5
Gerenciamento de Mobilidade Aplicado a um Sistema Publish/Subscribe

5.1.
Requisitos de um Sistema Publish/Subscribe para Suportar Mobilidade e Desconexão

Este capítulo apresenta alguns dos principais requisitos que um sistema publish/subscribe deve atender para suportar a mobilidade e desconexão de dispositivos. A lista abaixo mostra uma compilação de requisitos, apresentados e discutidos por diversos autores (Huang e Garcia-Molina, 2004), (Cugola e Jacobsen, 2002), (Mühl, Ulbrich et al., 2004), (Sutton, Arkins et al., 2001):

- **A. Gerenciamento de Mobilidade**
  - A.1. Sessões de Comunicação
  - A.2. Suporte a Mobilidade de Terminais
  - A.3. Suporte a Mobilidade Pessoal
- **B. Suporte a Desconexão**
  - B.1. Tempo de Vida de Subscrições e Consulta de Subscrições pelo Cliente
  - B.2. Persistência de Notificações
  - B.3. Restrições de Geração de Notificações
    - B.3.1. Restrições Temporais
      - B.3.1.1. Freqüência
      - B.3.1.2. Intervalos de Tempo
  - B.4. Políticas de Enfileiramento de Notificações
    - B.4.1. Limitação da Idade ou Número Máximo de Notificações

A Seção 5.2, apresenta a descrição da implementação destes requisitos na adaptação de um sistema publish/subscribe utilizado como estudo de caso para a solução de Gerenciamento de Mobilidade.

A. Gerenciamento de Mobilidade

O sistema publish/subscribe deverá utilizar alguma Solução de Gerenciamento de Mobilidade que garanta a conectividade dos componentes do sistema em um cenário de mobilidade.

A.1. Sessões de Comunicação

A Solução de Gerenciamento de Mobilidade utilizada pelo sistema publish/subscribe deve oferecer uma entidade lógica, que pode ser chamada de

Sessão de Comunicação, utilizada para organizar a interação adequada com os usuários. Uma Sessão de Comunicação é um vínculo lógico com um usuário. Ela deve conter o identificador independente de localização (e.g. URI) do usuário, endereços IP, números de portas de conexão, e tipos de protocolo de comunicação, de um ou mais dispositivos que o usuário está utilizando. Uma Sessão de um usuário deve conter uma indicação do estado de conectividade do usuário com o sistema (i.e. se o usuário pode ser contactado3 através de algum dos endereços contidos na Sessão).

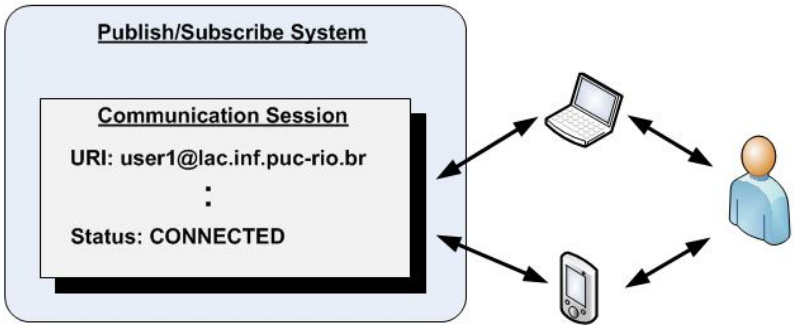

Figura 23 – Exemplo de Sessão de Comunicação de um usuário.

Quando uma Sessão é utilizada para enviar algum dado para um usuário e o usuário não puder ser contactado através de nenhum endereço contido na Sessão, o usuário da Sessão deverá ser definido como desconectado do sistema publish/subscribe até que a Sessão seja atualizada com algum endereço onde o usuário possa ser contactado. Além disto, quando um dispositivo do usuário detectar sua própria desconexão e em seguida o re-estabelecimento da sua conectividade, ele deverá atualizar sua Sessão de Comunicação com o sistema publish/subscribe indicando sua conectividade re-estabelecida.

A.2. Suporte a Mobilidade de Terminais

A solução de Gerenciamento de Mobilidade utilizada pelo sistema publish/subscribe deverá suportar a mobilidade dos dispositivos, ou seja, os usuários deverão continuar subscritos no sistema e receber notificações do Servidor de Eventos mesmo que seus dispositivos mudem de endereço IP. A Solução de Gerenciamento de Mobilidade utilizada deve, portanto, permitir a atualização de uma Sessão de Comunicação com o endereço corrente de um ou

[3] Contactar um usuário se refere ao sistema publish/subscribe conseguir entregar uma mensagem (e.g. uma notificação de um evento) a um dispositivo que ele esteja utilizando. Não se refere à garantia de que o usuário de fato teve conhecimento da mensagem.

mais dispositivos que o usuário para aquela sessão está utilizando. Sempre que um dispositivo em utilização pelo usuário detectar uma mudança em seu endereço IP, deverá atualizar a Sessão de Comunicação com seu novo endereço.

A.3. Suporte a Mobilidade Pessoal

A solução de Gerenciamento de Mobilidade utilizada pelo sistema publish/subscribe deverá suportar a mobilidade pessoal dos usuários, isto é, os usuários deverão poder ser globalmente alcançáveis através de um único identificador pessoal e também deverão poder utilizar o sistema e serem notificados independentemente do terminal que estão utilizando. Para isto, uma Sessão de Comunicação com o sistema publish/subscribe deverá conter o identificador independente de localização (e.g. URI) do usuário a quem ela pertence, e poder ser estabelecida através de diferentes dispositivos do mesmo usuário, independentemente de sua localização (i.e. ponto de conexão com a rede de dados). Quando uma subscrição de um usuário corresponder a um evento gerado, a notificação correspondente deve ser encaminhada a todos os dispositivos que o usuário está utilizando para acessar o sistema. Opcionalmente, este comportamento poderia ser parametrizado na própria subscrição, por exemplo, o usuário poderia especificar que aquela subscrição só vale para o mesmo dispositivo utilizado no momento da subscrição. Assim, a notificação do evento só seria gerada para aquele dispositivo que foi utilizado para submeter aquela subscrição.

B. Suporte a Desconexão

O sistema publish/subscribe deve tratar corretamente a desconexão temporária dos produtores e consumidores de eventos, já que em um cenário móvel os dispositivos freqüentemente sofrem desconexão da rede, seja porque são movidos para regiões sem cobertura da rede de dados, ou porque são desligados para poupar energia (Huang e Garcia-Molina, 2004). Portanto, devem ser implementados mecanismos para suportar tal desconexão, tais como a permanência das subscrições no serviço de eventos mesmo para usuários desconectados (por um tempo determinado), a persistência de notificações, e a capacidade de definição, nas subscrições, de restrições adicionais sobre a geração de notificações e políticas de enfileiramento de notificações geradas (Mühl, Ulbrich et al., 2004).

B.1. Tempo de Vida de Subscrições e Consulta de Subscrições pelo Cliente

Deve ser permitida a permanência de subscrições no sistema publish/subscribe mesmo para usuários desconectados, e definido um tempo de vida para estas subscrições. O limite no tempo de vida das subscrições evita que usuários que se desconectaram definitivamente tenham subscrições desnecessárias consumindo recursos do Servidor, gerando um acúmulo de subscrições ao longo do tempo (Mühl, Ulbrich et al., 2004).

O cancelamento, ou varredura, de subscrições com tempo de vida expirado não necessitaria ser executado com uma alta taxa de freqüência (o que geraria uma sobrecarga de processamento no servidor), contanto que o sistema garanta que nenhuma notificação de evento será gerada para uma subscrição expirada. Uma varredura de tempos em tempos pode ser suficiente para realizar a remoção das subscrições expiradas sem sobrecarregar o servidor (Sutton, Arkins et al., 2001).

Além disso, deverá existir uma forma do cliente detectar que suas subscrições foram canceladas. O Servidor de Eventos poderia enviar um tipo de notificação especial ou o oferecer uma interface de consulta de subscrições existentes. Deve ser possível especificar qual o comportamento do cliente para lidar com subscrições canceladas de um usuário: se ele irá renová-las ou se irá re-enviar as subscrições canceladas. No momento em que um usuário se recupera de uma desconexão temporária, o cliente que ele está utilizando deve verificar se o usuário possuía subscrições que foram canceladas. Caso o dispositivo seja desligado e todo o estado local do cliente seja perdido (ciência de quais subscrições o usuário possuía no servidor), ele será capaz de obter do servidor as subscrições que o usuário possuía através da interface de consultas de subscrições oferecida pelo servidor (caso a aplicação necessite de tal recuperação de estado).

B.2. Persistência de Notificações

O sistema publish/subscribe deverá armazenar as notificações geradas para um usuário desconectado em uma fila, para que estas sejam entregues quando o usuário se re-conectar. Logicamente, deverá ser parametrizado no Servidor de Eventos o tamanho máximo para as filas de notificações armazenadas de cada usuário, para evitar um crescimento indeterminado das filas no servidor (Mühl, Ulbrich et al., 2004). Quando uma notificação for gerada para um usuário desconectado, mas a fila de notificações deste usuário estiver cheia, a notificação

do início da fila será descartada para a inserção da nova notificação. Isto representa para um usuário do sistema que, caso ele permaneça desconectado, poderá receber um número máximo de notificações mais recentes quando se re-conectar. O tamanho máximo das filas deve, portanto, ser definido considerando um compromisso entre a quantidade de memória exigida no servidor, para armazenar as filas de todos os usuários, contra o número máximo de notificações mais recentes oferecidas para os usuários do sistema em caso de desconexão.

B.3. Restrições de Geração de Notificações

Segundo (Mühl, Ulbrich et al., 2004), a limitação de recursos dos dispositivos móveis deve ser levada em consideração na forma pela qual o sistema gera notificações para os usuários da ocorrência de eventos. Uma alta taxa de geração de notificações ou um grande número de notificações desnecessárias irá sobrecarregar os recursos computacionais dos dispositivos móveis e do servidor.

Outro fator a ser observado é que algumas aplicações podem estar interessadas em serem notificadas apenas em situações específicas, e a forma de definição da expressão de interesse baseada em conteúdo pode não atender completamente a suas necessidades de filtro sobre as notificações de eventos que desejam receber. Por exemplo, uma aplicação pode estar interessada em receber notificações apenas para eventos gerados em um determinado horário do dia.

Com o objetivo de regular a geração de notificações de forma a minimizar o consumo de recursos e também para oferecer formas adicionais de filtragem de notificações, além das expressões condicionais típicas de um sistema publish/subscribe baseado em conteúdo, os usuários devem ser capazes de especificar alguns tipos adicionais de restrições para geração de notificações, tais como os listados a seguir. Tais restrições devem poder ser definidas nas subscrições, e especificadas para serem executadas em todos os momentos ou apenas quando os usuários ficarem desconectados. Por exemplo, em uma determinada Subscrição S1 seria definida uma Restrição R1 para ser executada sempre que um evento corresponder à Subscrição S1. Outra Restrição R2 poderia ser definida para a mesma Subscrição S1 para ser executada também quando um evento corresponder à Subscrição S1, mas somente quando o usuário dono da subscrição estiver desconectado. Uma notificação para um evento só será gerada,

portanto, se o evento corresponder à expressão de interesse contida na subscrição e satisfizer as restrições adicionais definidas para esta.

B.3.1. Restrições Temporais para Geração de Notificações

Os usuários deverão poder definir restrições temporais que filtram a geração de notificações de eventos de acordo com variáveis tais como a freqüência entre notificações, ou intervalos de tempo em que desejam receber notificações.

B.3.1.1. Restrições de Freqüência para Geração de Notificações

O consumo da energia nos dispositivos móveis é diretamente proporcional à freqüência do uso da interface de rede sem fio. Por outro lado, muitas aplicações não requerem uma entrega de notificações frequentes e seriam bem atendidas com baixas taxas de entrega. Por exemplo, em uma subscrição poderia ser especificado que eventos com uma determinada propriedade sejam entregues no mínimo a cada 1/15 segundos. Nesse caso, portanto, o Servidor de Eventos só iria gerar uma notificação para um evento com tal propriedade se a última notificação de um evento para aquela subscrição foi gerada a pelo menos 15 segundos. Por exemplo, um usuário só está interessado em receber o preço de uma ação da bolsa de valores se já se passaram 15 segundos desde a última publicação com esta informação.

B.3.1.2. Restrições de Intervalos de Tempo para Geração de Notificações

Poderia ser especificado um intervalo de tempo em que notificações para eventos devem ser geradas. O Servidor só irá gerar notificações para os usuários da ocorrência de eventos que ocorram dentro das datas e horas iniciais e finais especificadas. Estes intervalos também poderiam ser periódicos, por exemplo, uma subscrição poderia definir que o usuário deseja ser notificado todos os dias da semana, mas somente em determinado intervalo de tempo, por exemplo, entre 16:00hs e 17:00hs (que significa, por exemplo, o final do pregão do dia na bolsa de valores).

B.4. Políticas de Enfileiramento de Notificações

Como um dispositivo móvel pode ser desligado ou desconectado por longos períodos de tempo, muitas notificações de eventos poderiam ser geradas para o usuário durante este tempo. Mesmo que o armazenamento no Servidor de Eventos não seja uma preocupação, a grande quantidade de tempo e largura de banda

necessária para transmitir todas as notificações enfileiradas para um usuário quando ele se re-conecta pode não ser viável. Por isso, devem existir políticas que minimizem esses problemas. A limitação do número de notificações enfileiradas para um usuário já foi mencionada no item Persistência de Notificações. Outras políticas deverão poder ser definidas no sistema (Huang e Garcia-Molina, 2004).

B.4.1. Limitação da Idade ou Número Máximo de Notificações

Além da definição de um número máximo de notificações enfileiradas poderia ser definido um limite de tempo em que estas permanecem enfileiradas para um usuário. Por exemplo, poderia ser especificado que a infra-estrutura só guarde notificações durante não mais do que 20 minutos. Isto evitaria um acúmulo de notificações no servidor e uma sobrecarga no envio de todas as notificações para o usuário no momento de sua re-conexão (Sutton, Arkins et al., 2001).

5.2. Implementação dos Requisitos do Sistema Publish/Subscribe para Suportar Mobilidade e Desconexão

Esta Seção apresenta a implementação dos requisitos listados na Seção 5.2, na extensão de um sistema publish/subscribe existente chamado ECI (Event-Based Communication Interface) (Baptista, Endler et al.). O ECI foi desenvolvido no LAC (Laboratory for Advanced Collaboration) e faz parte do middleware MoCA (Sacramento, Endler et al., 2004; Viterbo, Sacramento et al., 2008). O ECI apresenta uma estrutura clássica de um sistema publish/subscribe centralizado, baseado em conteúdo, tal como na descrição de sistemas deste tipo na Seção 2.3. Ele possui uma API Java para subscritores e publicadores de eventos, chamada de ECIAgent, além de uma API Java que representa um Servidor de Eventos (ou Broker) chamada de ECIBroker. As APIs do ECI podem ser utilizadas por qualquer aplicação que necessite de uma comunicação publish/subscribe baseada em conteúdo.

Neste trabalho, estas APIs são reestruturadas e estendidas para a incorporação dos requisitos apresentados no capítulo anterior. O novo sistema publish/subscribe resultande desta extensão será chamado de MD-ECI (Mobility and Disconnection Support for the Event Based Communication Interface). Primeiramente, é assumido que o servidor de eventos (ECIBroker) deve ser estabelecido em um servidor na rede fixa, pois ele é o intermediador de todas as

comunicações, e uma desconexão deste componente levaria a uma total paralisação do funcionamento do sistema (Huang e Garcia-Molina, 2004). Os subscritores e publicadores de eventos são estabelecidos em dispositivos móveis portáteis, e podem se mover entre diferentes redes, mudando de endereço IP.

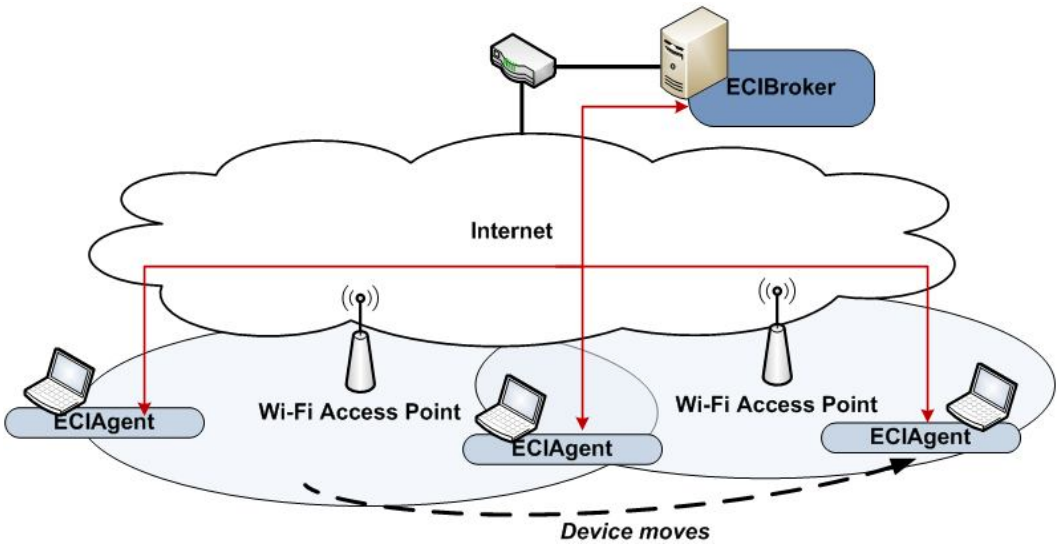

Figura 24 - Estabelecimento dos componentes do MD-ECI em um ambiente de mobilidade.

A. Gerenciamento de Mobilidade

Como já foi mencionado, o SIP User Agent é utilizado pelo MD-ECI para prover suporte à mobilidade dos produtores e consumidores de eventos. Portanto, a API SIP User Agent é acoplada tanto no ECIAgent quanto no ECIBroker, permitindo que estes estabeleçam Sessões SIP de Comunicação entre si.

Figura 25 – Utilização da API SIP User Agent pelo ECIAgent e ECIBroker

Na instanciação do ECIAgent ou do ECIBroker, o URI do usuário que está utilizando o sistema deve ser informado (no ECIBroker deve ser criado um URI de um usuário que representa o Servidor de Eventos, tal como broker@dominio). A API SIP User Agent então envia uma mensagem REGISTER para o Servidor SIP da rede doméstica do usuário (que deve ser parametrizável) com o mapeamento do URI do usuário para o endereço real do SIP User Agent que ele está utilizando.

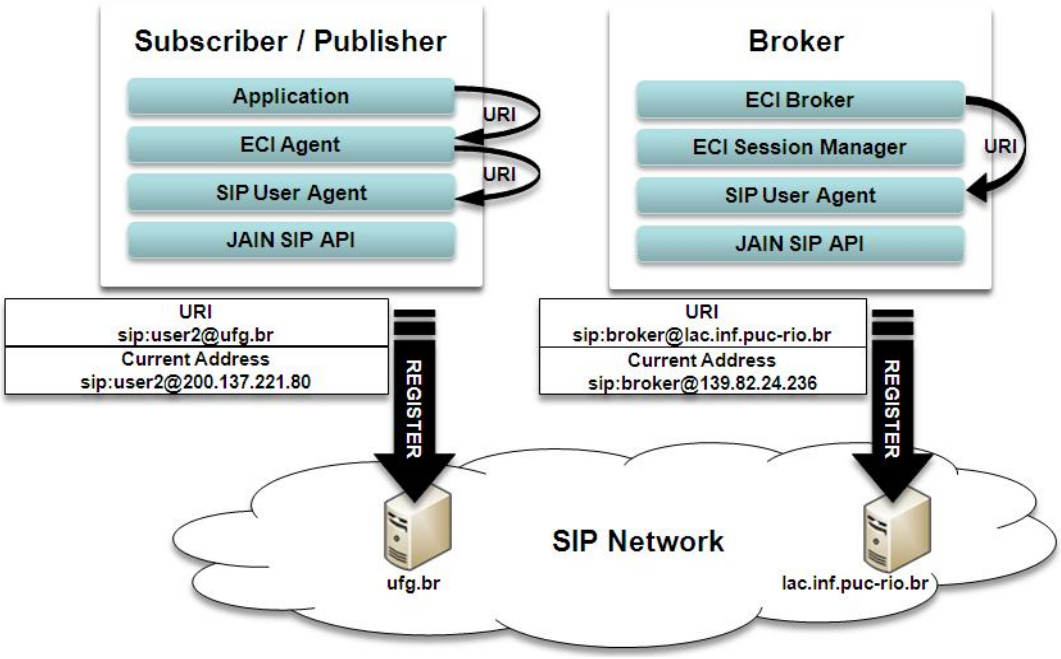

Figura 26 – Registro no Servidor SIP da rede doméstica do usuário

O URI do Servidor de Eventos também deve ser configurado no ECIAgent, para que quando o ECIAgent for inicializado, o SIP User Agent dele estabeleça uma sessão com o SIP User Agent contido no ECIBroker, Sessão esta que será mantida durante todo tempo de vida das duas entidades (Ver o passo 1 na **Figura 27**).

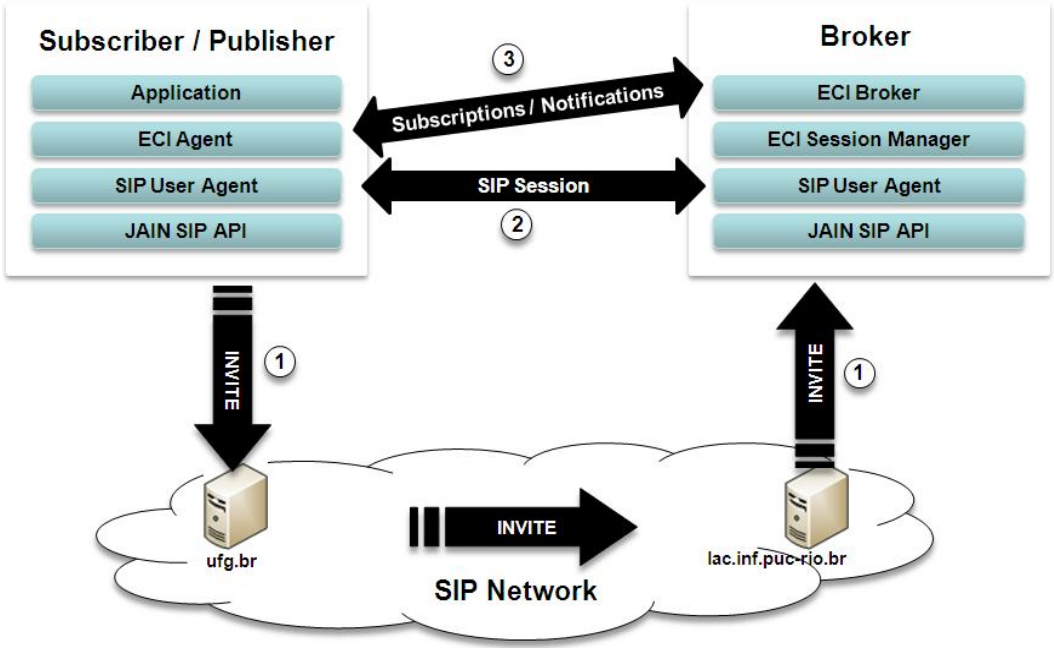

Figura 27 – Estabelecimento de uma Sessão entre o ECIAgent e o ECIBroker

Estabelecida uma sessão entre o ECIAgent e o ECIBroker (ver passo 2 na **Figura 27**), o ECIBroker armazena a sessão em seu ECI Session Manager (mais

detalhes sobre o controle de sessões serão explicados em seguida), para que ele possa obtê-la sempre que necessário. Já o ECIAgent necessita apenas guardar a única sessão estabelecida com o ECIBroker. Ambos os lados da comunicação podem então utilizar os dados das descrições SDP para enviar e receber dados, tais como subscrições, notificações e qualquer outro tipo de mensagem (ver passo 3 na **Figura 27**).

A.1. Sessões de Comunicação

No ECI original, um usuário era identificado por seu endereço IP e porta. Ao invés disso, No MD-ECI, o ECIBroker deverá utilizar o identificador independente de localização (i.e. URI) dos usuários e utilizar a Sessão SIP como entidade lógica para organizar a interação com os mesmos.

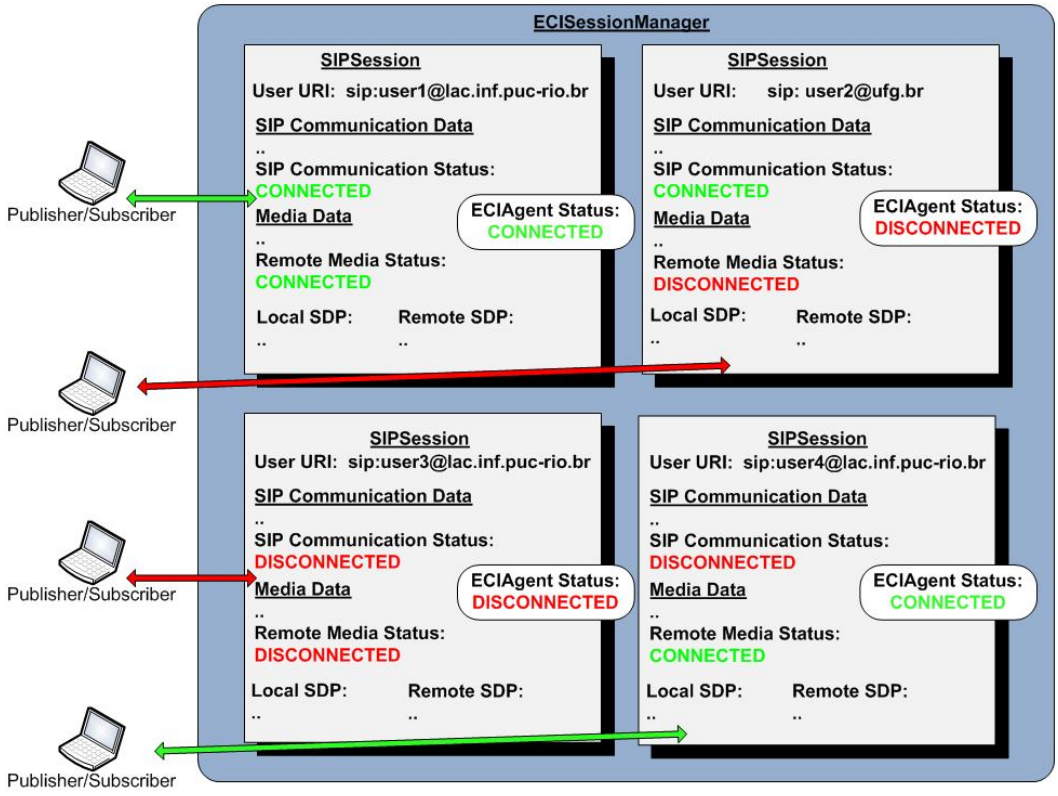

Figura 28 – Gerenciamento de Sessões de Comunicação

Um componente adicional do ECIBroker, chamado ECI Session Manager, irá gerenciar as Sessões estabelecidas com todos os ECIAgents. Quando um usuário conectado através de um ECIAgent estabelece uma Sessão com o SIP User Agent do ECIBroker, o ECIBroker armazena esta Sessão no ECI Session Manager (no ECIAgent é armazenada uma única Sessão, aquela estabelecida com o ECIBroker). A **Figura 28** mostra o ECISessionManager no ECIBroker com diversas Sessões para diferentes usuários.

Uma das principais entidades lógicas do ECI é a Subscrição, que contém a expressão de interesse (i.e. um filtro sobre o conteúdo) dos eventos que o cliente deseja receber. No ECI original uma Subscrição era identificada pelo IP e porta do dispositivo que enviou a Subscrição, e estes dados eram utilizados para o envio das notificações. Com a utilização do SIP User Agent como solução de Gerenciamento de Mobilidade, cada Subscrição será identificada pelo URI do usuário a quem ela pertence. Assim, para enviar uma notificação para um usuário, o ECIBroker utiliza o URI contido na subscrição para relacioná-la com a Sessão SIP de Comunicação do usuário e enviar os dados para o endereço contido na descrição SDP que o usuário especificou na Sessão.

A **Figura 29** mostra um exemplo de uma Subscrição identificada pelo URI do usuário a quem ela pertence (user1@lac.inf.puc-rio.br). O assunto "StockExchange" indica um assunto hipotético sobre eventos relacionados a informações de uma Bolsa de Valores. A Subscrição contém também uma expressão de interesse hipotética "(Stock == PETR4) and (Price < 25)", na qual está sendo especificado que o usuário está interessado em eventos com a propriedade "StockName" igual a "PETR4" e a propriedade "Price" com valor menor que 25.

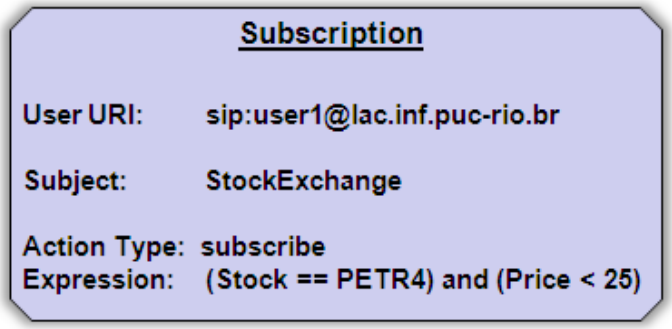

Figura 29 – Exemplo de Subscrição identificada pelo URI do usuário a quem ela pertence

A.2. Mobilidade de Terminais

O SIP User Agent oferece suporte à mobilidade de terminais para o MD-ECI. A **Figura 30** apresenta como esta funcionalidade implementada usando o SIP. O RegistrationManager possui uma thread que monitora mudanças no endereço do dispositivo. Quando este módulo detecta que o endereço IP do dispositivo mudou (possivelmente por causa da mobilidade do dispositivo), ele dispara o envio de uma nova mensagem REGISTER para o Registrar da rede doméstica do usuário, para que o dispositivo possa ser localizado em seu novo endereço

corrente (passo 1 na **Figura 30**). Caso exista alguma Sessão SIP ativa, esta é atualizada através de um envio de um REINVITE para o outro participante da Sessão (passo 2 na **Figura 30**). Assim, os dados de comunicação SIP e a descrição SDP, juntamente com os dados das mídias de comunicação sendo utilizadas, são atualizados com o novo endereço IP em ambos os User Agents participantes da Sessão. Quando o REINVITE é processado, ambos os estados, tanto de comunicação SIP como o de mídias de comunicação, são definidos como CONNECTED, pois é assumido que após uma re-negociação da Sessão, ambos os participantes possuem suas descrições compatíveis com os endereços e portas nas quais desejam trocar dados (passo 3 na **Figura 30**). Finalmente, os participantes da Sessão podem voltar a trocar dados normalmente (passo 4 na **Figura 30**).

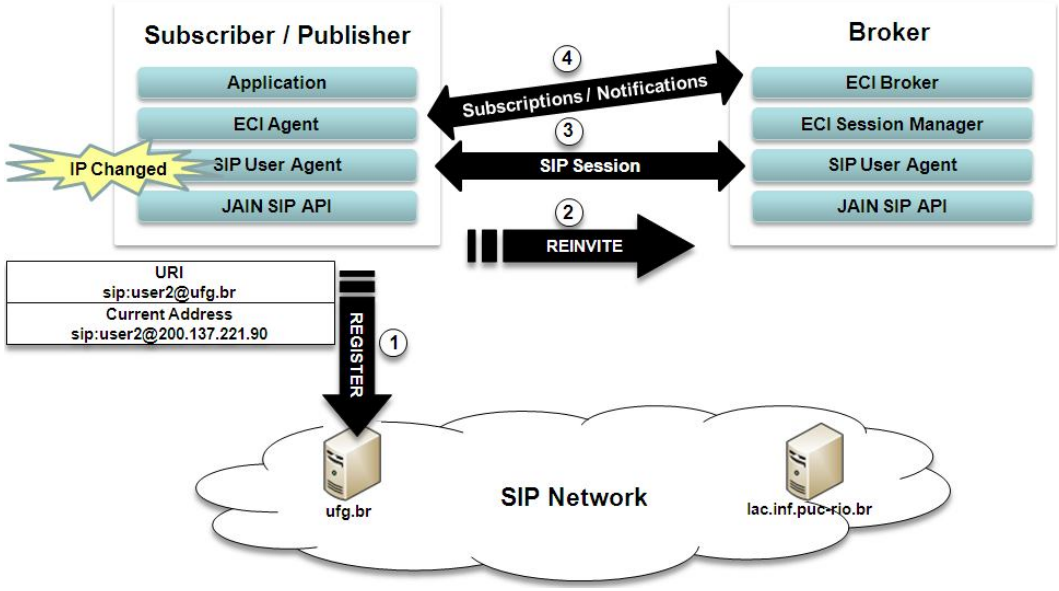

Figura 30 - Recuperação de uma Sessão após a mobilidade de um publicador/subscritor

B. Mecanismos para Suporte a Desconexão

Como todos os passos envolvidos na renociação da seção causada pela mobilidade demandam um certo tempo, no qual eventos do MD-ECI não podem ser entregues para o consumidor, é necessário que o serviço dê suporte também para o tratamento de desconexões temporárias de clientes (subscritores e publicadores de eventos), conforme requisito B do Capítulo 5.

Para lidar com a desconexão, primeiramente é necessário definir como este estado é detectado pelos componentes do sistema. Como foi explicado anteriormente, o estado da mídia de comunicação (Remote Media State) contido dentro de uma Sessão SIP indica se o outro participante da Sessão pode ser contactado através dos endereços que especificou em sua descrição SDP da

Sessão. Por exemplo, um participante da comunicação A, para enviar dados para o outro participante B, obtém a descrição SDP remota da Sessão (Remote SDP), que contém o endereço em que o participante B está esperando pela recepção de dados. Caso o protocolo de comunicação sendo utilizado para enviar os dados fosse o TCP (que oferece entrega confiável de dados), uma exceção recebida do Socket durante o envio significaria que o participante B está desconectado, uma vez que não pode receber dados no endereço que especificou em sua descrição da Sessão. Entretanto, para sistemas Publish/Subscribe a utilização do protocolo TCP aumentaria consideravelmente o overhead na comunicação entre os componentes do sistema. Por outro lado, como o protocolo UDP não oferece entrega confiável de dados, foi adaptada para o MD-ECI (e para a MoCA) a implementação de um protocolo de UDP confiável, chamado RUDP (Reliable UDP) (Stevens e Narten, 1990; Pitt, 2006). Este protocolo atribui a cada pacote enviado por um cliente RUDP um número de sequência que é usado pelo receptor (servidor RUDP) para detectar uma eventual perda de algum pacote UDP. A cada pacote que recebe com sucesso, o servidor RUDP envia um pacote de confirmação ACK de volta para o cliente. O cliente utiliza um temporizador para esperar pelos os pacotes de confirmação por um tempo determinado. O temporizador utiliza o tempo de roundtrip (i.e. o tempo que um pacote leva para ser transmitido e retornar para o cliente em uma confirmação) de cada pacote enviado e confirmado com sucesso para definir o tempo de expiração a ser aguardado em cada nova tentativa de envio de pacote. Quando um pacote é enviado e o cliente não recebe o pacote de confirmação no tempo marcado pelo temporizador, caso o número máximo de tentativas de envio não tenha sido atingido, o tempo de expiração do temporizador é exponencialmente aumentado (com um fator exponencial configurável) para evitar uma sobrecarga da rede com o reenvio de pacotes. Caso o número máximo de tentativas de envio seja alcançado e um pacote de confirmação de envio não seja recebido, uma exceção é retornada para a camada de software superior que está utilizando a implementação do protocolo. No caso do nosso trabalho, quando o sistema publish/subscribe tenta enviar um dado através da rede com um cliente RUDP e recebe uma exceção de envio, isto significa que o destinatário da mensagem não pode ser contactado e, portanto, o mesmo será marcado como DISCONNECTED.

Como é assumido que o servidor de eventos (ECIBroker) está sempre localizado em um nó na rede fixa que nunca sofre desconexões, quando um ECIAgent não consegue enviar dados para o ECIBroker, o ECIAgent implicitamente descobre que há algum problema com sua própria conectividade. Portanto, é definido que o ECIAgent pode detectar sua própria desconexão quando o Remote Media Status de sua Sessão SIP com o ECIBroker passa para o estado DISCONNECTED. O ECIAgent possui um indicador auxiliar (apenas para fins de organização) que indica seu estado de conectividade com o ECIBroker, chamado de ECIAgent Status, que na verdade é sempre igual ao estado corrente do indicador Remote Media Status da Sessão estabelecida com o ECIBroker. O ECIBroker, por sua vez, pode detectar a desconexão de um ECIAgent quando não consegue enviar algum dado para o mesmo (modificando o Remote Media Status da Sessão estabelecida com o ECIAgent para DISCONNECTED).

O ECIBroker possui também um estado auxiliar, para cada usuário, que indica o estado de sua conectividade com o sistema e que é denominado User Status. Portanto, para o ECIBroker, uma Sessão com um indicador Remote Media State definido como DISCONNECTED indica que o usuário dono da Sessão está desconectado do sistema (i.e. não pode ser contactado pelo sistema) e representa tal situação definindo o indicador User Status para DISCONNECTED. Por outro lado, uma Sessão com o indicador Remote Media State definido como CONNECTED indica que o usuário está conectado e pode ser contactado, e consequentemente o indicador User Status deste usuário estará definido como CONNECTED.

Quando uma nova Sessão é estabelecida entre um ECIAgent e um ECIBroker, o indicador Remote Media State dentro da Sessão é definido como CONNECTED para ambos os lados da comunicação, já que é assumido que ao final de uma negociação de Sessão ambos os participantes estão preparados para trocar dados através dos endereços especificados em suas descrições SDP da Sessão. Isto significa que quando uma Sessão é atualizada, ambos os participantes tornam-se imediatamente novamente re-conectados (até que ocorra alguma falha na troca de dados, o que definiria novamente o estado do lado da comunicação que não pode ser contactado como desconectado).

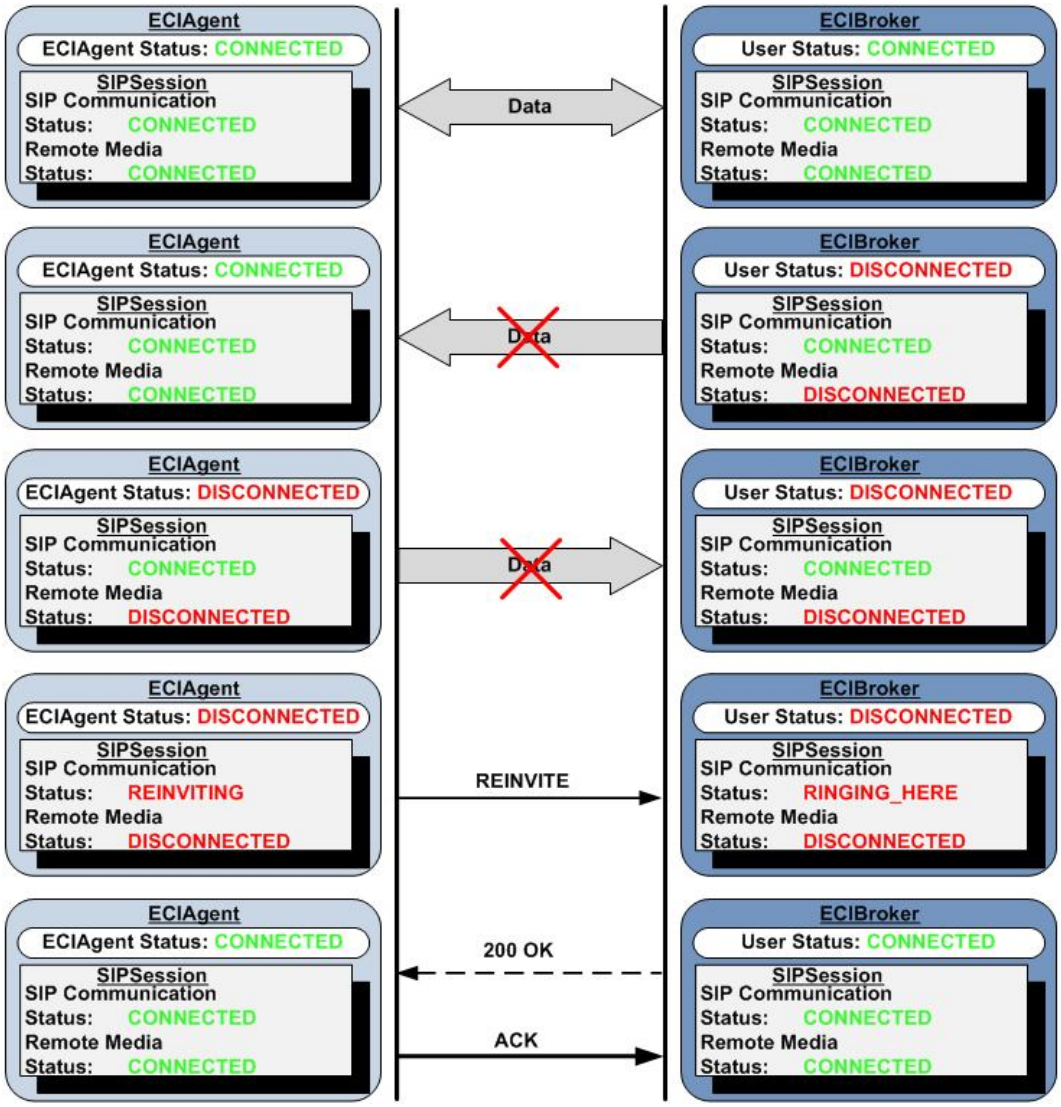

Figura 31 - Detecção da desconexão pelo ECIAgent e ECIBroker

B.1. Tempo de Expiração de Subscrições e Consulta de Subscrições pelo Cliente

As subscrições são mantidas no servidor de eventos (ECIBroker) por um tempo determinado, mesmo para usuários desconectados. Elas contêm um atributo que especifica o limite de tempo em que permanecem válidas. Quando o tempo de validade de uma Subscrição expira, esta é removida do ECIBroker. Esta verificação não necessita ser executada com uma alta periodicidade, pois não é possível gerar uma notificação para uma subscrição expirada.

Portanto o ECIBroker realiza esta verificação sempre que realiza a correspondência de eventos com as subscrições, aproveitando assim para fazer a remoção das subscrições inválidas. Caso existam notificações enfileiradas (tal como descrito no Item B.2., a seguir) para um usuário desconectado, correspondentes a uma subscrição cancelada, estas não são removidas e são

entregues normalmente quando o usuário se re-conecta, pois foram geradas quando a subscrição ainda estava válida.

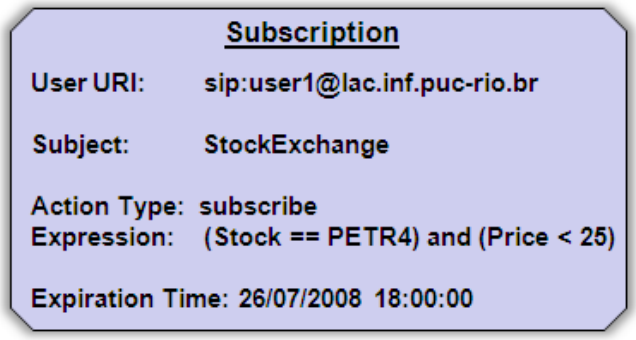

Figura 32 – Subscrição com Tempo de Expiração especificado

Em um cenário de mobilidade, um dispositivo poderia ser desconectado por sua saída da área de alcance da rede sem fio ou desligamento do dispositivo (e.g. para poupar bateria ou por falta de energia). No caso do desligamento do dispositivo, um ECIAgent executando no mesmo perderá todo seu estado (e.g. ciência das subscrições que possui no ECIBroker), assim seria necessário um armazenamento em memória persistente nos dispositivos das subscrições que um ECIAgent possui no ECIBroker. Um mecanismo que resolve tal questão é a capacidade do ECIAgent consultar o ECIBroker sempre que necessário (e.g. no momento da inicialização) para obter a lista de subscrições que possui. No momento da inicialização do ECIAgent, é parametrizado se as subscrições existentes no ECIBroker para aquele usuário devem ser consideradas ou descartadas, já que algumas aplicações podem requerer uma total recuperação das subscrições existentes como outras podem requerer uma total reinicialização do estado do ECIAgent para aquele usuário.

B.2. Persistência de Notificações

Quando um usuário está desconectado do sistema e um evento correspondente a sua subscrição é publicado, ou quando o usuário ainda não está definido como desconectado, mas não é possível entregar a notificação gerada para o mesmo (neste caso o usuário é marcado como desconectado no momento da falha de envio da notificação), a notificação é armazenada para entrega quando o usuário se re-conectar ao sistema. Tal armazenamento de notificações é realizado em filas dentro do ECIBroker, devidamente indexadas pelo URI dos usuários para otimizar sua recuperação quando necessário. Nas propriedades de

configuração do ECIBroker, pode ser definido o tamanho máximo para as filas de notificações.

B.3. Restrições de Geração de Notificações

Além da expressão de interesse sobre o conteúdo de eventos, uma coleção de restrições pode ser definida pelo usuário dentro da subscrição. Quando um evento é gerado no sistema, o ECIBroker percorre as subscrições existentes para realizar a correspondência das propriedades do evento publicado com as expressões de interesse. A cada subscrição que o ECIBroker encontra, ele primeiramente verifica se a expressão de interesse corresponde às propriedades do evento publicado. Caso o evento seja correspondente à subscrição, o ECIBroker então executa as restrições adicionais contidas na subscrição, na ordem em que estas estiverem especificadas. Somente então se as restrições adicionais forem atendidas (e.g. se o evento foi gerado há um tempo superior do que aquele especificado em uma Restrição de Freqüência) uma notificação é gerada para aquela subscrição.

B.3.1. Restrições Temporais para Geração de Notificações

São disponibilizados para o usuário dois tipos de Restrições Temporais para serem inseridas nas Subscrições, Restrições Temporais de Freqüência e Restrições Temporais de Intervalo de Tempo. Uma Subscrição só pode conter uma única Restrição de Freqüência ao mesmo tempo, pois mais do que uma Restrição de Freqüência não iria permitir distinguir a qual freqüência o usuário deseja receber eventos. Diversas restrições de Intervalo de Tempo podem ser incluídas ao mesmo tempo, possibilitando que o usuário possa definir intervalos de tempo complementares. O ECIAgent oferece métodos para adicionar tais restrições a uma subscrição.

B.3.1.1. Restrições de Freqüência para Geração de Notificações

Uma Restrição Temporal de Freqüência irá conter a grandeza da freqüência de geração de notificações (e.g. segundos, horas, dias e etc) e a quantidade em unidades relacionadas à grandeza escolhida. Por exemplo, uma Restrição Temporal de Freqüência seria criada para só permitir a geração de notificações a cada 5 minutos, neste caso a grandeza seria expressa em minutos e a quantidade seria 5. A Figura 33 mostra uma Subscrição com uma Restrição Temporal de Freqüência.

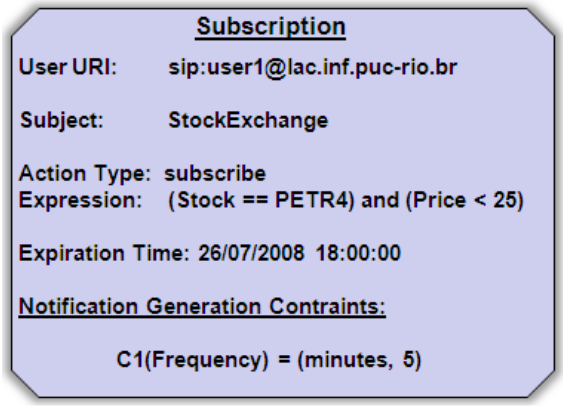

Figura 33 – Subscrição com uma Restrição de Geração de Notificações Temporal de Freqüência limitando a freqüência de geração de Notificações para 5 minutos.

B.3.1.2. Intervalo de Tempo

Uma Restrição de Intervalo de Tempo irá permitir que o usuário defina intervalos de horários diários ou datas em que Notificações devem ser geradas. Por exemplo, uma Restrição de Tempo poderá conter a data completa inicial e data completa final (i.e. ano, mês, dia, hora, segundos) representando o intervalo de tempo no qual o cliente está interessado em receber notificações, ou apenas a hora inicial e hora final (o que significaria o interesse em um horário diário).

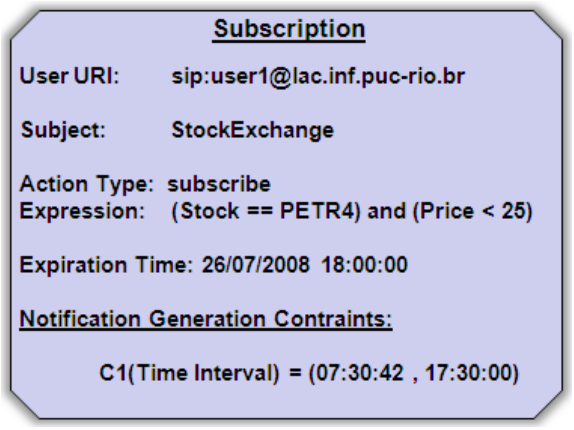

Figura 34 – Subscrição com uma Restrição de Geração de Notificações Temporal de Intervalo de Tempo limitando a geração de Notificações para um determinado horário diário.

Como já foi mencionado, o MD-ECI foi desenvolvido na linguagem Java de programação, assim como todas as extensões apresentadas neste trabalho. A API JAIN SIP, utilizada como base para o desenvolvimento da API SIP User Agent, apresentada neste trabalho, pode ser utilizada em dispositivos com JME (Java Mobile Edition) e perfil CDC (Connected Device Configuration). Por isto, o ECI que já podia ser utilizado em dispositivos com JME/CDC continua podendo ser utilizado neste tipo de ambiente. Além disso, o MD-ECI foi portado para o Sistema Operacional Android, e testado em um dispositivo G1.

Para a utilização do sistema através de NATs, é utilizado um Media Proxy tal como descrito na Seção 2.2. Como já foi mencionado, esta solução é transparente para os User Agents SIP, logo nenhuma mudança foi necessária no SIP User Agent para utilizá-la e, portanto este capítulo não apresentou nenhuma questão relacionada à travessia de NAT.