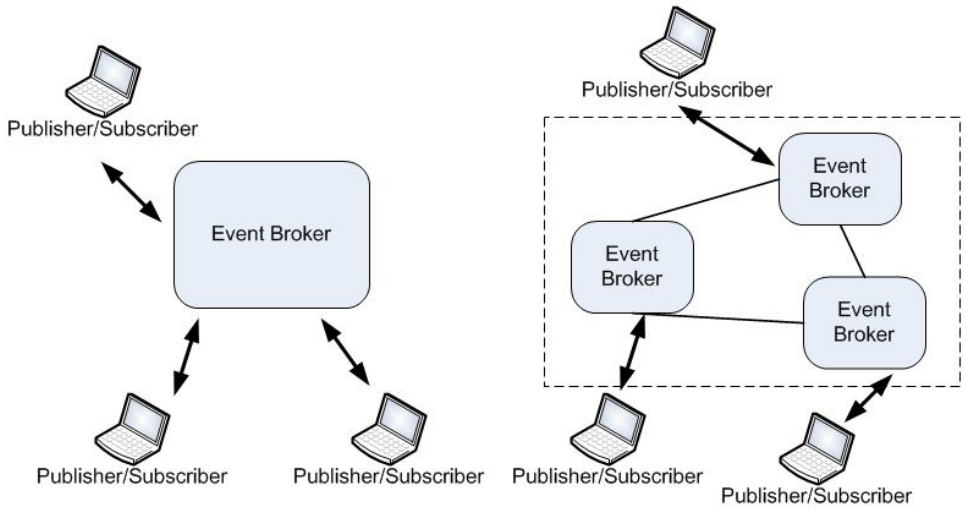

Figura 11 - Arquitetura centralizada vs. arquitetura distribuída para sistemas publish/subscribe

Neste trabalho, é considerado um sistema baseado em conteúdo, com arquitetura centralizada. Utilizamos apenas um único servidor de eventos, pois o foco de estudo é o tratamento da mobilidade IP e da desconexão de dispositivos conectados a um mesmo servidor, e não aspectos de mobilidade e desconexão de clientes em sistemas publish/subscribe com vários servidores de eventos, pois isto iria implecar no estudo de protocolos para o roteamento de subscrições e notificações entre diferentes servidores de eventos, como conseqüência de migrações de clientes entre servidores. Logo, este assunto ultrapassaria o escopo

deste trabalho. Isto não impede que em um trabalho futuro o sistema adaptado nesta dissertação possa ser incorporado ou estendido para uma arquitetura distribuída de publish/subscribe, com vários servidores de eventos interconectados.