José Paulo de Oliveira Petry

Comunicações Unificadas usando os protocolos SIP e XMPP

São José – SC

agosto / 2010

José Paulo de Oliveira Petry

Comunicações Unificadas usando os protocolos SIP e XMPP

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

Orientador:
Prof. Ederson Torresini, M.SC.

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

São José – SC

agosto / 2010

Monografia sob o título *"Comunicações Unificadas usando os protocolos SIP e XMPP"*, defendida por José Paulo de Oliveira Petry e aprovada em 09 de agosto de 2010, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Ederson Torresini, M.SC.
Orientador

Prof. Emerson Ribeiro de Mello, Dr.
IFSC

Giovani Vargas Poletto
Unimed

O sucesso é construído à noite!

Durante o dia você faz o que todos fazem. Mas, para obter um resultado diferente da maioria, você tem que ser especial. Se fizer igual a todo mundo, obterá os mesmo resultados.

Roberto Shinyashiki

Agradecimentos

Em primeiro lugar agradeço imensamente ao meu pai que nunca mediu esforços com relação à minha educação, sempre estimulando a procura por novos conhecimentos.

Agradeço à minha mãe, por todo amor e carinho a mim dedicados.

À minha esposa Luísa, por estar ao meu lado.

Aos meus amigos pelo aprendizado e momentos felizes juntos.

Ao professor e orientador Ederson, por toda a ajuda e empolgação transmitida que foram de extrema importância para a conclusão deste trabalho.

Ao IFSC, pela oportunidade de aprender e evoluir como profissional e como cidadão.

Resumo

Comunicações Unificadas é a convergência de vários tipos de comunicação - telefonia, mensagens instantâneas, email, vídeoconferência e outros - sendo utilizada principalmente em empresas com o objetivo de tornar a comunicação de seus colaboradores mais eficaz. Existem diversas soluções proprietárias que implementam um ambiente de Comunicaçõe Unificadas, como por exemplo o Adobe ConnectNow da Adobe, o Webex da Cisco, o Exchange Server da Microsoft entre outros. São soluções com um custo elevado e que permitem pouca - ou nenhuma - interoperabilidade com equipamentos de outras empresas, tornando quem adquire esses produtos presos ao seu fabricante. Porém existem protocolos e soluções abertas que possibilitam a implementação de um sistema de Comunicações Unificadas completo, com baixo custo e com a liberdade de escolha dos equipamentos que farão parte da solução, independente de serem do mesmo fabricante ou não.

Existem vários trabalhos que discutem a convergência digital, porém são poucos os que tratam especificamente de Comunicações Unificadas. Menos ainda os que falam do uso de protocolos abertos nestes ambientes. Neste trabalho foi realizado um estudo referente a possibilidade de interoperabilidade entre dois dos principais protocolos abertos utilizados para comunicações: o SIP e o XMPP. Ambos foram criados para usos diferentes, porém novas extensões estão ampliando suas possibilidades de uso.

Este trabalho apresenta um estudo individual teórico destes dois protocolos e de suas extensões. Uma comparação entre suas funcionalidades e uma análise sobre a possibilidade de interoperabilidade - que consiste basicamente no mapeamento de campos semelhantes entre ambos os protocolos- demonstra que em alguns casos esta convergência será facilitada pela existência de campos em comum e em outros casos será dificultada por implementarem de formas diferentes um mesmo campo (como a identificação de uma sessão, por exemplo).

Um ambiente de testes é então criado visando a simulação de um ambiente de Comunicações Unificadas. São realizados testes de comunicação entre dois clientes de Mensagens Instantâneas, um SIP e um XMPP, onde é possível verificar que muito do que se promete na teoria ainda não está efetivamente em funcionamento na prática. Há funcionalidades que funcionam perfeitamente, porém a outras que não são implementadas ou são pela metade, possibilitanto apenas a conversão de um protocolo para outro impossibilitando o caminho de volta.

Palavras-chave: *Comunicações Unificadas, SIP, XMPP, Mensagens Instantâneas, Voip, E-mail.*

Abstract

Unified communication is the convergence of several sorts of communications, i.e. telephony, instant messages, email, videoconference, being used mainly by companies in order to have a more effective communication among the staff. There are several owned solutions that implement the Unified Communication environment, such as Adobe ConnectNow by Adobe, the Webex by Cisco, the Exchange Server by Microsoft, and others. It is a high cost solution that allows low, or none, interoperability with other companies´ equipments, tying costumers who acquire this product to its manufacturer. However there are protocols and open solutions that allow the implementation of a complete Unified Communication system, with low cost and autonomy of choosing the equipments, from the same manufacturer or not.

Numerous works discuss the digital convergence, nevertheless few of them are specifically related to Unified Communication, and even less works discuss the use of open protocols in these environments.

This work presents an assessment of the possibilities of interoperability among the two main open protocols: the SIP and the XMPP. Both of them were created for different uses, but new extensions are expanding their possibilities of use. An individual theoretical study is here presented for each of these protocols and their extensions. A comparison between their functionalities and an analysis of the interoperability possibility – which consists on mapping similar fields among both protocols – demonstrate that in some cases the convergence will be facilitated by the existence of shared fields. On the other hand, it can also become more difficult by the implementation of the same field in different ways (i.e. session identification).

A test situation was though created in order to simulate a Unified Communication environment. Communication tests were realized between two Instant Messages clients, one SIP and one XMPP client, where was possible to observe that much of what is assured in theory is no longer working in practice. There are functionalities that work perfectly, but others are partially or not implemented, ensuring only the conversion of one protocol to another, making the way back not possible.

Key-words: *Unified communication; SIP; XMPP; Instant Messages; Voip.*

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Cada vez é maior a preocupação das empresas com relação a produtividade de seus colaboradores e um fator que influência muito é a eficácia da comunicação. Existem muitas formas de comunicação disponíves hoje, como: telefone fixo, celular, email, Mensagens Instantâneas (MI) e outros. Aparentemente esta grande quantidade de opções de contatar alguem facilita a comunicação com esta pessoa, o que não é verdade pois nem sempre a opção de contato escolhida é a melhor para se comunicar naquele momento. E para tentar resolver esse problema surge o conceito de *Unified Communications* (UC) , onde através de um único número (ou endereço) de contato se alcança o destino pela melhor opção no momento.

Uma solução de UC ideal é aquela que integra todas as formas de comunicação existentes em uma empresa. Porém, dificilmente uma empresa possui um único fornecedor para todos os seus sistemas de comunicação. Este é um fator importante que dificulta a implementação de uma solução de UC, visto que a interoperabilidade entre sistemas de fabricantes diferentes raramente é total. A migração para sistemas baseados em protocolos abertos é uma forma de fugir destas dificuldades de comunicação entre sistemas proprietários.

Como será visto, soluções de UC utilizam sinalização, mídias (som, imagem, etc), MI e informações de presença. Não existe atualmente um protocolo aberto que implemente todas essas características com qualidade, onde a solução tem sido o uso em conjunto de vários protocolos abertos, sendo cada um responsável pela parte que faz melhor.

Neste trabalho será realizado um estudo dos dois principais protocolos abertos utilizados para comunicações: o *Session Initiation Protocol* (SIP), que é o mais utilizado em comunicações que envolvam a negociação de mídias (como a telefonia), e o *Extensible Messaging and Presence Protocol* (XMPP), que é muito utilizado em sistemas de MI e presença. Serão vistas as características destes protocolos e as possibilidades de sua utilização em conjunto para montar uma solução de UC mais completa possível.

1.1 Motivação

O uso de protocolos - e soluções - abertas vem se intensificando a medida que as vantagens de seu uso são percebidas pelo mercado. Porém a escolha da melhor solução é dificultada pela escassa disponibilidade de estudos sobre o uso e, principalmente, a interoperabilidade destes protocolos.

Com o interesse crescente das empresas com relação a facilidade de comunicação, seja com seus clientes ou fornecedores e também internamente, pode-se afirmar que há uma demanda por soluções que tenham como objetivo facilitar essa comunicação.

1.2 Objetivo

O objetivo geral deste trabalho é:

- Realizar um estudo comparativo dos protocolos abertos SIP/SIMPLE e XMPP de modo a verificar a possibilidade de interoperabilidade entre eles para implementação em soluções de UC

Os objetivos específicos são:

- Estudar individualmente o funcionamento e as características dos protocolos SIP/SIMPLE e XMPP.
- Identificar as vantagens e desvantagens destes protocolos.
- Comparar as funcionalidades existentes em ambos os protocolos visando a interoperabilidade entre eles.
- Verificar, em um ambiente simulado, o que a teoria promete e o que a prática cumpre.

1.3 Organização do texto

O texto está organizado em quatro capítulos. No capítulo 2 é feito uma introdução ao conceito de Comunicações Unificadas, familiarizando o leitor às terminologias e aos serviços existentes.

No capítulo 3 serão apresentados os protocolos SIP e XMPP e suas extensões, *SIP for Instant Messaging and Presence Leveraging Extensions* (SIMPLE) e as *XMPP Extension Protocol*s (XEPs). Nesta parte é feito um estudo individualmente dos protocolos, procurando entender seu funcionamento e suas características, o que ajudará futuramente no entendimento das facilidades e dificuldades que serão encontradas no momento de realizar a conversão de uma comunicação iniciada em um protocolo e finalizada em outro.

Já no capítulo 4 será realizada a comparação entre os dois protocolos. Neste ponto será comparado como cada protocolo se comporta em determinadas funcionalidades, sendo possível concluir que tipo de conversão deverá ser feita para que haja a interoperabilidade.

Após o estudo individual e a conclusão da comparação, será apresentado no capítulo 5 os testes práticos que foram executados procurando demonstrar o que já existe em funcionamento hoje com relação a conversão entre estes protocolos. Com base nestes testes será possível concluir até que ponto o que é prometido na teoria já se encontra hoje disponível - em soluções abertas - para implementação na prática.

Por fim, no capítulo 6, serão apresentadas as conclusões do trabalho, onde pretende-se apresentar os resultados obtidos nos estudos realizados.

As referências utilizadas, além de livros sobre os protocolos utilizados, se basearão principalmente em *Request for Comment*s (RFCs), devido a dificuldade em encontrar trabalhos acadêmicos abordando o uso de protocolos abertos em soluções de Comunicações Unificadas, assim como também referente a interoperabilidade destes protocolos.

2 *Comunicações Unificadas*

2.1 Definição

Como é um termo criado no mercado, é importante analisar primeiramente a sua visão no contexto comercial e, em um segundo momento, chegar a uma definição formal do que são UCs. Em uma empresa existem diversas formas para os seus colaboradores trocarem informação entre si - entre as pessoas da empresa - e com seus clientes e fornecedores - pessoas que estão fora do ambiente da empresa. Uma dessas formas é através do telefone, que pode ser fixo ou móvel: é uma forma de contato em tempo real que permite uma comunicação eficiente e com retorno imediato. Porém, caso os interlocutores desejarem realizar uma troca de arquivos durante uma conversa, isso não será possível através do telefone. Eles deverão escolher outro serviço para fazer isso, o *email* por exemplo, mas se algum deles não estiver próximo a um computador e não conseguir acessar de qualquer foram seu *email* eles terão de aguardar até conseguir acesso a este serviço. Ou seja, tais tecnologias não possuem integração entre si, como por exemplo sincronismo entre o teor da conversa e os assuntos enviados para auxiliar/complementar a conversa. A integração depende das pessoas e não da tecnologia.

Outra forma de comunicação muito utilizado nas empresas são os comunicadores instantâneos, como os conhecidos *MSN*, *Skype* e *Gtalk*. O problema é que são serviços prestados por terceiros, onde a privacidade e segurança (em que as mensagens e arquivos ali trocados), embora declarados nas licenças de uso, podem não estar de acordo com a política da empresa - ao não evidenciar os fluxos de dados e pontos de armazenamento das mensagens. Ademais, tais serviços proveem pouca ou nenhuma integração ao usuário final, exigindo que os interessados em participar desses três serviços citados, ou dos diversos outros existentes, devam criar uma conta em cada um dos mesmos. Isso acaba gerando diversos endereços para contatos além do *email*, telefone fixo, celular, fax e outros. Desta forma, sem nenhum tipo de integração, o usuário fica obrigado a gerenciar suas contas, contatos, mensagens e histórico. Aliás, cabe ressaltar, que agendas e contatos são pontos cruciais em um ambiente corporativo, mas são gerenciados por seus usuários com nenhum, ou com pouco, tipo de centralização dessas

informações.

Comunicações Unificadas [1] são a convergência destas várias formas de comunicação como telefonia, *email*, mensagens instantâneas e vídeoconferência, permitindo que os usuários se comuniquem em tempo real com qualquer pessoa em qualquer lugar e, quando possível, com tecnologias equivalentes. Essas formas de comunicação podem ser divididas em síncrona, como telefone e VoIP, e assíncrona, como o *email*. Aplicações de mensagens instantâneas podem se enquadrar nos dois modelos: síncrono, ao enviar e receber as mensagens em um limite de tempo próximo a um bate-papo, ou assíncrono, onde os tempos de envio e recebimento são díspares, como o *email*. Assim, é possível que a partir de uma única ferramenta seja possível enviar um *email*, iniciar uma ligação ou chamar alguém para um bate papo, independente da forma que essa pessoa esteja acessando a rede - pois as comunicações unificadas procuram justamente a conversão entre essas várias ferramentas de comunicação.

Por ser um termo criado no mercado, a definição mais aceita para Comunicações Unificadas é a integração das comunicações visando otimizar os processos corporativos. Integração seria a interoperabilidade entre as formas de comunicações, e comunicações, por sua vez, envolve toda a forma de troca de informação, seja por MI, voz, dados ou imagem. Processos corporativos se referem à sequência de ações que geram um resultado, como um produto final, por exemplo. Otimizar é tornar esses processos mais eficientes obtendo mais e/ou melhores resultados.

Existem diversas empresas que comercializam soluções de Comunicações Unificadas; porém, ao analisá-los separadamente é possível verificar que existem significativas diferenças de funcionalidades entre eles. Ou seja, cada vendedor aplica o termo Comunicações Unificadas para se adequar ao seu produto particular. No mundo empresarial, o que realmente importa é o que o cliente quer, e hoje isso significa um sistema de comunicações rico e flexível que atenda às suas necessidades particulares. E embora haja uma certa padronização nos requisitos (de sistema), ainda assim essas necessidades variam, e muito, de cliente para cliente.

É importante ressaltar que com a integração de todas as formas de comunicação é possível gerar um ambiente de comunicação mais eficaz dentro das organizações, resultando em redução de custos, aumento de produtividade e melhoria da satisfação do cliente. Outro ponto interessante a ser lembrado é que Comunicações Unificadas não é um produto único, mas sim uma solução que pode variar de cenário para cenário e que envolve basicamente a conversão de protocolos e mídias. Este processo pode ser realizado através de produtos abertos ou soluções proprietárias. Como não há um padrão bem definido a maior dificuldade reside, portanto, no

[1] Que não é um termo tão novo, mas que apresenta um crescimento maior recentemente devido a novas e mais rápidas tecnologias de acesso, fixa e móvel, e enlaces com garantia de entrega, levando ao uso mais disseminado de Voz sobre IP (VoIP) e até conferências *online*

uso de padrões para a plena integração de produtos de fabricantes diferentes. No caso de uso de protocolos abertos, existem em especial dois que são amplamente utilizados: SIP e XMPP, com suas devidas extensões, os quais serão melhor vistos ao longo deste documento.

2.2 Serviços básicos

A *Gartner* (SHORETEL, 2009), conhecida empresa de consultoria e pesquisa da área de Tecnologia da Informação (TI), identificou em uma de suas pesquisas, conforme demonstrado abaixo, dezesseis funcionalidades que formam uma solução de Comunicações Unificada completa.

- **Presença:** Informação de disponibilidade, como por exemplo "Ocupado", "Em reunião", "Disponível" e outros. Maiores detalhes serão explicados na seção 2.2.1.
- **Telefonia:** Sistema de transmissão de áudio.
- ***Email*:** Possibilita o envio e recebimento de mensagens eletrônicas.
- **Cliente *Desktop*:** Software único que centraliza diversas funções, permitindo - por exemplo - realizar a troca de mensagens instantâneas ou iniciar uma ligação telefônica.
- **Mensagens Unificadas:** Possibilita a integração de tipos de mensagens diferentes, como por exemplo SMS[2] e *email*.
- **Mensagens Instantâneas:** Forma de comunicação em tempo real baseada em texto. Maiores detalhes serão explicados na seção 2.2.3.
- **Conferência de áudio:** Conferência entre três ou mais participantes onde apenas há o áudio destes participantes.
- **Conferência de vídeo:** Conferência entre três ou mais participantes onde além do áudio há o compartilhamento de vídeo, permitindo que se vejam ou compartilhem imagens, ou vídeos, como uma apresentação por exemplo.
- **Conferência *web*:** Conferência realizada utilizando uma rede de dados, como a internet. Pode ser feito apenas com a transmissão de áudio, bem como também vídeo, texto, compartilhamento de *desktop*, quadro de notas, entre outras opções.

[2]Serviço de mensagens curtas, muito utilizado em telefones celulares

- **Conferência convergente:** Possibilita a reunião de participantes que estejam usando meios diferentes de acesso a conferência. Por exemplo, pode ser possível a participação de usuários através de telefone enquanto outros estejam acessando através de um serviço de conferência *web*.

- **Serviço de notificação:** Possibilita a notificação a partir de determinados eventos. Por exemplo, ao receber uma ligação e a mesma não ser atendida, um *email* pode ser gerado informando do ocorrido, informando por exemplo quem ligou e horário.

- **Assistente pessoal:** Agenda onde estão disponíveis informações sobre os contatos. Maiores detalhes serão explicados na seção a 2.2.2.

- **Comunicações Integradas aos processos de negócio**: Dispara mensagens a partir de informações dos processos de negócio, como por exemplo quando o estoque ficar abaixo de um limiar.

- **Centro de contato:** Centraliza todas as informações de todos os contatos.

- **Soluções móveis:** Faz uso de celulares e outros dispositivos móveis facilitando o processo de comunicação.

- **Colaboração:** Soluções como compartilhamento de *desktop* ou edição de arquivos em conjunto.

Não há um consenso sobre quais serviços constituem uma solução de UC, conforme pode-se observar no trabalho (MAYER; POIKSELKA, 2009) onde um conjunto de serviços diferentes é demonstrado. Muitas empresas colocam uma ou outra das funcionalidades citadas em seus produtos e os vendem como uma solução de Comunicações Unificadas. A necessidade do cliente, pois, é quem diz se esse é produto completo para atender às suas necessidades.

Para algumas empresas, um colaborador ser alcançado na telefonia através de um único número, independente de onde ele se encontra, seja na sua mesa dentro da empresa, no seu *home office*, na rua com seu celular ou mesmo do outro lado do mundo utilizando um *softphone*, é um dos desafios na área da telefonia. Integrar tal funcionalidade a um sistema de correio de voz, onde as mensagens gravadas possam ser enviadas por *email*, ou, através de um serviço de notificação, um aviso de nova mensagem a ser enviada para o usuário através do comunicador de MI são funcionalidades de grande ajuda para a comunicação das pessoas. Assim como a conversão de emails em mensagens instantâneas e vice-versa ou dessas para áudio, de acordo com a disponibilidade do usuário, também são soluções que demonstram o poder de uma solução

de Comunicações Unificadas. Aliás, a disponibilidade do usuário, mais conhecida pelo termo de presença (que será melhor explicado mais adiante), se torna fundamental para o funcionamento de toda essa integração. Afinal, saber qual a melhor maneira para se contatar uma pessoa agiliza, e muito, a comunicação.

Há casos também de empresas, grande parte delas multinacionais, que têm como um grande problema seus gastos com viagens para reuniões e apresentações. Com o objetivo de redução de custos, e indiretamente utilizar a propaganda de empresa ecologicamente correta graças a economia de combustível gerada, uma solução que tem sido cada vez mais utilizada são as conferências pela *Internet*. Essas reuniões podem ser com apenas o áudio dos participantes ou também com a transmissão de vídeos, seja dos participantes da conferência como também pode ser as imagens de uma apresentação (*slides*). Podem ser realizadas através de equipamentos específicos para este fim, ou também utilizando os computadores dos participantes.

Por outro lado, há outros cenários onde há um grande volume de colaboradores trabalhando na rua, ou seja, visitando clientes ou constantemente em viagem. Para estas empresas, possibilitar o acesso externo a informações importantes internas, como documentos, contatos, agendas, através de aplicativos instalados nos celulares pode trazer muitos benefícios para ela.

Como foi visto, há diversos perfis de empresas - e suas respectivas necessidades. Apesar de empresas de consultoria especializada, como a *Gartner*, apresentar dezesseis funcionalidades que compõem uma solução de Comunicações Unificadas, isso não significa que uma empresa, ao adquirir uma solução dessas, estará adquirindo todas essas. Identificar as suas necessidades é um ponto fundamental para o sucesso da implementação. Este trabalho não irá tratar todas estas funcionalidades: o escopo do trabalho será o de entender e prover a interoperabilidade entre os diferentes protocolos abertos que os implementam, especialmente no que se refere a presença e mensagens instantâneas.

2.2.1 Presença

Com tantas formas de contato, perde-se cada vez mais tempo na decisão de qual forma utilizar para falar com alguém. E muitas vezes a forma escolhida não é a melhor maneira para se comunicar com aquele usuário naquele momento. Visando facilitar essa decisão existe o conceito de presença, um termo muito utilizado nas Comunicações Unificadas, que nada mais é que a informação da disponibilidade de um usuário e da melhor forma para se comunicar com ele neste momento. Essa informação de disponibilidade pode servir para fornecer a simples informação de disponível ou não, como também pode passar informações mais completas como as coordenadas obtidas por um *Global Positioning System* (GPS) informando a localização exata

do usuário. A partir destes dados, torna-se mais fácil a decisão da melhor maneira de se comunicar com o usuário, decisão essa que pode ser transparente para o usuário ao ligar para um ramal e o *Private Automatic Branch eXchange* (PABX) rotear [3] a ligação para o telefone fixo, móvel ou *softphone* de acordo com a informação de presença do destinatário.

É importante destacar a importância deste conceito em uma solução de Comunicações Unificadas. Dentre os dezesseis itens apresentados anteriormente, este é um conceito fundamental para o funcionamento dos demais. Não basta apenas realizar a integração entre todas as formas de contatos, se não for possível identificar qual dessas formas é a melhor para se contatar um usuário em um determinado momento. Com o uso da presença, a informação disponibilizada auxilia aos próprios elementos da rede rotearem da melhor forma possível a mídia que quer se transmitir a um contato, como por exemplo um áudio ou captura sequenciada da tela de um computador.

Com a evolução das redes de comunicação para uma arquitetura baseada em *Internet Protocol* (IP), surge uma facilidade de comunicação entre serviços de aplicação e seus dispositivos. Isso facilita o avanço dos serviços de presença, que são fortemente utilizados por serviços de MI, para utilização em conjunto desta informação entre redes e dispositivos diferentes como computadores, telefone móveis e fixos, aparelhos de GPS e diversos outros equipamentos. Em resumo, o conceito de presença pode se tornar uma importante ferramenta para os usuários gerenciarem seu tempo e suas comunicações com mais eficiencia.(SOLUTIONS, 2000)

Para o correto funcionamento do serviço de presença o servidor responsável por obter e divulgar estas informações deve conseguir se comunicar com todas as formas possíveis que este usuário pode disponibilizar esta informação como, por exemplo através de um ramal em um PABX, de um telefone celular através de uma rede *General Packet Radio Service* (GPRS) ou através de um dos diversos serviços de mensagem instantânea disponíveis. Outra característica é a centralização destas informações, possibilitando a todos os serviços o devido acesso.

Verifica-se então a necessidade de interoperabilidade entre o servidor responsável por este serviço de presença com os diversos serviços que fornecem essa informação. Como não há no mercado uma empresa que tenha produtos e serviços que atendam todas as formas de comunicação, surgirão problemas como a incompatibilidade de protocolos caso cada fabricante utiliza um protocolo proprietário.(SOLUTIONS, 2000) Esse problema pode ocorrer também com a utilização de protocolos abertos, já que existe mais de um com essa capacidade. É interessante, inclusive, mencionar o Sistema de Sinalização número 7 (SS7)(UNION, 1993) e as redes *Next Generation Network* (NGN)(WILKINSON, 2002) que, para comportar os diversos

[3] Identificar a melhor rota ao destino

padrões já existentes, tornaram-se bastante complexas em teoria/implementação.

Antes de abordar os protocolos utilizados, é interessante apresentar um modelo genérico de serviço de presença. Como resultado do trabalho do Grupo de Trabalho de Redes da *Internet Engineering Task Force* (IETF), foi publicado a RFC 2778(DAY; ROSENBERG; SUGANO, 2000) que tem como objetivo demonstrar um modelo abstrato de um Serviço de Presença e Comunicações Instantâneas. Nele são definidos as várias entidades envolvidas, definidas terminologias e descrito os serviços prestados por estes sistemas.

Um serviço de presença basicamente possui dois tipos de clientes: *PRESENTITIE* - que é quem está provendo a informação de presença - e *WATCHERS* - que é quem recebe a informação de presença. É importante ressaltar que isso é apenas um modelo, e que esses dois "clientes" não são necessariamente implementados separadamente.

Dentro deste modelo existem dois tipos de *WATCHERS*: o *FETCHER* que simplesmente solicita a situação atual de presença de um *PRESENTITIE* e o *SUBSCRIBER* que assina as futuras atualizações de um *PRESENTITIE* e receberá uma notificação assim que houver alguma alteração na sua informação de presença. Um tipo especial de *FETCHER* é o *POLLER*, que periodicamente solicita a informação de presença de um *PRESENTITIES*.

Alterações da informação de presença serão distribuídas aos *SUBSCRIBERS* através de notificações. Isto ocorre com o *PRESENTITIE* alterando sua informação de presença e informando ao seu serviço de presença dessa alteração. Esse serviço então se encarregará de informar a todos os *SUBSCRIBERS* deste *PRESENTITIE* sobre a alteração da informação de presença deste. Este serviço também armazenará esta informação para repassá-la a um *FETCHER* assim que for solicitado. Esta é a visão geral de funcionamento de um serviço de presença

Outro trabalho realizado pelo mesmo Grupo de Trabalho da IETF resultou na RFC 2779(DAY et al., 2000), a qual propõe requisitos mínimos que protocolos que implementem serviços de Mensagem Instantânea e/ou Presença devem possuir. Utilizando estes requisitos mínimos e seguindo o modelo de serviço de presença apresentado anteriormente há dois principais protocolos abertos e com bastante aceitação: o XMPP, definido nas RFCs 3920(SAINT-ANDRE, 2004b) e 3921(SAINT-ANDRE, 2004a), e o SIMPLE, que é um conjunto de extensões do protocolo SIP(ROSENBERG; CAMARILLO et al., 2002).

2.2.2 Agenda

Uma empresa ou instituição de ensino, pode, se assim o quiser, centralizar em um ponto central todos os contatos de seus colaboradores, ou alunos, bem como sua informação de

presença, o que traz enormes benefícios, seja para consulta manual dos próprios usuários por informações de outros usuários que desejam contatar, seja para integrar as diversas aplicações usadas para comunicação entre as pessoas.

Pelo fato de em uma solução de UC existirem vários aplicativos integrados, centralizar as informações comuns a todos resulta em um melhor funcionamento da solução, já que todas as informações estão disponíveis em um mesmo lugar, evitando informações conflitantes e/ou vencidas. Porém, como este é um serviço que deverá ter interface com praticamente todos os demais existentes, a necessidade de conseguir comunicar-se com diversos protocolos é extremamente alta.

A principal informação disponibilizada será a de presença, seguido das informações de contato dos usuários. Assim, uma aplicação que deseja contatar um usuário irá consultar nesta agenda qual a disponibilidade deste usuário, ou seja, qual a melhor forma de contato com ele. De posse desta informação a aplicação deverá tentar contatar o usuário utilizando esta melhor forma de contato, onde pode surgir a necessidade da conversão de formatos.

2.2.3 Mensagem Instantâneas

O serviço de mensagems instantâneas consiste na troca de mensagens de texto, de forma síncrona ou assíncrona, entre duas ou mais pessoas através de aplicativos específicos - rodando em computadores ou celulares - em rede. Através deste aplicativo e de um endereço de identificação é possível a inserção, ou remoção, de usuários em uma lista de amigos, permitindo uma comunicação entre usuário conhecidos. Nota-se aqui a necessidade de informar todos os clientes de mudanças remotas, como modificação do estado de presença de cada amigo na lista de contatos, e o modelo *PubSub*(SAINT-ANDRE; MILLARD; MEIJER, 2010) se mostra interessante como forma de implementação. Aém disso, há as salas de bate-papo, as quais qualquer usuário pode entrar e conversar com todos os demais que estiverem nesta mesma sala. A estrutura básica destas duas formas consistem em aplicativos clientes nas pontas e um servidor central que redireciona as mensagens entre os usuários.

Além de causar uma economia expressiva em telefonia, a comunicação via texto aumenta significantemente a produtividade. A telefonia convencional é uma atividade monotarefa. Através de mensagens instantâneas, é possível conversar com várias pessoas “ao mesmo tempo”: mantendo várias conversas “em paralelo”.

O conceito de presença apresentado anteriormente é bem comum nas aplicações de mensagens instantâneas. Uma informação básica que todo aplicativo apresenta é se os usuários

da sua lista de amigos estão em rede (*online*) ou não. A partir desta informação já se sabe que o usuário, uma vez estando disponível, poderá conversar de forma síncrona. Porém, mais informações podem ser agregadas, onde o usuário pode especificar se está disponível naquele momento ou se está ocupado, o lugar onde se encontra no momento (casa, trabalho, rua), e até informações como a música que está ouvindo no momento ou o sítio (*site*) que está visitando. A todo momento, como se pode perceber, cada cliente precisa informar aos outros de mudanças de estado.

Softwares de MI podem agregar mais funcionalidades tornando a experiência de seu uso mais rica. Opções como a visualização dos usuários através de uma *webcam*, bate-papo utilizando microfone e caixas de som, transferência de arquivos, entre outras, permitem uma comunicação mais eficaz trazendo vários benefícios para os usuários, seja para o uso doméstico como também para o empresarial. Porém, a crescente preocupação com privacidade e segurança da informação vai de encontro à utilização de serviços de terceiros, como por exemplo o Skype. Algumas empresas, não querendo desperdiçar as vantagens trazidas por estas aplicações, criam soluções como a instação de um servidor próprio de MI. Desta forma elas tem o controle total sobre quem usa e as mensagens que são trocadas. Aqui, novamente a dupla XMPP e o SIP/SIMPLE aparecem na tentativa de criar padrões abertos para suprir esta necessidade. No caso do XMPP, a RFC 3921 padroniza o seu funcionamento, e para o SIP existe a RFC 3428(ROSENBERG; HUITEMA et al., 2002) que introduz o método *MESSAGE* para a troca de mensagens sem sessão, e as RFCs 4975(JENNINGS; CAMPBELL; MAHY, 2007) e 4976(JENNINGS; MAHY; ROACH, 2007) que padronizam o *Message Session Relay Protocol* (MSRP). Mais uma vez, a questão de interoperabilidade vem a tona devido a necessidade do funcionamento em conjunto da solução de MI com os demais protocolos de MI, bem como com as demais ferramentas de comunicação que as empresas possuem, inclusive a propagação da informação de presença.

2.2.4 Email

O *Electronic Mail*, ou correio eletrônico, consiste na troca de mensagens digitais. São baseados em um modelo de armazena-e-encaminha (*store-and-forward*, semelhante ao IP), no qual um servidor aceita, encaminha, armazena e entrega mensagens aos usuários que basicamente buscam suas mensagens em um servidor. Uma mensagem de *email* consiste de duas partes: o cabeçalho com informações de controle, com no mínimo a informação de remetente e destinatários, e do corpo que é a mensagem a ser enviada propriamente dita . O protocolo de envio,

Simple Mail Transfer Protocol (SMTP), está definido na RFC 821(POSTEL, 1982)[4]. Seu funcionamento é completamente semelhante a um sistema de correios, onde um remetente escreve uma mensagem a endereçando para um destinatário e com base no endereço do destinatário os servidores vão encaminhando até que ela chegue na caixa postal de destino, e fica a cargo de outros protocolos, como *POP*(MYERS; ROSE, 1996) e *IMAP*(CRISPIN, 2004), implementarem a leitura por parte do destinatário. No corpo da mensagem é posssível o envio de texto simples como também há a possibilidade da transferência de arquivos. Esta é uma grande vantagem deste sistema visto que esta transferência de arquivos ocorre de uma maneira assíncrona, ou seja, o destinatário não precisa estar online no momento da transferência, podendo receber a mensagem, e o arquivo, a qualquer momento.

No contexto das Comunicações Unificas a utilização do Email pode ser interessante dependendo da disponibilidade do usuário. Caso alguém deseje transmitir uma mensagem e anexar a ela um documento importante para um usuário que não está disponível para MI no momento, há a possibilidade de realizar este envio por email para este usuário. O servidor de email deste usuário, ao receber esta mensagem, pode avisar o usuário através de um *Short Message Service* (SMS), ou até mesmo realizando uma ligação para o usuário com uma mensagem prégravada, ou utilizando algum software de *Text-to-Speech* que "leia" o email para o usuário.

2.2.5 Conversão de formatos

A conversão de formatos é o que permite a integração e interoperabilidade entre as diversas formas de comunicação. Consiste basicamente na conversão de protocolos de sinalização ou de codecs de mídia, permitindo que uma mensagem enviada a partir de um cliente de um protocolo possa ser entregue para um cliente de outro protocolo ou que uma mídia codificada com um formato seja entregue com uma codificação em um formato diferente. Para que isso ocorra, existe a figura do *gateway*, responsável por esta conversão que, a partir do mapeamento de funções semelhantes entre os dois protocolos - ou as duas mídias, realiza a conversão entre eles.

É possível converter a sinalização da mídia ou mesmo a mídia em si. O foco deste trabalho é, inclusive, analisar a interoperabilidade entre os protocolos de sinalização, onde foram escolhidos para fins de estudo SIP e XMPP. Já existem, inclusive, propostas (*drafts*)(SAINT-ANDRE; HOURI; HILDEBRAND, 2009) que procuram resolver a essa questão mapeando os protocolos XMPP e SIP para que ambos possam trocar informações de presença entre si. Porém a questão de presença não é a única pendência para a total interoperabilidade entre eles: lista de contatos, sinalização para início e término de sessões que envolvam a transferência de mídias e

[4]Criado no ano de 1982. Um email enviado naquela época é muito semelhante ao enviado atualmente

transferência de arquivos são problemas que devem ser estudados para encontrar possibilidades de uso de ambos os protocolos em conjunto, ampliando o leque de soluções aos usuários.

Basicamente, a conversão de protocolos de sinalização se dá com o mapeamento de campos comuns de ambos os protocolos. Apesar de parecer uma ação simples, alguns protocolos possuem campos que outros não possuem, impedindo que haja uma relação direta entre cada elemento - o ideal para haver interoperabilidade seria uma função bijetora entre os dois conjuntos. Criar formas de contornar estes obstáculos podem auxiliar para uma melhor integração entre dois ambientes. Porém, um ponto importante deve ser observado: esta conversão deve ser totalmente transparente ao usuário, e isto inclui o tempo total de conversão e seu custo de processamento, o qual pode se mostrar proibitivo na prática. Um alto tempo de conversão pode degenerar a experiência do usuário, ocasionando a desistência de uso da ferramenta devido a atrasos excessivos nas operações. Como o intuito de uma solução de UC é facilitar e agilizar as comunicações dos usuários, esse não é o resultado esperado.

2.2.6 Histórico de tráfego (texto, voz, vídeo...)

O histórico do tráfego gerado é uma excelente forma de manter uma associação direta entre as informações e a linha temporal para futuras consultas bem como para auditorias de uma empresa. Graças a gravação é possível por exemplo inserir um terceiro usuário em uma conversação (MI, email..) e este obter todo o histórico do que já foi conversado, ganhando agilidade visto que não será necessário parar essa conversação para situar este novo usuário, permitindo maior agilidade e eficiencia. Todos estes serviços apresentados até agora podem ser configurados de diversas formas através de protocolos diferentes. Em uma implementação de uma solução de UC, a integração destes serviços - protocolos - é de vital importância para o sucesso da mesma. No decorrer deste trabalho será visto o funcionamento destes protocolos - principalmente para os serviços de presença e MI - e a interoperabilidade entre eles.

3 *Protocolos*

Segundo Kurose e Ross (2006) “Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou recebimento de uma mensagem ou outro evento.” O uso difundido e a expansão dos protocolos de comunicação é ao mesmo tempo um pré-requisito e uma contribuição para o poder e sucesso da *Internet*. O par formado por IP e *Transmission Control Protocol* (TCP) é uma referência a uma coleção dos protocolos mais utilizados. A maioria dos protocolos para comunicação via *Internet* é descrita nos documentos RFC do IETF.

Contudo, após a elaboração e divulgação de um protocolo, pode ser necessário ou mesmo interessante a implementação de novas funções que agregam valor ao protocolo original. Para evitar alterações constantes aos protocolos, os quais podem já contar com extensa base instalada de equipamentos e produtos, são criados as extensões de protocolos, que definem regras, funções ou mensagens adicionais para o protocolo original.

Para este trabalho, serão estudados apenas os protocolos abertos SIP e XMPP, ambos definidos em RFCs, e suas extensões. Para presença, há o XMPP, definido nas RFCs 3920(SAINT-ANDRE, 2004b) e 3921(SAINT-ANDRE, 2004a), e a extensão para SIP chamada SIMPLE é uma extensão do protocolo SIP . Existem também soluções proprietárias lançadas por empresas interessadas neste mercado, como Cisco (Webex), Adobe (Adobe ConnectNow), Avaya (Aura) e Microsoft (Exchange Server), que podem ou não fazer uso de protocolos abertos, porém não ficam disponíveis para estudo ou integração.

A escolha do protocolo, ou da solução, ideal para cada empresa depende de vários fatores. Soluções proprietárias são caras e dificilmente possibilitam a comunicação entre equipamentos de fabricantes diferentes. Soluções abertas, gratuitas, são complexas e exigem um conhecimento técnico maior para sua instalação e manutenção, mas estão em um nível aceitável de implementação em termos de conectividade e conversão de formatos tornando-as fortes candidatas a serem escolhidas comparadas as soluções proprietárias.

3.1 SIP

O SIP, definido na RFC 3261, é um protocolo da camada de aplicação, que utiliza o modelo ”requisição-resposta” utilizado para a criação, modificação e finalização de sessões multimídia entre usuários. É semelhante ao protocolo HTTP, sendo também baseado em texto. Possui seu código aberto e flexível, funcionando em uma arquitetura cliente/servidor.

O SIP atua como um protocolo de sinalização de nível de aplicação. Ele negocia os termos e as condições de uma sessão, além de auxiliar na localização dos participantes da mesma.(COLCHER; GOMES, 2005) É composto por quatro principais componentes, sendo eles:

3.1.1 SIP User Agents

O *User Agent* é basicamente a entidade responsável por enviar e receber requisições, agindo como cliente (*USer Agent Client* (UAC)) quando envia requisições ou recebe resposta ou como servidor (*User Agent Server* (UAS)), enviando respostas e recebendo requisições.

É a entidade que interage com o usuário, possibilitando que o usuário crie sessões para transmissão de mídia. Na origem, o *User Agent* (UA) do usuário envia as mensagens necessárias para o estabelecimento de uma sessão e irá tratar as respostas destas requisições. No recebimento, o usuário poderá além da opção de aceitar a chamada, rejeitá-la ou transferi-la para outro lugar. De acordo com a opção escolhida pelo usuário o UA irá responder a requisição com as mensagens necessárias para cumprir o que foi solicitado.

Existem situações que o UA não precisará interagir diretamente com o usuário, agindo de acordo com configurações automatizadas. Um usuário pode programar para que, ao receber um pedido de sessão, esse seja transferido para uma secretária eletrônica. Esta secretária eletrônica possuirá um UA que irá negociar a sessão sem uma intervenção manual do usuário. Outro exemplo, agora originando uma chamada, é a de uma programação de despertador: um usuário programa uma chamada para ocorrer em um determinado horário.

Outra função do UA é encaminhar as mídias para as devidas ferramentas. Nas mensagens trocadas entre um UAC e um UAS, a descrição da mídia utilizada (áudio, vídeo) é uma das informações negociadas. De posse desses dados, o UA sabe para qual ferramenta encaminhar a mídia trocada.

3.1.2 SIP Redirect Server

Sua função é auxiliar na localização de UA. Como será visto mais adiante, o endereço SIP é semelhante ao do *email*: `usuário@domínio.com`. Ao tentar iniciar uma sessão, o UA irá fazer uma requisição ao endereço `usuário@domínio.com`.

Um servidor *redirect server* atende às requisições do domínio `dominio.com` e responderá a requisição informando a localização do usuário. É importante ressaltar que o *redirect server* apenas informa a possível localização de determinado usuário e não faz o encaminhamento das requisições. O *redirect server* também não inicia nenhuma ação para localizar o usuário, ele apenas responde sugerindo um servidor que provavelmente terá melhores informações de localização do usuário.

Uma função interessante que pode ser habilitada em um *redirect server* é o endereço de grupo. O *redirect server* poderá ter respostas diferentes para uma mesma requisição de um mesmo endereço de acordo com, por exemplo, o horário da requisição.

3.1.3 SIP Proxy Server

Servidor intermediário que atua tanto como cliente como servidor. Ao contrário do *redirect server* que responde as requisições com os endereços de onde o usuário pode estar, o *Proxy Server* ao receber uma requisição faz a intermediação entre as pontas envolvidas na sessão. Pode haver mais de um *proxy server* entre os usuários de uma sessão. O *proxy server* pode restringir as sessões que os usuários podem criar, bem como também podem alterar as mensagens SIP trocadas. Serviço de bilhetagem também pode ser executado.

Existe dois tipos: o *Stateful Proxy Server*, que mantém o estado das transações e permite dividir a chamada para múltiplos servidores na tentativa de localizar o usuário criando uma árvore de busca e possuindo maior confiabilidade. Possui a capacidade de computar o gasto do cliente e utiliza o protocolo TCP. E o *Stateless*, que não armazena o estado da transação e envia adiante as requisições e respostas, possuindo uma maior velocidade porém com menor confiabilidade. Como há o tratamento da informação (mensagem SIP), é possível associar ao *proxy server* um *gateway* para conversão para XMPP.

3.1.4 SIP Registrar Server

Servidor que armazena registros sobre usuários, fornecendo um serviço de localização. Trabalha em conjunto com o *Redirect* e o *Proxy server*.

3.1.5 Mensagens SIP

Conforme Colcher e Gomes (2005), as mensagens SIP podem ser requisições ou respostas. Como o protocolo é baseado em texto, essas mensagens são construídas com base no conjunto de caracteres *UTF-8*. Suas mensagens consistem em um cabeçalho e o corpo com a informação. A primeira linha do cabeçalho define o método (tipo de operação de requisição) sendo os principais listados na tabela 3.1 - e de acordo com o escopo definido neste trabalho:

Método	Objetivo
INVITE(RFC3261)	Pedido de estabelecimento de conexão
ACK(RFC3261)	Reconhecimento do INVITE pelo receptor final da mensagem
BYE(RFC3261)	Término da sessão
CANCEL(RFC3261)	Término de uma conexão ainda não estabelecida
REGISTER(3261)	Registro do User Agent Client no SIP Proxy
OPTIONS(RFC3261)	Pedido de opções do servidor
REFER(RFC3515)	Faz a transferência de uma ligação SIP
SUBSCRIBE(RFC3265)	Pedido para receber notificações de eventos
NOTIFY(RFC3265)	Envia notificações de eventos
INFO(RFC2976)	Envia diversas mensagens (Ex. DTMF)
PUBLISH(RFC3903)	Publica notificações de presença
MESSAGE(RFC3428)	Envio de mensagens instantâneas

Tabela 3.1: *Métodos SIP*

Os métodos destacados na tabela 3.1 são utilizados nos serviços apresentados no capítulo 2, especialmente em implementações de presença e mensagens instantâneas. Seu uso será melhor descrito no capítulo 4.

As respostas são semelhantes as do protocolo HTTP, sendo as mais importantes demonstradas na tabela 3.2.

O cabeçalho é uma sequência estruturada de campos que podem ser incluídos em mensagens de requisição ou resposta, oferecendo mais informações sobre a mensagem ou indicando seu tratamento apropriados. A obrigatoriedade desses campos é dependente do tipo da mensagem.(COLCHER; GOMES, 2005)

A forma como uma mensagem SIP é montada é basicamente no formato `campo:valores` e não são sensíveis a caixa. Sua codificacão é em modo texto puro, o que facilita a análise das mensagens por administradores de sistemas, porém implica uma carga maior do que se fosse utilizado codificação binária. Cada campo, sendo alguns obrigatórios em algumas mensagens e outros opcionais, funciona como uma variável que pode assumir diversos valores. Caso um campo não seja entendido na recepção ele é simplesmente ignorado.

Código	Significado
1XX	Mensagens de informação
100	Tentando
180	Campainha
182	Progresso
2XX	Sucesso
200	Ok
3XX	Encaminhamento de chamada, o pedido deve ser direcionado para outro lugar
302	Movido temporariamente
305	Use proxy
4XX	Erro
403	Proibido

Tabela 3.2: *Mensagens de resposta SIP*

Uma requisição básica possui uma estrutura parecida com a da figura 3.1 , composta por um método de requisição, cabeçalhos obrigatórios e opcionais, uma linha em branco e um espaço para mensagens. A seguir serão destacados alguns pontos importantes do protocolo que serão utilizados mais adiante no trabalho.

```
1  REGISTER sip:ekiga.net SIP/2.0
2  CSeq: 2 REGISTER
3  Via: SIP/2.0/UDP 189.85.182.196:5063;branch=z9hG4bK1a62c281-b1c1-de11-9053-000
       c29b1de01;rport
4  User-Agent: Ekiga/2.0.12
5  Authorization: Digest username="petrybr", realm="ekiga.net", nonce="4
       ae8ea1f000010151926635e5af2e7fe764c90dc8cd6044b", uri="sip:ekiga.net",
       algorithm=md5, response="efcd919c39a920ef0d5016863479c25c"
6  From: <sip:petrybr@ekiga.net>;tag=10659281-b1c1-de11-9053-000c29b1de01
7  Call-ID: 6ecb7381-b1c1-de11-9053-000c29b1de01@JP
8  To: <sip:petrybr@ekiga.net>
9  Contact: <sip:petrybr@189.85.182.196:5063;transport=udp>
10 Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE
11 Expires: 3600
12 Content-Length: 0
13 Max-Forwards: 70
```

Figura 3.1: Exemplo do método REGISTER

A primeira linha, que contém o método, é composta por três partes onde a primeira identifica o método, a segunda a *Uniform Resource Indicator* (URI)(FIELDING; MASINTER et al., 2005) e a terceira a versão do SIP utilizada. O método identifica o tipo de requisição, a URI identifica o próximo nó ao qual a mensagem deve ser enviada e a versão é utilizada para fins de compatibilidade. No exemplo o método utilizado é o *REGISTER*, ou seja, é um pedido de registro e a versão do SIP é a *2.0*. Este pacote é direcionado ao servidor *ekiga.net*.

Até chegar ao destinatário a mensagem será acrescida de um campo `Via` contendo o endereço de cada elemento (os SIP *proxy servers*) por onde a mensagem passou. Isto facilita a detecção de um laço fechado (*loop*), caso a mensagem chegue em um elemento e este identifique que já existe um campo `Via` com seu endereço. Ao detectar um *loop* uma mensagem de resposta 482 (*Loop* detectado) é enviada. Esta implementação garante também que uma resposta percorrerá o mesmo caminho da mensagem original. Neste campo também é identificado o protocolo de transporte utilizado e opcionalmente a porta na qual o cliente aguarda a resposta.

O campo `Max-Forwards` é utilizado em todos os métodos e serve para limitar o número de saltos que essa mensagem poderá dar até o destinatário, evitando que a mensagem entre em *loop* e sobrecarregue a rede. Pode ser muito útil na tentativa de identificar uma falha que esteja ocorrendo, como o própio *loop*. Seu valor é um número inteiro que pode variar de `0` a `255` que indica quantos saltos - ou quantas vezes a mensagem ainda pode ser encaminhada.

O campo `To` define para quem a requisição é direcionada e usualmente contem o endereço público. e o `From` é quem está enviando a solicitação.

Já o campo `Call-ID` é composto pela combinação de um número aleatório com o IP, ou *host*, do transmissor formando um identificador único para a mensagem. Este valor é único para uma sessão, servindo para identificá-la para uso em sistemas de tarifação por exemplo. Esse campo, por exemplo, pode ser transcodificado, mantendo o seu valor original (apenas mudando o nome do campo), ou o conversor atuará como *gateway* mesmo: para cada "ponta" da transmissão, será criado um *ID* por ligação e uma tabela de relacionamentos. Embutir XML(BRAY; PAOLI et al., 2008) dentro do SIP, por exemplo, pode levar a uma transcodificação sem mudança do *Call-ID*, o que é bastante interessante se for pensar em tarifação e outros serviços[1].

O campo `CSeq` é formado por um número inteiro e o nome do método de requisição. Quando uma transação é iniciada este campo recebe um número aleatório. A cada nova mensagem esse número é acrescido de um, permitindo um controle sobre a perda de pacotes, ou a entrega de mensagem fora de ordem. Este campo facilita a organização de diferentes requisições para uma mesma sessão, permitindo identificar para qual requisição uma resposta recebida corresponde. É interessante destacar a fraca relação entre o SIP e o protocolo de transporte, podendo inclusive rodar sobre *User Datagram Protocol* (UDP), TCP ou mesmo *Stream Control Transmission Protocol* (SCTP) sem qualquer prejuízo às mensagens trafegadas (*payload*).

Como já foi comentado, após o cabeçalho uma linha em branca é inserida para em seguida o corpo da mensagem aparecer. A interpretação do corpo da mensagem varia de acordo com o método da própria mensagem. Nele pode aparecer desde parâmetros adicionais que

[1] O XMPP também usa XML, outro ponto facilitador de uma transcodificação

não são adicionados no cabeçalho, como também mensagens do protocolo *Session Description Protocol* (SDP). Através do SDP, que é especificado na RFC 4566(HANDLEY; JACOBSON; PERKINGS, 2006) e explicado mais adiante, os participantes de uma sessão negociam a mídia que será utilizada bem como as respectivas informações para a transmissão dessa mídia.(COLCHER; GOMES, 2005) No decorrer do trabalho será visto que no corpo da mensagem do protocolo SIP podem ser enviados mensagens de outros protocolos, como o XMPP.

Uma mensagem de resposta para uma requisição é composta por uma linha de *status*, cabeçalhos obrigatórios e opcionais, uma linha em branco e um espaço para mensagens. O cabeçalho da resposta para o método *REGISTER* demonstrado anteriormente é apresentado na figura 3.2:

```
1 SIP/2.0 200 OK
2 CSeq: 2 REGISTER
3 Via: SIP/2.0/UDP 189.85.182.196:5063;branch=z9hG4bK1a62c281-b1c1-de11-9053-000
      c29b1de01;rport=5063
4 From: <sip:petrybr@ekiga.net>;tag=10659281-b1c1-de11-9053-000c29b1de01
5 Call-ID: 6ecb7381-b1c1-de11-9053-000c29b1de01@JP
6 To: <sip:petrybr@ekiga.net>;tag=c64e1f832a41ec1c1f4e5673ac5b80f6.f285
7 Contact: <sip:petrybr@189.85.182.196:5063;transport=udp>;expires=1200
8 Server: Kamailio (1.4.0-notls (i386/linux))
9 Content-Length: 0
```

Figura 3.2: Exemplo de mensagem de resposta

A linha de *status* é composta por três campos: versão do protocolo, código do *status* e o *status*. No exemplo ela mostra que a versão do SIP utilizada é a 2.0 e o *status* é 200, que conforme a tabela 3.2, significa que a requisição foi aceita (ou que o *status* é OK). A resposta do pedido foi, portanto, respondida na própria aplicação, uma funcionalidade encontrada em aplicações que rodam sobre protocolos não orientados a conexão como UDP. Se fosse uma aplicação específica para TCP, por exemplo, a aplicação poderia simplesmente confiar no protocolo da camada inferior.

Quando um cliente deseja iniciar uma sessão, seja de voz, vídeo ou ambos, ele encaminha uma mensagem *INVITE*. Esta requisição solicita a abertura da sessão, conforme demonstrado na figura 3.3, podendo ser aceita ou rejeitada através das mensagens de respostas demonstradas anteriormente. No corpo desta mensagem pode ser inserido o protocolo SDP, que irá gerenciar informações sobre a sessão (*codec* que será utilizado, por exemplo).

O método *MESSAGE*, definido na RFC 3428(ROSENBERG; HUITEMA et al., 2002), é uma extensão do protocolo SIP, introduzindo a possibilidade de troca de mensagens instantâneas entre usuários usando o SIP. O texto da mensagem é inserido no corpo do método

```
INVITE sip:jpaulo@sip.uc.com SIP/2.0
     Via: SIP/2.0/UDP pc.boi.com;branch=z9hG4bK776asdhds
     Max-Forwards: 70
     To: Jose Paulo <sip:jpaulo@sip.uc.com>
     From: Ederson <sip:ederson@boi.com>;tag=1928301774
     Call-ID: a84b4c76e66710@pc.boi.com
     CSeq: 314159 INVITE
     Contact: <sip:ederson@pc.boi.com>
     Content-Type: application/sdp
     Content-Length: 142

[SDP]
```

Figura 3.3: Exemplo do método *INVITE*

MESSAGE, como pode ser visto na figura 3.4.

```
MESSAGE sip:user2@domain.com SIP/2.0
 Via: SIP/2.0/TCP user1pc.domain.com;branch=z9hG4bK776sgdkse
 Max-Forwards: 70
 From: sip:user1@domain.com;tag=49583
 To: sip:user2@domain.com
Call-ID: asd88asd77a@1.2.3.4
 CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 26

 Por favor, venha aqui.
```

Figura 3.4: Exemplo do método *MESSAGE*

Como foi visto, o SIP serve para iniciar, modificar e encerrar uma sessão. Nele não trafegam pacotes com as mídias propriamente ditas. Aliás, o transporte de mídias não é uma tarefa fácil para protocolos como o TCP e o UDP. Para estes tipos de dados qualquer atraso no recebimento de pacotes não é aceitável, tornando protocolos confiáveis e largamente utilizados ineficientes. O TCP, por exemplo, é altamente confiável visto que o mesmo envia uma confirmação para cada pacote recebido e garante a entrega destes pacotes na ordem correta. Porém o tempo de espera para o recebimento de um pacote que teve que ser retransmitido impossibilita o uso deste protocolo em aplicações como chamadas telefônicas, onde qualquer atraso é perceptível e incômodo. Já o UDP é o protocolo mais utilizado para a transmissão dessas mídias, embora seja demasiadamente "simples" para esse tipo de aplicação. Outro protocolo, que foi criado para este fim, é o SCTP que é utilizado pelo *3rd Generation Partnership Project* (3GPP) nas especificações de redes de celular e será explicado mais adiante.

3.1.6 SIMPLE

O *SIP for Instant Messaging and Presence Leveraging Extensions* (SIMPLE) é um grupo de trabalho criado pela IETF com o objetivo de criar extensões ao protocolo SIP de modo que esse possa oferecer soluções de Presença e Mensagens Instantâneas, já que originalmente o SIP não possui suporte a estas funcionalidades. Como resultado, novas RFCs foram publicadas agregando novos serviços ao SIP.

No caso da presença, foi necessário criar soluções para lidar com as assinaturas, bem como com as notificações e publicações. Desta forma a RFC 3265(ROACH, 2002) define os métodos *SUBSCRIBE* e *NOTIFY*, onde o primeiro permite a inscrição a eventos e o segundo a notificação de novos eventos aos assinantes. Já na RFC 3903(NIEMI, 2004) o método *PUBLISH* é introduzido, o qual permite que UAs informem suas alterações de presença. As informações de presença são codificadas em *eXtensible Markup Language* (XML) e são transportadas no corpo das mensagens SIP, fugindo do padrão de codificação do SIP - que como será visto possui um overhead menor que o XML.

O método *SUBSCRIBE*, exemplificado na figura 3.5, é utilizado para se inscrever nas atualizações de um usuário ou serviço. Estas atualizações são enviadas instantâneamente aos inscritos utilizando o método *NOTIFY*. Para cancelar a assinatura o mesmo método é utilizado, porém utilizando no cabeçalho o valor zero para campo `Expires`.

```
1  SUBSCRIBE sip:presentity@uc.comSIP/2.0
2  Via: SIP/2.0/UDP host.uc.com;branch=z9hG4bKnashds7
3  To: <sip:presentity@uc.com>
4  From: <sip:watcher@uc.com>;tag=12341234
5  Call-ID:12345678@host.uc.com
6  CSeq: 1 SUBSCRIBE
7  Max-Forwards: 70
8  Expires: 3600
9  Event: presence
10 Contact: sip:user@host.uc.com
11 Content-Length: 0
```

Figura 3.5: Exemplo do método SUBSCRIBE

Como falado o método *NOTIFY*, definido na mesma RFC do método *SUBSCRIBE*, serve para enviar notificações de atualizações a quem solicitou (se inscreveu) recebê-las. Na figura 3.6 é apresentado um exemplo de mensagem do método *NOTIFY*, onde pode-se observar a transmissão da informação de presença - codificada em XML - no corpo da mensagem.

Já o método *PUBLISH*, demonstrado na figura 3.7, é utilizado para a divulgação de uma atualização de um cliente para um servidor que gerencia as assinaturas deste cliente. Ao receber

```
NOTIFY sip:user@host.uc.com SIP/2.0
Via: SIP/2.0/UDP pa.uc.com;branch=z9hG4bK4cd42a
To: <sip:watcher@uc.com>;tag=12341234
From: <sip:presentity@uc.com>;tag=abcd1234
Call-ID:12345678@host.uc.com
CSeq: 2 NOTIFY
Max-Forwards: 70
Event: presence
Subscription-State: active; expires=3400
Contact: sip:pa.uc.com
Content-Type: application/pidf+xml
Content-Length: ...

[PIDF Document]
```

Figura 3.6: Exemplo do método NOTIFY

um *PUBLISH* o servidor verifica quem assina e gera um *NOTIFY* informando da atualização ocorrida.

```
PUBLISH sip:presentity@uc.com SIP/2.0
Via: SIP/2.0/UDP pua.uc.com;branch=z9hG4bK652hsge
To: <sip:presentity@uc.com>
From: <sip:presentity@uc.com>;tag=1234wxyz
Call-ID:81818181@pua.uc.com
CSeq: 1 PUBLISH
Max-Forwards: 70
Expires: 3600
 Event: presence
Content-Type: application/pidf+xml
Content-Length: ...

[PIDF Document]
```

Figura 3.7: Exemplo do método PUBLISH

Session Description Protocol

Durante a negociação de uma sessão, sejá no início ou durante a mesma, informações sobre as mídias que serão utilizadas podem ser trocadas no corpo das mensagens SIP utilizando o protocolo SDP. Essas informações indicam basicamente o tipo de mídia que será utilizado, o protocolo de transporte, porta lógica da conexão, *codecs*. Estas informações possuem o mesmo formato do SIP: `tipo=valor`. Na tabela 3.3 são apresentados algumas das informações que podem ser transmitidas:

Na figura 3.8 é demonstrado um exemplo de negociação. Inicialmente uma mensagem oferecendo os *codecs* disponíveis para cada tipo de mídia (neste caso áudio e vídeo) é enviada.

Tipo	Valor
v=	Versão do protocolo
o=	Originador e identificador da sessão
s=	Nome da sessão
i=	Informações sobre a sessão
a=	Atributos da sessão
t=	Tempo pela qual a sessão ficará ativa
m=	Nome da mída

Tabela 3.3: Exemplo de algumas informações transmitidas pelo SDP

Em seguida uma resposta onde um *codec* para cada tipo de mídia é escolhido.

```
1  Oferta:
2  v=0
3  o=jose 2890844526 2890844526 IN IP4 host.uc.com
4  c=IN IP4 host.uc.com
5  m=audio 49170 RTP/AVP 0 8 97
6  a=rtpmap:0 PCMU/8000
7  a=rtpmap:8 PCMA/8000
8  a=rtpmap:97 iLBC/8000
9  m=video 51372 RTP/AVP 31 32
10 a=rtpmap:31 H261/90000
11 a=rtpmap:32 MPV/90000
12
13 Resposta:
14 v=0
15 o=paulo 2808844564 2808844564 IN IP4 host.sip.com
16 c=IN IP4 host.sip.com
17 m=audio 49174 RTP/AVP 0
18 a=rtpmap:0 PCMU/8000
19 m=video 49170 RTP/AVP 32
20 a=rtpmap:32 MPV/90000
```

Figura 3.8: Exemplo de negociação SDP

3.2 XMPP

O *Extensible Messaging and Presence Protocol* (XMPP) é um protocolo aberto, definido nas RFCs 3920(SAINT-ANDRE, 2004b) e 3921(SAINT-ANDRE, 2004a). Criado inicialmente por Jeremie Miller em 1998, o XMPP (que inicialmente era chamado de *Jabber*(XMPP Standards Foundation, 1998)) é um protocolo para comunicações em tempo real, atualmente utilizado por uma ampla gama de aplicações com diversos fins como mensagens instatânes, presença, bate-papo em grupo, chamadas de voz/vídeo, colaboração e várias outras.

Foi inicialmente desenvolvido para criar um sistema de mensagens instantâneas, sendo baseado na troca de mensagens formatadas em XML. Devido a natureza extensível do XML, o XMPP tornou-se atraente para programadores que precisavam de um protocolo confiável para a rápida troca de dados estruturados. Com isso, surgiram diversas extensões que possibilitam o uso do XMPP em diversos serviços como compartilhamento de *desktop*, alertas, notificações e - entre outras - negociação de sessões multimídias.

XML é um padrão aberto para criação de documentos de forma estruturada, organizada e hierárquica.(BRAY; PAOLI et al., 2008) Herda algumas características do HTML, como o uso de *tags*. Porém o XML se concentra em dados ao contrário do HTML que tem como objetivo a apresentação destes dados.

O XMPP possui uma arquitetura cliente-servidor descentralizada, federada, semelhante a utilizada na rede de email. A figura 3.9 ilustra o funcionamento desta arquitetura, onde cada aplicação/cliente/entidade(*entitie*) está conectado a um servidor e este conectado a um ou mais outros servidores. Desta forma a rede torna-se mais robusta pois não há um único ponto de falha.

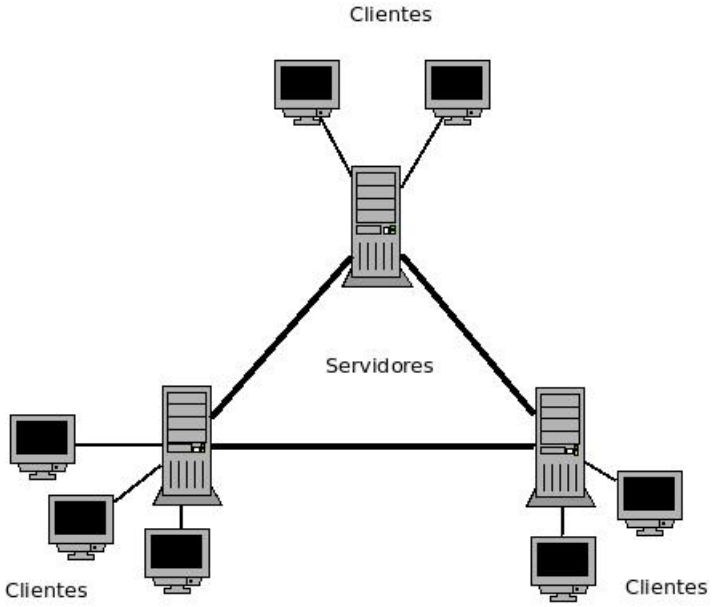

Figura 3.9: Arquitetura do XMPP

Para haver comunicações entre as entidades elas precisam ser identificadas na rede. Para facilitar a utilização pelas pessoas, é utilizado um endereçamento semelhante ao utilizado em um email: `usuário@dominio.com`. Este endereço é conhecido como *Jabber ID* (JID) e é composto por um usuário, um domínio e um identificador de recurso (geralmente o nome do apliativo em uso). Ao conectar com um cliente ao servidor XMPP, um identificador de re-

curso é escolhido. Desta forma é possível que um usuário conecte-se ao servidor simultaneamente a partir de mais de um cliente utilizando o mesmo usuário, o que permite inclusive o uso de agentes *(bots)* compartilhando o mesmo JID. Este identificador serve para rotear o tráfego para o cliente correto e é transparente para o usuário final. Desta forma o JID fica: `usuário@dominio.com/idderecurso`. Caso uma mensagem seja encaminhada sem a informação do ID de recurso, então o servidor encaminhará a mensagem para o recurso com maior prioridade. Esse valor de prioridade será explicado mais adiante, quando for explicado as mensagens de presença.

3.2.1 Estrutura

Conforme comentado, o XMPP tem um funcionamento semelhante ao do email. Ao enviar uma mensagem XMPP para um contato em outro domínio, o cliente originador envia inicialmente esta mensagem para seu servidor. Ao receber esta mensagem este servidor conectará diretamente - sem intermediários - ao servidor onde encontra-se o cliente de destino para dar início as trocas de mensagens. A diferença ocorre no servidor de destino: enquanto que no email a mensagem fica armazenada para leitura, no XMPP ela é armazenada no servidor e, em seguida, encaminhada ao usuário quando esse estiver conectado, sem a necessidade de outro serviço/protocolo. Por esse motivo, diz-se que XMPP é tanto síncrono como assíncrono, pois se o usuário não estiver conectado, o servidor poderá armazenar a mensagem até a disponibilidade do usuário, conforme a XEP-0013(SAINT-ANDRE; KAES, 2005).

Basicamente o XMPP é um tecnologia de *streaming* de mensagens XML. Ao iniciar uma sessão com um servidor XMPP, o cliente abre uma conexão permanente TCP - por onde trafegarão as mensagens XML do cliente para o servidor - e o servidor abre outra conexão permanente - para o *stream* originado no servidor com destino ao cliente. Ao serem abertos esses canais de ida e volta, três tipos de mensagens - chamadas de XML *stanzas* - são trocados entre as duas pontas: `<message/>`, `<presence/>`e `<iq/>`. Essas três mensagens são as unidades básicas do XMPP e, após o estabelecimento de conexão entre o cliente e o servidor, podem ser trocadas inúmeras dessas mensagens.

Exemplo de processo de conexão entre cliente e servidor:

1. Determina o o IP e porta ao qual irá se conectar
2. Inicia uma conexão TCP
3. Inicia um stream XML sobre a conexão TCP

4. Opcionalmente negocia os protocolos de segurança para a comunicação (*Transport Layer Security* (TLS) e *Simple Authentication and Security Layer* (SASL)).

5. Troca incontáveis números de XML *stanzas* com outras entidades na rede.

6. Fecha o *stream* XML

7. Fecha a conexão TCP

Exemplo de processo de conexão entre servidores:

1. Determina o o IP e porta ao qual irá se conectar

2. Inicia uma conexão TCP

3. Inicia um stream XML sobre a conexão TCP

4. Opcionalmente negocia os protocolos de segurança para a comunicação (TLS e SASL).

5. Troca incontáveis números de XML *stanzas* com outros servidores, ou reencaminha mensagens dos clientes para o destino correto.

6. Fecha o *stream* XML

7. Fecha a conexão TCP

Um ponto importante a se ressaltar é que após aberta esta conexão entre o cliente e o servidor a mesma é mantida neste estado mesmo se momentâneamente não há informação a ser trocada entre as duas pontas. Dessa forma, uma mensagem XML é aberta no início da comunicação e a mesma só é fechada no momento da desconexão do cliente com o servidor. Ou seja, ao longo do tempo é como se um grande arquivo XML tivesse sido trocado entre as partes.

Esta abordagem do XMPP de abrir uma conexão permanente TCP e através dele trocar uma número ilimitado - e de forma síncrona ou assíncrona - de mensagens difere completamente da abordagem utilizada dos modos utilizados nas trocas de *email* e *web* - onde é aberto uma conexão, trocado informações e a conexão é encerrada. A forma utilizada pelo XMPP permite uma comunicação em tempo real pois há sempre um canal aberto para a troca de informações, independente da origem. Para tal, é preciso manter a sessão ativa, e por consequência a conexão TCP, mesmo quando não há dados a transmitir, usando para tal pacotes do tipo *ping* (que será melhor explicado mais adiante), usados basicamente para gerar algum tráfego periodicamente.

Como falado, existem basicamente três tipos de mensagens XML no XMPP - conhecidas como *stanzas*: <message/>, <presence/>e <iq/>. Um *stanza* é uma unidade básica no XMPP, assim como um pacote é a unidade básica de diversos outros protocolos. Cada um dessas *stanzas* possui características (atributos) específicas que os identificam e que serão discutidas a seguir.

3.2.2 PUBLISH e SUBSCRIBE/UNSUBSCRIBE

O *PUBLISH* serve para publicar uma atualização, semelhante ao método *NOTIFY* no SIP, e que será depois enviada difundida (geralmente usando *unicast*) aos assinantes.

```
<iq type='set'
    from='jpaulo@uc.com/pda'
    to='pubsub.uc.com'
    id='pub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='princely\_musings'>
      <item>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Ocupado</title>
          <summary>
Estou ocupado
          </summary>
          <link rel='alternate' type='text/html'
                href='http://uc.om/2010/02/13/atom03'/>
          <id>tag:uc.com,2010:entry-32397</id>
          <published>2010-02-13T18:30:02Z</published>
          <updated>2010-02-13T18:30:02Z</updated>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

Figura 3.10: Exemplo de mensagem *PUBLISH* do XMPP

Já o *SUBSCRIBE* serve para assinar a atualização de um usuário ou serviço, como por exemplo um blog de notícias. Após a assinatura ser efetivada atualizações deste usário, ou serviço, serão recebidas até a solicitação de cancelamente da assinatura através do *UNSUBSCRIBE*.

3.2.3 MESSAGE

A mensagem <message/> é uma forma básica de transmitir informações rapidamente, visto que normalmente ela não possui uma resposta de confirmação de recebimento, haja vista que XMPP se baseia exclusivamente sobre TCP e, portanto, há uma forte confiança no protocolo

da camada inferior na entrega do pacote. Um de seus principais atributos é o `type`, que identifica o tipo de mensagem trocada. Na tabela 3.4 são apresentados os valores - e seus significados - que esse atributo pode assumir, conforme a RFC3921(SAINT-ANDRE, 2004a).

Valor	Significado
`normal`	Mensagens deste tipo são similares a mensagens de email, podendo ou não existir respostas
`chat`	São mensagens trocadas em uma sessão em tempo real entre duas entidades
`groupchat`	Semelhante ao chat, porém envolvendo mais de duas entidades
`headline`	Usado em mensagens de alertas e notificações
`error`	aso seja detectado algum erro em uma mensagem enviada anteriormente, a entidade que identificar esse problema envia essa mensagem de erro

Tabela 3.4: *Valores que o atributo type do stanza <`message/`> pode assumir*

Outros atributos existentes no *stanza* <`message/`> são os endereços JID de origem (`from`) e destino (`to`). É importante destacar que o endereço `from` não é preenchido pelo cliente, e sim pelo servidor de origem para evitar fraudes. Outro ponto importante é que não há uma ordem especifica desses atributos nas mensagens.

Outros atributos podem ser inseridos, como <`body/`> onde estará o conteúdo da mensagem, <`subject/`> que basicamente especifica um título para a mensagem, <`id/`> que identifica a sessão e <`thread/`> que identifica uma sessão de conversação.

3.2.4 PRESENCE

Utilizado para informar a disponibilidade de um cliente. A mensagem pode ter significados diferentes, de acordo com o parâmetro *type* que pode assumir os valores demonstrados na tabela 3.5. Cabe destacar os termos `subscribe` e variantes, demonstrando que a lista de contatos é controlada por assinaturas e publicações via método *push*.

Valor	Significado
`unavailable`	Avisa que a entidade não está mais disponível para comunicação
`subscribe`	Solicita inscrição
`subscribed`	Informa que a solicitação foi aceita
`unsubscribe`	Solicita desinscrição
`unsubscribed`	Solicitação aceita
`probe`	solicita o status atual
`error`	informa algum erro

Tabela 3.5: *Valores que o atributo `type` do stanza <`presence/`> pode assumir*

Uma mensagem <presence> sem informar o *type* significa que o usuário que enviou a mensagem está *online* (seria o contrário do type=unavailable). Alé do parametro *type*, existem também os parametros *show, status* e *priority*. Na tabela 3.6 são demonstrados os valores que o parâmetro *show* pode receber e seus significados. O parâmetro *status* é opcional, onde o usuário pode especificar o que está fazendo. Por exemplo, o usuário pode definir que não quer ser encomodado (show=dnd) e coloca como status: "Em reunião". Atua, na prática, como um complemento ao atributo *type*.

Valor	Significado
away	Informa que a entidade, ou recurso, está temporariamente ausente.
chat	Informa que a entidade, ou recurso, está interessado em iniciar um chat.
dnd	Informa que a entidade, ou recurso, não quer ser encomodada.
xa	Informa que a entidade, ou recurso, está ausente por um período prolongado

Tabela 3.6: *Valores que o atributo type do stanza <presence/> pode assumir*

Já o campo *priority* informa a prioridade da entidade que enviou a mensagem. Pode assumir um valor inteiro entre -128 e 127. Como foi falado anteriormente, um servidor ao receber uma mensagem destinado a um de seus clientes, e esta mensagem não possuir o ID de recurso que deverá receber esta mensagem ele encaminhará ao recurso que possuir maior prioridade. Recursos com prioridade negativas não recebem mensagens, sendo utilizado por aplicações que divulgam informações (*bots*) e não precisam receber nada. Um exemplo seria uma aplicação monitorando um servidor e que quando um disco estivesse com a ocupação muito alta seria enviado mensagem aos responsáveis informando deste alarme.

3.2.5 O *stanza* <iq>

O *stanza* <iq> (*Info/Query*) fornece uma interação de Pergunta-Resposta[2]. Ao contrário do *message* apenas uma informação pode ser transmitida no *stanza* <iq>, que é a requisição que está sendo feita. A entidade que gera esta mensagem sempre espera por uma resposta, sendo que estas mensagens são identificadas por um atributo chamado *id* que é gerado por quem envia a requisição e incluído na resposta desta requisição. Já o atributo *type* pode assumir quatro valores, os quais são apresentados na tabela 3.7.

Um exemplo de uso desta mensagem é em uma situação de mensagem instantânea quando um usuário se conecta. Uma de suas primeiras ações é solicitar a sua lista de contatos, utilizando um *stanza* <iq> com type=get. No exemplo da figura 3.11 é possível ver o envio desta

[2]É o único dos tipos básicos de *stanza* que espera uma resposta, o que lembra o funcionamento Requisição-Resposta do SIP

Atributo	Significado
get	Solicitando uma informação
set	Provendo uma informação
result	Resposta a um get
erro	Falha no processamento da mensagem get ou set

Tabela 3.7: *Atributos possíveis e seus significados no stanza <iq>*

mensagem, e na figura 3.12 a resposta do servidor. Na figura 3.13 é demonstrado um exemplo de uso do `type=set`, com o usuário solicitando a adição de um novo contato. Como comentado, os *stanzas* <iq> exigem que toda requisição tenha uma resposta, a qual também é demonstrada na figura 3.14.

```
Cliente -> Servidor:
<iq from="jpaulo@uc.com/pda"
      id="rr82a1z7"
      to="jpaulo@uc.com"
      type="get">
   <query xmlns="jabber:iq:roster"/>
</iq>
```

Figura 3.11: Exemplo de uso do stanza <iq>com o parâmetro `type=get`

```
<iq from="jpaulo@uc.com"
     id="rr82a1z7"
     to="jpaulo@uc.com/pda"
     type="result">
   <query xmlns="jabber:iq:roster">
     <item jid="pedro@uc.com"/>
     <item jid="joao@uc.com"/>
     <item jid="maria@uc.com"/>
  </query>
</iq>
```

Figura 3.12: Exemplo de uso do stanza <iq>para resposta da mensagem anterior

```
<iq from="jpaulo@uc.com/pda"
      id="ru761vd7"
      to="jpaulo@uc.com"
      type="set">
   <query xmlns="jabber:id:roster">
      <item jid="tiago@uc.com"/>
   </query>
</iq>
```

Figura 3.13: Exemplo de uso do stanza <iq>com o parâmetro `type=set`

```
<iq from="jpaulo@uc.com"
      id="ru761vd7"
      to="jpaulo@uc.com/pda"
      type="result"/>
```

Figura 3.14: Exemplo de uso do stanza <iq>para resposta da mensagem anterior

3.2.6 Jingle

Jingle é uma extensão do XMPP(SAINT-ANDRE; LUDWIG et al., 2009a) que serve para a negociação, manutenção e encerramento de sessões multímidias, as quais suportam diversos tipos de mídias como áudio, vídeo ou transferência de arquivos. Usa o procedimento básico de uma negociação de mídia, e outros parâmetros, que será utilizado na sessão. As mensagens trocadas para esta negociação são basicamente *stanzas* <iq> com campos semelhantes aos existentes no SIP/SDP de modo a facilitar a interoperabilidade.

A extensão XEP167(SAINT-ANDRE; LUDWIG et al., 2009b) define o funcionamento para negociações de mídias RTP e consiste basicamente no envio de um convite com sugestões de paramêtros da mídia e uma resposa de confirmação, ou não, deste convite. Caso esta resposta seja afirmativa, ela conterá as sugestões aceitas de mídias, dando início a sessão.

3.3 Real-time Transport Protocol

Protocolo para transporte em tempo real definido na RFC3550(SCHULZRINNE et al., 2003), para transmissão de dados referentes a vídeo e/ou áudio. Normalmente utiliza o protocolo UDP na camada de transporte, porém pode utilizar outros protocolos. O RTP não provê nenhum mecanismo para garantir a entrega em tempo real ou qualquer outra garantia de qualidade de serviço. Também não previne quanto a entrega dos pacotes fora de ordem, pois assume que as camadas inferiores são confiáveis e farão esse controle.

Funciona em conjunto com o protocolo *RTP Control Protocol* (RTCP), que funciona de forma *out-of-band*, ou seja, fora do canal de comunicação do RTP e auxilia no controle da qualidade da transmissão gerando relatórios estatísticos da mesma. Com base nas informações providas pelo RTCP, as partes envolvidas na comunicação RTP podem negociar, por exemplo, a troca do *codec* que está sendo utilizado. Esta troca pode ser para um com maior qualidade, caso seja percebido que o *link* suporta isso, ou para um com menor qualidade caso o RTCP demonstre uma quantidade de erros muito excessiva.

4 *Comparação entre os protocolos SIP/SIMPLE e XMPP*

4.1 Funcionalidades

Existem algumas diferenças básicas entre os protocolos SIP/SIMPLE e o XMPP. A seguir seguem algumas comparações dos protocolos SIP e XMPP com relação a algumas de suas carecterísticas. (SINGH, 2009)

1. Proposta
   (a) **SIP**: Provê uma forma de negociação e estabelecimento de sessões multimídias.
   (b) **XMPP**: Provê um canal para troca de dados estruturados (XML) entre usuários - com a ajuda de servidores - utilizado por exemplo para mensagens instantâneas e presença
2. Protocolo
   (a) **SIP**: Protocolo baseado em texto do tipo pergunta-resposta semelhante ao HTTP. Seus atributos principais são negociados através de parâmetros nos cabeçalhos das mensagens, e dados adicionais são transmitidos no corpo das mensagens
   (b) **XMPP**: Protocolo baseado em troca de mensagens codificadas em XML em uma estrutura cliente-servidor. Essas mensagens são trocadas por um canal permanente que é aberto entre o cliente e o servidor e entre servidores.
3. Transporte
   (a) **SIP**: Pode funcionar sobre UDP, TCP com ou sem TLS[1] e SCTP[2].
   (b) **XMPP**: TCP com ou sem TLS.

[1] Protocolo que roda sobre o TCP com o objetivo de criptografar estes pacotes de modo a torná-los mais seguros.
[2] Protocolo de camada de transporte (assim como o TCP e o UDP)

4. Conexão

   (a) **SIP**: Pode ser iniciada tanto pelo cliente como pelo servidor, o que dificulta seu uso em ambientes com *Network Address Translation* (NAT) e/ou *firewalls*. Extensões são utilizadas para tentar amenizar este problema, criando conexões reversas quando o servidor quer enviar uma mensagem ao cliente.

   (b) **XMPP**: O cliente que inicia a conexão com o servidor, abrindo um canal permanente, o que funciona bem mesmo em ambientes com NAT e/ou *firewalls*. Existem extensões, como *Bidirectional-streams Over Synchronous HTTP* (BOSH), que permitem enviar as mensagens sobre HTTP, permitindo seu funcionamento com *firewall* muito restritivos.

4.1.1 Problemas com sessões/conexões abertas

É uma "invenção" manter o estado em UDP, já que ele não tem *flags*. O que acontece é que quando origina-se uma comunicação a partir do ambiente externo, a cada 120 segundos, em média, deve haver um pacote em algum sentido para manter a "sessão antiga".

Ambos os protocolos enfrentam problemas com clientes localizados atrás de *firewalls* ou em redes onde NAT é utilizado. Estas dificuldades podem ser contornadas - com algumas exceções - utilizando extensões que ambos os protocolos possuem e servidores específicos(*Session Traversal Utilities for NAT* (STUN)(ROSENBERG, 2008), ou então o *Interactive Connectivity Establishment* (ICE)(CAMARILLO; ROSENBERG, 2005)) para este fim.

XMPP: XMPP Ping

Como já foi visto, uma conexão XMPP entre um cliente e seu servidor é realizado através de um canal permanente entre as partes utilizando o protocolo TCP como transporte. Porém essa conexão pode cair sem que as entidades XMPP sejam noticiados destes problema. Uma maneira de fazer essa verificação de conectividade entre um cliente e um servidor, ou entre quaisquer duas entidades XMPP, é realizando um *ping*.

No caso do XMPP, a extensão XEP0199(SAINT-ANDRE, 2009) define como esse teste de conectividade deve acontecer. Seu funcionamento consiste basicamente no envio, a partir da entidade que quer verificar a conectividade, de um *stanza* <iq>, do tipo *get* (type=get), com um elemento <ping/>, conforme demonstrado na figura 4.1. Ao receber esta mensagem a entidade que está sendo testada deve responder com um *stanza* <iq>, do tipo *result* (*type=result*)

conforme demonstrado na figura 4.2. Caso a entidade que está sendo testada não tenha suporte a esta extensão uma mensagem *stanza* <iq> do tipo *error* (type=error) deve ser respondida.

```
<iq from='jose@floripa.com/home'
    to='paulo@saojose.com/work'
    type='get'
    id='e2e1'>
  <ping xmlns='urn:xmpp:ping'/>
</iq>
```

Figura 4.1: Exemplo de envio do XMPP Ping

```
<iq from='paulo@saojose.com/work'
    to='jose@floripa.com/home'
    id='e2e1'
    type='result'/>
```

Figura 4.2: Exemplo resposta ao XMPP Ping

SIP: OPTIONS periódicos (keep-alive)

O SIP não possui, formalizado, uma forma de verificação de conectividade entre entidades. Uma forma de realizar este teste, comumente utilizado, é através do método *OPTIONS*. Esse método, definido na RFC3261, foi criado para que uma entidade seja consultada quanto as suas capacidades. Ao receber essa mensagem uma resposta é gerada informando ao solicitante informações de quais métodos, extensões, codecs e outras informações esta entidade suporta.

Dessa forma, esse método pode ser utilizado para verificar a conectividade entre as entidades, basicamente realizando um envio e aguardando sua resposta. Caso essa resposta não chegue um problema de conectividade foi encontrado. Outra forma é utilizado por alguns clientes SIP é o envio periódico de um *REGISTER*, com o mesmo objetivo.

4.1.2 A segurança da autenticação e transmissão

Ambos os protocolos podem utilizar o protocolo TLS(DIERKS; ALLEN, 1999) para realizar a autenticação e criptografia das mensagens de sinalização trocadas entre as entidades. Lembrando que o uso do SIP, ou XMPP, com TLS a segurança ocorre apenas na sinalização da sessão. Para que haja uma maior segurança no transporte das mídias, protocolos como o *Secure Real Time Protocol* (sRTP)(BAUGHER; MCGREW et al., 2004) devem ser usados.

4.1.3 Sessão

A forma como as sessões são tratadas pelo SIP e pelo XMPP são bem diferentes. No caso do XMPP, um cliente abre uma conexão com um servidor que permanecerá aberta por todo o período que o cliente ficar online. Por esta conexão podem ser negociadas diversas sessões como também nenhuma. Após o encerramento desta conexão entre o cliente e o servidor, caso todas as mensagens que foram trocadas sejam colocadas na sequência que ocorreram seu formato será de um único - e grande - arquivo XML.

Já no caso do SIP o cliente inicialmente envia uma mensagem ao seu servidor solicitando registro. Dessa forma o servidor fica sabendo que o cliente está *online* e como encontrá-lo - através do endereço IP dele, por exemplo. Caso receba alguma solicitação de sessão destinada a este cliente o servidor informa ao solicitante como encontrá-lo, iniciando a comunicação e negociação da sessão entre as duas pontas. Ou seja, diferentemente do XMPP não há um canal permanente entre os elementos.

4.1.4 Envio de arquivos

O XMPP possui algumas extensões que definem formas de transferência de arquivos entre duas entidades. Estas transferências podem ocorrer *in-band* [3] ou *out-of-band*[4]. A extensão XEP0066(SAINT-ANDRE, 2006) é um exemplo de protocolo para transferência de arquivos *out-of-band*, onde o usuário que quer enviar o arquivo o disponibiliza em um servidor *File Transfer Protocol* (FTP) ou HTTP e através de mensagens especificadas nesta extensão o endereço onde o arquivo está é informado a outra entidade. Já a extensão XEP0047(SAINT-ANDRE; KARNEGES, 2009) define uma forma de transferência de arquivo *in-band*, onde o arquivo binário a ser transmitido é codificado em *base64/MIME*(FREED; BORENSTEIN, 1996) e o resultado desta codificação é enviado nos *stanzas* XML trocado entre as entidades.

XMPP é ineficiente para a transferência de arquivos dentro do *stream*(*in-band*) de XMPP, sendo recomendado criar uma conexão externa (*out-of-band*) ao *stream* para o envio de arquivos sendo essa conexão gerênciada utilizando o XMPP. Esse seria o mesmo conceito da transmissão de mídias porem sendo utilizando para a transferência de arquivos, podendo ser utilizada a extensão *Jingle*(SAINT-ANDRE; LUDWIG et al., 2009a) para isso. As mesmas dificuldades encontradas em uma transmissão de mídia ocorrerão neste caso, como os problemas com *NATs* e *firewall* por exemplo. Neste caso a extensão XEP0234(SAINT-ANDRE, 2010)

[3] transferências que ocorrem no mesmo *stream*

[4] semelhante a transmissão de mídias, transferencias de arquivos *out of band* os arquivos binários são enviados em uma conexão extra que é criada especificamente para este fim e gerência pelo *stream* principal

define a utilização do *Jingle* para a transferência de arquivos.

No caso do SIP a RFC sugere o uso do SDP para negociar uma sessão MSRP por onde o arquivo será enviado. Desta forma o arquivo será codificado em *base64/MIME* e transferido dentro das mensagens MSRP trocadas entre as entidades.

4.1.5 Confirmação

Como já foi visto o XMPP funciona sobre TCP (com ou sem TLS). Baseado nisso ele tem plena confiança no protocolo de transporte, visto que o TCP é um protocolo que garante a entrega dos pacotes, diferentemente do UDP por exemplo. Já o SIP pode rodar tanto sobre o TCP (ou TLS) como sobre UDP. E uma característica é que ele não confia no protocolo de transporte[5], independente de qual ele esteja utilizando. Então para cada mensagem enviada no SIP uma resposta é aguardada (como a `200 OK`).

4.1.6 Roteamento

O roteamento dos *stanzas* são realizados pelos servidores XMPP. Ou seja, caso um cliente A - que está conectado a um servidor A - queira se comunicar com um cliente B - que está conectado em um servidor B - então todas as mensagens trocadas entre esses clientes passarão pelos seus respectivos servidores conforme demonstrado na figura 4.3. Já no caso do SIP os servidores servem para auxiliar na procura pela localização dos clientes. Após o cliente ser encontrado a comunicação ocorre diretamente entre os clientes.

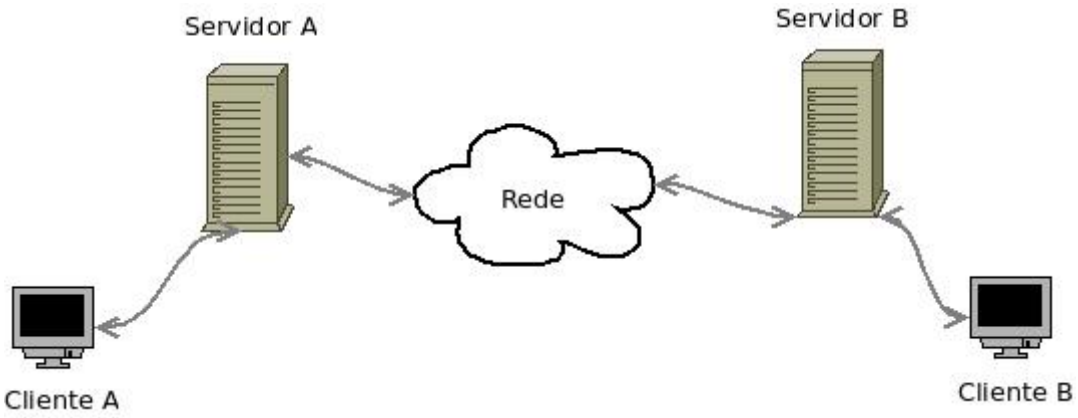

Figura 4.3: No XMPP os servidores são responsáveis pelo roteamento das mensagens trocadas entre os clientes

[5]Podendo utilizar o SCTP também, protocolo esse que roda na mesma camada do TCP e do UDP e tenta juntar as qualidades destes dois protocolos, conforme RFC 3286(ONG; YOAKUM, 2002)

4.1.7 Envio de mídia

O XMPP foi criado inicialmente com o intuito de prover MI e anúncios de presença. Sua utilização para a negociação de mídias (como chamadas de voz ou vídeo conferências por exemplo) foi acrescentada com a criação de extensões ao protocolo original - como a extensão *Jingle*. Já o SIP nasceu para a negociação de mídias. Ambos os protocolos servem para iniciar, negociar, modificar e encerrar sessões multimídias porém não são responsáveis pelo transporte da mídia propriamente dita, sendo esta comumente uma função do protocolo RTP (que será melhor explicado no item 3.3).

Na tabela 4.1 temos um resumo destes pontos que foram observados.

	SIP	**XMPP**
Verificação de conectividade	Realizado de forma alternativa, utilizando um método que não foi criado para este fim (OPTIONS)	Possui uma extensão que define uma forma de teste de conectividade
Segurança	Possui suporte a TLS (apenas sobre TCP)	Possui suporte a TLS
Sessão	Para cada sessão nova uma nova conexão é gerada	Permanente entre cliente e servidor, por onde são trocadas as mensagens XML.
Envio de arquivos	RFC5547(GARCIA-MARTIN; LORETO et al., 2009) - Sinaliza através de SDP, envia através de uma sessão MSRP. Possibilita apenas o envio de arquivos binários base64/MIME como fotos e clipes de vídeo.	Possui várias extensões, como a XEP0234, XEP0047 ou a XEP0066.
Confirmação	Não confia na camada inferior, possuindo mensagens de confirmação (como o 200 OK).	Mensagens (*message*): confia no TCP; consultas (*query*): espera a resposta (*result/error*)
Roteamento	Cliente recebe informações como Via e Route-to, podendo escolher o melhor caminho	Tudo feito no servidor, de forma federada (normalmente clienteA-servidorA-servidorB-clienteB)
Envio de mídias	Nativo do protocolo em conjunto com o RTP	Através da extensão *Jingle*(SAINT-ANDRE; LUDWIG et al., 2009b) em conjunto com o RTP

Tabela 4.1: *Resumo das comparações realizadas entre os protocolos SIP e XMPP*

4.2 Análise de possibilidades e estudos envolvendo interoperabilidade entre os protocolos

4.2.1 Presença

Tanto o SIP/SIMPLE quanto o XMPP permitem que clientes assinem e mantenham-se informados a respeito da presença de outros clientes. Seu funcionamento básico consiste em um cliente realizando uma assinatura, que pode ser aceita ou não. Sendo a assinatura realizada com sucesso, a cada nova atualização da entidade assinada, os assinantes serão notificados instantaneamente. Existe também a possibilidade de cancelar a assinatura, parando com o recebimento das notificações.

SIP

O SIP é hoje o protocolo mais utilizado para VoIP, adotado inclusive para telefonia 3G pelo grupo 3GPP. Pelo seu uso em franca expansão, foram criadas algumas extensões as quais agregam mais informações nos pacotes SIP. Uma dessas extensões refere-se a transmissão de informação de presença, e a forma como essa informação é inserida em um pacote SIP é definida através da RFC 3863. Esta RFC determina que as informações de presença inseridas no pacote SIP serão formatadas em XML, e introduz os campos que serão utilizados. A maneira como estes elementos são representados, e seus significados, são demonstrados a seguir:

<presence> Elemento raiz. É seguido de N elementos <tuple>que é seguido por N elementos <note>que é seguido por N elementos opcionais. Deve possuir o endereço padronizado e único da *PRESENTITY*.

<tuple> Possui uma tupla de *PRESENCE*, que consiste obrigatoriamente de um elemento <status>seguido por N elementos opcionais. Provê uma maneira de segmentar a informação de presença.

<status> Deve possuir um elemento <basic> seguido de N elementos opcionais. Deve obrigatoriamente possuir um elemento filho que possuirá a informação de presença (o *status*) deste elemento.

<basic> Pode assumir dois valores: *OPEN* ou *CLOSED*, que possuem significados de acordo com seu uso. Para MI *OPEN* pode significar Disponível para conversa e *CLOSED* indisponível

<contact> Contém a URL do endereço de contato

<note> Utilizado apenas para comentários que serão lidos por humanos como observações extras.

<timestamp> Contém a data e hora da alteração da informação de presença

Existem, basicamente, duas formas de divulgação, ou consulta, da informação de presença: *polling* e *push*. Na primeira, um cliente faz periodicamente uma consulta ao servidor sobre uma dada informação. Simples e fácil, porém com dificuldades em escalabilidade: quanto mais clientes realizando essas perguntas periodicamente maior o trabalho do servidor para respondê-las. Uma outra abordagem é através do segundo método, o *push*: nesse, o cliente avisa ao servidor que quer ser informado de determinadas atualizações e, assim que o servidor identificar essas atualizações, uma mensagem é disparada para os clientes que a solicitaram anteriormente.

O *Publish/Subscribe* (PubSub) é um modelo de notificação usando *push*. Tanto o SIP como o XMPP possuem esta implementação dando a possibilidade de clientes assinarem um determinado serviço (presença de um usuário, canal de notícias, atualizações de *blog*, etc) e serem informados assim que uma atualização ocorre.

No caso do XMPP, a extensão que define o seu funcionamento genérico é a XEP060[6]. Basicamente, consiste em um nó interessado em receber avisos de atualizações de outro nó, realizando a assinatura deste serviço através de um servidor de *PubSub*. Este servidor, ao receber uma atualização, irá divulgá-la para todos os clientes, também denominados nós, que assinaram estas atualizações.

Já para o SIP, há mais de uma extensão que define a utilização do *Publish/Subscribe*. Abrangendo os conceito de *Subscribe* e *Notify*, existe a RFC 3265(ROACH, 2002). Nela são apresentados os parâmetros que envolvem o processo de assinatura e de cancelamento desta assinatura. Também há a definição das mensagens de notificação, as quais servem para informar aos assinantes sobre atualizações do serviço assinado. Já na RFC 3903(NIEMI, 2004) há a definição de como um cliente SIP publica uma atualização.

Estas mensagens que foram apresentadas englobam o funcionamento básico da divulgação da informação de presença pelos protocolos SIP e XMPP - e suas devidas extensões. Detalhes destes protocolos estão devidamento especificados nas RFCs informadas. Como foi visto, ambos possuem ações semelhantes - onde podemos concluir que há a possibilidade de intero-

[6] A presença de pessoas em uma lista de contatos para bate-papo, um caso mais específico de PubSub, está definido na RFC 3920(SAINT-ANDRE, 2004b) e que, embora faça uso de outras mensagens, tem seu funcionamento equivalente ao genérico.

perabilidade entre eles - porém suas formatações são completamente diferentes, dificultando a utilização de ambos em conjunto necessitando da existência de um conversor entre estes dois protocolos.

Conforme a RFC 3265(ROACH, 2002), para um nó solicitar a um servidor *PubSub* uma assinatura de outro nó, ou de um serviço, é enviado para o servidor uma mensagem do tipo *SUBSCRIBE*, conforme demonstrado na figura 4.4, contendo a `URI` da entidade que se deseja assinar, bem como a informação de em quanto tempo essa assinatura expira. Ao processar corretamente o servidor responde com uma mensagem SIP do tipo 200, informando o sucesso na operação. Para o cancelamento de uma assinatura, esta mesma mensagem *SUBSCRIBE* é enviada com o valor do `expire` setado para 0 (zero).

```
1  SUBSCRIBE sip:presentity@uc.comSIP/2.0
2  Via: SIP/2.0/UDP host.uc.com;branch=z9hG4bKnashds7
3  To: <sip:presentity@uc.com>
4  From: <sip:watcher@uc.com>;tag=12341234
5  Call-ID:12345678@host.uc.com
6  CSeq: 1 SUBSCRIBE
7  Max-Forwards: 70
8  Expires: 3600
9  Event: presence
10 Contact: sip:user@host.uc.com
11 Content-Length: 0
```

Figura 4.4: Exemplo de mensagem SIP solicitando assinatura

Após o envio da mensagem de confirmação de assinatura, o servidor também envia uma mensagem de notificação com o objetivo de divulgar a última atualização do nó assinado. Esta mesma mensagem é enviada para os assinantes a cada atualização do nó assinado. Um exemplo desta mensagem pode ser verificado na figura 4.5. O formato da atualização é definido na RFC 3863(SUGANO; PETERSON et al., 2004).

Para publicar uma nova atualização, o nó deverá enviar para o servidor de *PubSub* uma mensagem do tipo *PUBLISH*, exemplificada na figura 4.6, contendo a nova atualização. Ao processar com sucesso o servidor responderá ao nó com uma mensagem SIP `200 OK`, e caso haja algum erro o nó será informado do que ocorreu. Após o processamento pelo servidor este enviará para os assinantes uma mensagem do tipo *NOTIFY*, notificando da atualização que ocorreu.

```
NOTIFY sip:user@host.uc.com SIP/2.0
Via: SIP/2.0/UDP pa.uc.com;branch=z9hG4bK4cd42a
To: <sip:watcher@uc.com>;tag=12341234
From: <sip:presentity@uc.com>;tag=abcd1234
Call-ID:12345678@host.uc.com
CSeq: 2 NOTIFY
Max-Forwards: 70
Event: presence
Subscription-State: active; expires=3400
Contact: sip:pa.uc.com
Content-Type: application/pidf+xml
Content-Length: ...

[PIDF Document]
```

Figura 4.5: Exemplo de mensagem SIP de notificação que é enviado aos assinantes com as informações de presença do nó assinado

```
PUBLISH sip:presentity@uc.com SIP/2.0
Via: SIP/2.0/UDP pua.uc.com;branch=z9hG4bK652hsge
To: <sip:presentity@uc.com>
From: <sip:presentity@uc.com>;tag=1234wxyz
Call-ID:81818181@pua.uc.com
CSeq: 1 PUBLISH
Max-Forwards: 70
Expires: 3600
 Event: presence
Content-Type: application/pidf+xml
Content-Length: ...

[PIDF Document]
```

Figura 4.6: Exemplo de mensagem SIP de publicação de uma nova atualização

XMPP

Na figura 4.7 é demonstrada a mensagem que é enviada ao servidor de *PubSub* solicitando a assinatura de um determinado serviço/nó. Conforme pode-se observar, no campo `subscribe` são informados os parâmetros `node`, que é o nó ao qual se quer assinar, e o JID do nó que se está assinando. Desta forma, o nó informado no parâmetro JID receberá uma notificação a cada atualização realizado pelo nó informado no campo `node`. Ao receber essa requisição o servidor deverá responder informação de a assinatura foi realizada com sucesso ou não - e informando o erro correspondente. Também é possível a aprovação da assinatura ou o envio de mais informações para realizar assinatura, como por exemplo autenticação. Todas essas informações são enviados de volta ao nó que deseja assinar, de acordo com o padrão definido na XEP0060(SAINT-ANDRE; MILLARD; MEIJER, 2010).

```
<iq type='set'
    from='francisco@uc.com/notebook'
    to='pubsub.uc.com'
    id='sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe
        node='princely\_musings'
        jid='francisco@uc.com'/>
  </pubsub>
</iq>
```

Figura 4.7: Exemplo de mensagem XMPP de solicitação de assinatura

Após a assinatura ser realizada com sucesso o servidor de *PubSub* tem a opção de enviar uma mensagem informando a última atualização do nó assinado. A partir de então, toda atualização do nó assinado será enviada para todos os assinantes deste nó. Um exemplo de mensagem de atualização pode ser vista na figura 4.8. Da mesma forma que ocorre com a mensagem para assinatura, o servidor irá responder ao nó se a atualização enviada foi processada com sucesso ou não. Um item importante da mensagem de atualização, além da própria atualização, é o parametro `id`, que identifica cada transação entre os elementos do serviço. Desta forma é possível editar notificações que já tenham sido enviadas utilizando essa identificação.

O servidor de *PubSub* ao receber uma atualização e processá-la com sucesso, enviará uma notificação a todos os assinantes do nó atualizado. Essa notificação poderá ser enviada informando a atualização em si, conforme demonstrado na figura 4.9, ou pode ser apenas um notificação de que houve uma atualização e, se o assinante desejar, deverá enviar uma mensagem solicitando a atualização. No segundo caso há uma diminuição no tamanho dos pacotes enviados, ocasionando um melhor desempenho por parte do servidor.

```
<iq type='set'
    from='jpaulo@uc.com/pda'
    to='pubsub.uc.com'
    id='pub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='princely\_musings'>
      <item>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Ocupado</title>
          <summary>
Estou ocupado
          </summary>
          <link rel='alternate' type='text/html'
                href='http://uc.om/2010/02/13/atom03'/>
          <id>tag:uc.com,2010:entry-32397</id>
          <published>2010-02-13T18:30:02Z</published>
          <updated>2010-02-13T18:30:02Z</updated>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

Figura 4.8: Exemplo de mensagem XMPP de atualização de presença

Para o caso de o assinante não desejar mais receber atualizações de um determinado nó, ele deverá enviar um pedido de cancelamento de assinatura ao servidor de PubSub. Este pedido possuirá basicamente o nome do nó que se deseja cancelar a assinatura. Igual ocorre com as demais mensagens, ao receber este pedido e processá-lo o servidor informará ao assinante se o cancelamento foi realizado com sucesso ou não, informando o erro caso ocorra. Na figura 4.10 está representado um exemplo de mensagem solicitando o cancelamento de uma assinatura.

4.2.2 Conversão SIP x XMPP

Como foi visto tanto o SIP quanto o XMPP possibilitam que entidades troquem atualizações de presença entre si. Essa informação pode ser tanto um simples *online/offline* como informações mais completas como coordenadas GPS.

A RFC 2779(DAY et al., 2000) define alguns requisitos básicos que protocolos que implementam presença, e mensagem instantâneas, devem seguir. Considerando que tanto o SIP como o XMPP tenham como base esses requisitos, para que haja interoperabilidade entre eles é necessário um mapeamento entre estes protocolos. Isso pode ser realizado de duas formas. A primeira consiste em um mapeamento de cada um destes dois protocolos para o *draft* que define um protocolo abstrato de presença(PETERSON, 2003). Esse mapeamento pode ser encontrado nos *drafts*: XMPP(SAINT-ANDRE, 2003) e SIP(CAMPBELL; ROSENBERG, 2002). Desta

```
<message from='pubsub.uc.com' to='jose@xmpp.uc.com' id='foo'>
  <event xmlns='http://jabber.org/protocol/pubsub\#event'>
    <items node='princely\_musings'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Ocupado</title>
          <summary>
Estou ocupado
          </summary>
          <link rel='alternate' type='text/html'
                href='http://uc.com/2010/02/13/atom03'/>
          <id>tag:uc.com,2010:entry-32397</id>
          <published>2010-02-13T18:30:02Z</published>
          <updated>2010-02-13T18:30:02Z</updated>
        </entry>
      </item>
    </items>
  </event>
</message>

<message from='pubsub.uc.com' to='paulo@xmpp.uc.com' id='bar'>
  <event xmlns='http://jabber.org/protocol/pubsub\#event'>
    <items node='princely\_musings'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Ocupado</title>
          <summary>
Estou ocupado
          </summary>
          <link rel='alternate' type='text/html'
                href='http://uc.com/2010/02/13/atom03'/>
          <id>tag:uc.com,2010:entry-32397</id>
          <published>2010-02-13T18:30:02Z</published>
          <updated>2010-10-13T18:30:02Z</updated>
        </entry>
      </item>
    </items>
  </event>
</message>
```

Figura 4.9: Exemplo de mensagem XMPP de divulgação de uma atualização de presença

```
<iq type='set'
    from='francisco@uc.com/notebook'
    to='pubsub.uc.com'
    id='unsub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
     <unsubscribe
         node='princely\_musings'
         jid='francisco@uc.com'/>
  </pubsub>
</iq>
```

Figura 4.10: Exemplo de mensagem XMPP cancelando uma assinatura

forma existiria uma interoperabilidade entre o SIP e o XMPP bem como destes com qualquer outro protocolo.

Outra forma seria realizar um mapeamento direto entre XMPP e SIP, permitindo que clientes destes dois protocolos troquem atualizações de presença entre si. Porém, ao contrário do que parece, esta não é uma tarefa tão simples. Cada protocolo possui suas particularidades, as quais são complexas de mapear de um protocolo para o outro.

Um exemplo de dificuldade é o campo que identifica a sessão, o `Call-id` no SIP e o `sid` (*jingle*) no XMPP.

Uma tentativa de realizar este mapeamento é o *draft*(SAINT-ANDRE; HOURI; HILDEBRAND, 2004). Porém este *draft* não trata de alguns pontos, como a assinatura de um serviço de presença por exemplo. Como já foi visto, no caso do XMPP informações de presença são enviadas através de *stanzas* <`presence/`>. Já no caso do SIP, informações de presença são enviadas através do método *NOTIFY*.

Supondo que um cliente XMPP fique *online*, o mesmo envia para o seu servidor uma mensagem de presença semelhante a demonstrada na figura abaixo. Essa mensagem é divulgada normalmente aos demais clientes XMPP - que queiram e estão autorizados a recebe-la. Caso algum dos clientes seja um cliente SIP, está mensagem deverá passar por um *gateway* que fará o mapeamento e conversão da mesma. Esse *gateway* pode ser um elemento intermediário entre o servidor XMPP e o servidor SIP, como também pode estar implementado junto a esses.

Ao receber esta mensagem, o *gateway* deverá fazer o mapeamento e conversão antes de envia-lo ao destinatário. Na figura 4.11 é demonstrado um exemplo de conversão da mensagem exemplificada anteriormente.

Já no caso de um cliente SIP realizar uma atualização, o sentido contrário deve ser seguido. Na figura 4.12 é demonstrado uma mensagem original criada por um cliente SIP e um exemplo de conversão para XMPP.

4.2.3 Mensagem Instantânea

SIP

A troca de mensagens no SIP ocorre utilizando o método *MESSAGE*, onde no cabeçalho vão as informações padrões do protocolo e no corpo vai a mensagem propriamente dita. Para cada mensagem *MESSAGE* recebida uma mensagem `200 OK` é respondida. Não há criação de sessão.

<presence from='jpaulo@uc.com/notebook'/>	NOTIFY sip:petry@uc.com SIP/2.0 Via: SIP/2.0/TCP uc.com;branch=z9hG4bKna998sk From: <sip:jpaulo@uc.com>;tag=fzd2 To: <sip:petry@uc.com>;tag=xfg9 Call-ID: j4s0h4vny@uc.com Event: presence Subscription-State: active;expires=599 Max-Forwards: 70 CSeq: 8775 NOTIFY Contact: sip:uc.com Content-Type: application/cpim-pidf+xml Content-Length: 192 <?xml version='1.0' encoding='UTF-8'?> <presence xmlns='urn:ietf:params:xml:ns:pidf' entity='pres:jpaulo@uc.com'> <tuple id='notebook'> <status> <basic>open</basic> </status> </tuple> </presence>
XMPP	SIP

Figura 4.11: Exemplo de conversão de mensagens de presença do protocolo XMPP para o SIP

NOTIFY sip:jpaulo@uc.com SIP/2.0 Via: SIP/2.0/TCP uc.com;branch=z9hG4bKna998sk From: <sip:petry@uc.com>;tag=ffd2 To: <sip:jpaulo@uc.com>;tag=xfg9 Call-ID: j0sj4sv1m@uc.com Event: presence Subscription-State: active;expires=599 Max-Forwards: 70 CSeq: 8775 NOTIFY Contact: sip:uc.com Content-Type: application/cpim-pidf+xml Content-Length: 193 <?xml version='1.0' encoding='UTF-8'?> <presence xmlns='urn:ietf:params:xml:ns:pidf' entity='pres:petry@uc.com'> <tuple id='desktop'> <status> <basic>closed</basic> </status> </tuple> </presence>	<presence from='petry@uc.com' to='jpaulo@uc.com/desktop' type='unavailable'/>
SIP	XMPP

Figura 4.12: Exemplo de conversão de mensagens de presença do protocolo SIP para o XMPP

A realização de bate-papo em grupo, ou seja, com mais de duas pessoas é implementada utilizando o protocolo MSRP que será melhor explicado no item 4.3.

XMPP

No XMPP mensagens de Mensagem Instantaneas são transmitidas através do *stanza* identificado por <message/>. Caso haja necessidade de ser monitarar uma conversação, o campo <thread/> pode ser adicionado ao *stanza* para facilitar essa monitoração.

Conversão SIP x XMPP

O princípio de interoperabilidade entre o SIP e o XMPP para Mensagens Instantâneas é o mesmo do já falado para presença. Na figura 4.13 é demonstrado um exemplo de conversão de uma mensagem XMPP para SIP, sendo que a conversão do SIP para o XMPP segue os mesmos princípios.

Figura 4.13: Exemplo de conversão de mensagens de mensagem instantânea entre os protocolos XMPP e SIP

4.2.4 Descrição e Transmissão de Mídia

Tanto o SIP como o XMPP são responsáveis pela criação[7] e manutenção da sessão que utilizará a transmissão de uma (chamada telefônica), ou mais (vídeo conferencia), mídias. Como já foi visto, o SIP pode utilizar o protocolo SDP e o XMPP usa o Jingle para realizar esta sinalização. As mídias são transmitidas externamente a estes protocolos, comumente utilizando o protocolo RTP.

[7] A sinalização pode ser endereçada a um URI - conforme já foi falado - bem como através da númeração padrão usado na telefonia (E.164)(ITU, 2005) - especialmente em *gateways* com a PSTN - ou através de traduções do E.164 para a Internet, como o ENUM(FALTSTROM; MEALLING, 2004)

4.2.5 Conversão SIP x XMPP

Na própria extensão XEP167(SAINT-ANDRE; LUDWIG et al., 2009b) há um capítulo que trata a interoperabilidade do Jingle RTP com o SDP, propondo alguns mapeamentos que são exemplificados na figura 4.14.

```
<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
  <payload-type id='96' name='speex' clockrate='16000' ptime='40'>
    <parameter name='vbr' value='on' />
    <parameter name='cng' value='on' />
  </payload-type>
</description>
```

```
m=audio 9999 RTP/AV 96
a=rtpmap:96 speex/16000
a=ptime:40
a=fmtp:96 vbr=on;cng=on
```

Figura 4.14: Exemplo de conversão de uma mensagem Jingle para SDP

4.3 Message Session Relay Protocol

O *Message Session Relay Protocol* (MSRP) é definido na RFC4975(JENNINGS; CAMPBELL; MAHY, 2007) e é utilizado para sessões de Mensagens Instantâneas - um para um ou um para muitos -, transferência de arquivos ou compartilhamento de imagens. Uma sessão MSRP é iniciada utilizando, dentre outros possíveis protocolos, a combinação SIP e SDP da mesma forma que uma sessão de áudio ou vídeo, e após o estabelecimento da sessão as mensagens são trocadas da mesma forma que no protocolo RTP. Um exemplo de criação de sessão MSRP é demonstrado na figura 4.15.

```
1  INVITE sip:petry@uc.com SIP/2.0
2  To: <sip:petry@uc.com>
3  From: <sip:jpaulo@uc.com>;tag=786
4  Call-ID: 3413an89KU
5  Content-Type: application/sdp
6
7  c=IN IP4 uc.com
8  m=message 7654 TCP/MSRP *
9  a=accept-types:text/plain
10 a=path:msrp://uc.com:7654/jshA7weztas;tcp
```

Figura 4.15: Exemplo de criação de uma sessão MSRP

Uma possibilidade interessante é quando um usuário quer falar com outro porém sem saber se o usuário que será chamado pode realizar uma chamada de voz ou não. Dessa forma, no

decorrer da inicialização de uma sessão usando o SIP, através do protocolo SDP o usuário que está chamando realiza um convite tanto para voz como para mensagem instantânea. Caso o usuário não possa criar uma sessão de voz, a sessão de mensagem instantânea pode ser criada.

5 *Cenário prático*

O cenário proposto é composto por dois clientes SIPs conectados a um servidor SIP e este conectado a um servidor XMPP, como componente(SAINT-ANDRE, 2005) , com dois clientes XMPPs. O objetivo é verificar o funcionamento em um ambiente com um único protocolo (entre os dois clientes do mesmo servidor) e testar a interoperabilidade entre os protocolos realizando uma comunicação entre os clientes dos dois servidores.

Inicialmente foi utilizado o Asterisk como servidor SIP e o Openfire como servidor XMPP. O primeiro por ser um famoso servidor SIP e o segundo por possuir um *plugin* de integração com o primeiro.

Ao se testar os serviços de MI e presença no Asterisk, em um ambiente puramente SIP, foi descoberto que o mesmo não implementa o métodos PUBLISH, conforme pode ser visto na figura 5.1 , impossibilitando o uso do Asterisk para estes testes. Na busca por um novo servidor SIP foi encontrado o OpenSIPS(GONCALVES, 2008) [1] que possui uma boa documentação e implementa os métodos para MI e presença. Devido ao tempo dispensado na procura e aprendizado deste novo servidor SIP, escolheu-se a substituição do servidor XMPP para um que a instalação e configuração fosse mais simples. Deste modo foi escolhido o servidor Prosody. É um servidor implementado na linguagem de programação Lua[2] e sua configuração é extremamente simples baseada em um único arquivo.

A configuração do OpenSIPS é baseada em um arquivo único que contém as instruções que devem ser seguidas para cada mensagem SIP que chega no servidor. Este arquivo inicia com a configuração de parâmetros globais, conforme apresentado na figura 5.2, como nível do log, porta que deverá escutar e outras.

Em seguida existem as instruções de carregamentos de módulos. Cada módulo é responsável por uma ou mais funções. Existe, por exemplo, módulo responsável pela interface com um banco de dados, módulo responsável pela implementação do serviço de presença, etc.

[1] Continuação do projeto OpenSER. Foi escolhido por possuir maior documentação disponível.
[2] http://www.lua.org

```
(...)
enum sipmethod {
    SIP_UNKNOWN,       /*!< Unknown response */
    SIP_RESPONSE,       /*!< Not request, response to outbound request */
    SIP_REGISTER,       /*!< Registration to the mothership, tell us where you are
         located */
    SIP_OPTIONS,       /*!< Check capabilities of a device, used for "ping" too */
    SIP_NOTIFY,       /*!< Status update, Part of the event package standard,
        result of a SUBSCRIBE or a REFER */
    SIP_INVITE,       /*!< Set up a session */
    SIP_ACK,       /*!< End of a three-way handshake started with INVITE. */
    SIP_PRACK,       /*!< Reliable pre-call signalling. Not supported in Asterisk.
         */
    SIP_BYE,       /*!< End of a session */
    SIP_REFER,       /*!< Refer to another URI (transfer) */
    SIP_SUBSCRIBE,       /*!< Subscribe for updates (voicemail, session status,
        device status, presence) */
    SIP_MESSAGE,       /*!< Text messaging */
    SIP_UPDATE,       /*!< Update a dialog. We can send UPDATE; but not accept it
        */
    SIP_INFO,       /*!< Information updates during a session */
    SIP_CANCEL,       /*!< Cancel an INVITE */
    SIP_PUBLISH,       /*!< Not supported in Asterisk */
    SIP_PING,       /*!< Not supported at all, no standard but still implemented
        out there */
};
(...)
```

Figura 5.1: Trecho do arquivo chan_sip.c onde pode-se confirmar que o Asterisk não tem suporte ao método PUBLISH

```
(...)
######## Global Parameters ##########

#debug=3
log_stderror=yes
log_facility=LOG_LOCAL0

#fork=yes
children=4

/* uncomment the following lines to enable debugging */
debug=6
fork=no
#log_stderror=yes

/* uncomment the next line to disable TCP (default on) */
#disable_tcp=yes
(...)
```

Figura 5.2: Trecho do arquivo de configuração do OpenSIPS onde parâmetros globais são configurados

Após o carregamento dos módulos é possível especificar alguns parâmetros destes módulos, como a definição da senha de acesso ao banco de dados. Essas configurações são demonstradas na figura 5.3.

```
(...)
# Modules loading
loadmodule "db_mysql.so"
loadmodule "signaling.so"
loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "mi_fifo.so"
loadmodule "uri.so"
loadmodule "xlog.so"
loadmodule "acc.so"
loadmodule "pua.so"
loadmodule "xmpp.so"
loadmodule "pua_xmpp.so"

# ----------------- setting module-specific parameters ---------------
modparam("presence|presence_xml", "db_url",
   "mysql://opensips:opensipsrw@1.1.1.1/opensips")
modparam("presence_xml", "force_active", 1)
modparam("presence", "server_address", "sip:1.1.1.2:5060")
(...)
```

Figura 5.3: Trecho do arquivo de configuração do OpenSIPS onde os módulos são carregados e configurados

Após esta configuração inicial vem os blocos de roteamento, responsáveis pelo tratamento de cada mensagem SIP que chega ao servidor. Nesta parte é especificado como cada método será tratado e o encaminhamento que será feito. Alterações de parâmetros, ou até mesmo inserção de novos parâmetros, podem ser realizados. Na figura 5.4 é demonstrado um exemplo desta parte da configuração.

Com a configuração básica do OpenSIPS foi possível realizar um teste de MI e presença entre dois clientes SIP. As trocas de mensagens ocorrida entre os clientes durante os testes estão demonstradas no diagrama de interação demonstrada na figura 5.5. Após realizar o registro do segundo cliente SIP, foi feito o teste de inscrição para receber notificações de presença entre os dois clientes. A solicitação foi realizada ao servidor, que encaminhou ao segundo cliente. Este então respondeu com um `200 OK` informando que a solicitação foi aceita. A partir de então, a cada alteração do *status* de um dos clientes o outro seria notificado.

Para testar o envio desta notificação de presença, o status de um dos clientes foi alterado.

```
(...)
# main request routing logic

route{
(...)
   if (is_method("CANCEL"))
   {
      if (t_check_trans())
         t_relay();
      exit;
   }
(...)
   if (is_method("REGISTER"))
   {
      if (!save("location"))
         sl_reply_error();

      exit;
   }
(...)
}
```

Figura 5.4: Trecho do arquivo de configuração do OpenSIPS onde o tratamento das mensagens recebidas pelo servidor é declarado

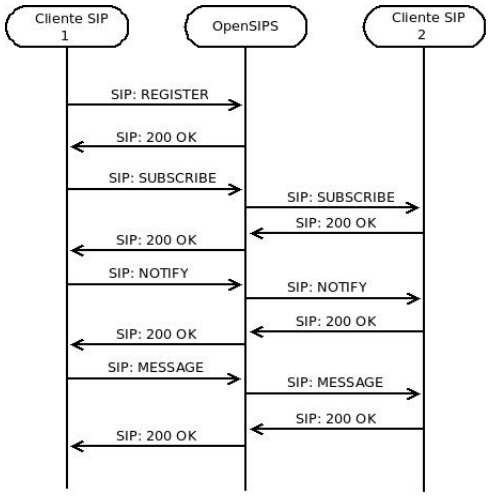

Figura 5.5: Diagrama de interação demonstrando as mensagens trocadas durante o teste em um ambiente puramente SIP

Este cliente gerou uma mensagem NOTIFY que foi enviada ao servidor. O servidor então encaminhou então esta mensagem, notificando o outro cliente a mudança que ocorreu. Desta forma verificou-se que o serviço de presença entre dois clientes SIPs funcionou perfeitamente.

Feito os testes de presença deu-se início aos testes de mensagem instantânea entre dois clientes SIP. Para isso, foi enviado uma mensagem de um dos clientes para o outros. O pacote desta mensagem foi encaminhado ao servidor e este reencaminhou para o cliente destinatário. A partir deste outro cliente foi enviada uma resposta, que percorreu o caminho de volta chegando ao primeiro cliente com sucesso. Desta forma confirmou-se o funcionamento da troca de mensagem instantânea.

Com o ambiente puramente SIP em funcionamento, foram iniciados os testes em um ambiente puramente XMPP. Utilizando a configuração padrão do Prosody, apenas adicionando para que o mesmo responda pelo domínio *xmpp.uc.com*, foram registrados dois usuários neste servidor para verificar a troca de mensagens de presença e de MI entre os dois. Na figura 5.6 é demonstrado um diagrama de interação das mensagens trocadas durante os testes,

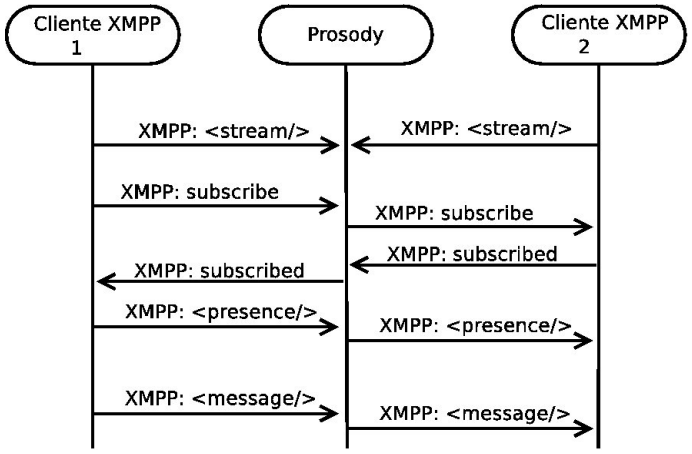

Figura 5.6: Diagrama de interação demonstrando as mensagens trocadas durante os testes em um ambiente puramente XMPP

Após o registro dos dois clientes no servidor, a partir de um cliente foi gerado uma solicitação de assinatura do outro cliente. Uma mensagem foi enviada ao segundo cliente solicitando permissão para a assinatura. Após aprovação do segundo cliente, ambos tiveram adicionados em sua lista de contatos o contato do outro cliente.

O primeiro teste realizado foi o de alteração de status de um cliente, de modo que pode-se verificar essa alteração no outro cliente. Após essa alteração um pacote é enviado aos assinantes, ou seja, neste caso o segundo cliente. Ao receber este pacote, houve uma alteração no status de presença do cliente que gerou esta mudança na lista de contatos comprovando o funcionamento da funcionalidade de presença em um ambiente puramente XMPP.

O último teste realizado neste ambiente comprovou o funcionamento de mensagens instantâneas. Foi gerado uma mensagem a partir de um dos clientes com destino ao outro cliente, a qual gerou um pacote que foi recebido e apresentado corretamente pelo cliente de destino.

Com os testes em ambientes com um único protocolo concluídos com sucesso, deu-se início a verificação da possibilidade de interoperabilidade entre eles. Para este cenário uma conexão entre o servidor SIP (OpenSIPS) e o servidor XMPP (Prosody) é necessária. Para isto ocorrer, um dos dois servidores será o responsável pela conversão de um protocolo para o outro. O OpenSIPS implementa esta possibilidade, podendo realizar uma comunicação *server-to-server* utilizando o protocolo XMPP, se conectando ao servidor Prosody como um componente.(SAINT-ANDRE, 2005).

Conforme pode-se observar na imagem 5.7, a conversão entre os protocolos para a mensagem enviada a partir de um cliente SIP para um cliente XMPP ocorreu com sucesso.

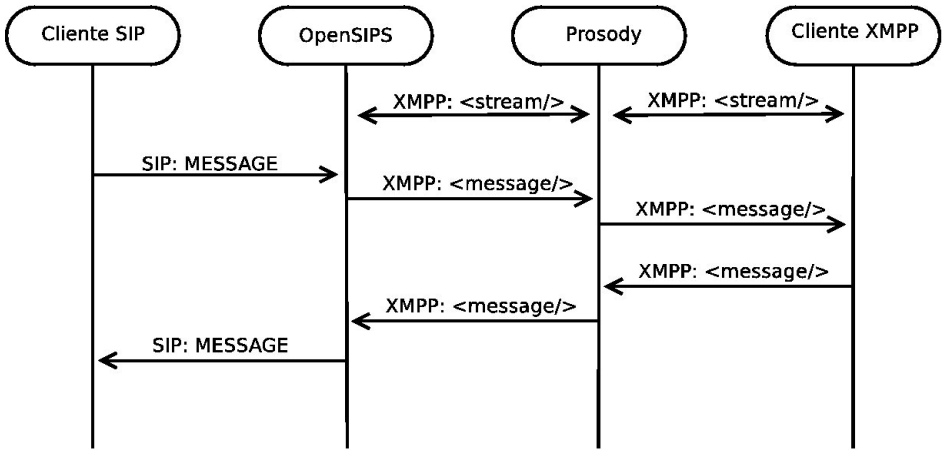

Figura 5.7: Diagrama de interação demonstrando as conversões de um protocolo para outro ocorridas

Testes realizados iniciando a mensagem tanto do lado do cliente SIP como do cliente XMPP demonstraram o funcionamento da conversão entre os protocolos. Verifica-se então o perfeito funcionamento da conversão de pacotes de Mensagem Instantânea entre cliente SIP e XMPP, possibilitando o uso em um mesmo ambiente de clientes que implementam os dois protocolos.

6 *Conclusões*

Este trabalho teve como objetivo realizar um estudo comparativo dos protocolos abertos SIP/SIMPLE e XMPP de modo a verificar a possibilidade de interoperabilidade entre eles para implementação em soluções de UC. Com base no estudo realizado é possível afirmar que a interoperabilidade entre os protocolos SIP e XMPP é possível. Porém ela ainda não é operacional pois, como foi visto no decorrer do trabalho, é tudo muito teórico ainda. Os desenvolvedores dos servidores e clientes que usam esses protocolos não implementam muitas das recomendações de convergência entre eles já que praticamente todas ainda são rascunhos (drafts), ou seja, podem sofrer alterações.

Outra questão é que não há uma definição de como essa interoperabilidade entre os protocolos deve realmente ocorrer. Algumas recomendações sugerem a conversão direta entre os protocolos enquanto outras sugerem a criação de um meio termo, um protocolo central, onde todos os demais protocolos possuem mapeamento para este e vice-versa. Esta última opção, inclusive, é utilizada nas redes *IP Multimedia Subsystem* (IMS)(MAYER; POIKSELKA, 2009) para a tradução de outros protocolos.

É claro que a opção de utilizar um ambiente com um único protocolo que abranja todos os serviços de uma solução de UC é muito interessante. Para isso existem as extensões dos protocolos, que permitem inserir novas funcionalidades aos mesmo. O SIP tem grande preferência para ser o escolhido, pois é um protocolo já bem conhecido, sendo robusto e escalável. Tanto é que foi o protocolo escolhido pela 3GPP para a arquitetura de IMS.

Após o estudo do funcionamento e das características do SIP, pode-se afirmar que trata-se de um protocolo excelente para uso em soluções de comunicação. Ele é simples, permitindo o estabelecimento de uma sessão - independente da mídia - em poucos passos. Sua formatação em texto puro facilita o seu entendimento bem como também a análise de falhas. É totalmente escalável possibilitando a inserção de novas funcionalidades sem muitas complicações. Sua implementação não necessita de mudanças na rede de transporte, apenas necessitando de conectividade entre os seus elementos para que eles possam trocas suas mensagens.

Pode-se descobrir também algumas de suas desvantagens. Uma delas é o fato de não confiar no protocolo de transporte, independente de qual esteja usando. Ao utilizar, por exemplo, o TCP, mesmo que esse garanta a entrega dos pacotes enviados, a implementação do SIP exige uma resposta de confirmação, sobrecarregando a rede com pacotes que poderiam ser evitados. Outra, apesar de ter sido citado anteriormente como um aspecto positivo, é o fato de sua formatação sem baseada em texto puro. O processamento destas mensagem exige uma certa carga de processamento dos servidores, o que não aconteceria caso utilizasse uma formatação binária por exemplo.

Já o XMPP possui uma característica muito interessante, que é o fato de manter uma conexão ativa entre o cliente e o servidor por todo o período que o cliente desejar. Isso é interessante pois é um avanço na forma de comunicação. O email por exemplo, o cliente necessita conectar periodicamente ao servidor para verificar se há alguma mensagem nova. Uma página web também, é sempre necessário fazer um questionamento (*polling*) ao servidor para saber se houve atualização. Esse *polling* acaba gerando sobrecarga na rede e principalmente no servidor, ao se pensar em larga escala. O fato de o XMPP manter uma conexão ativa entre o cliente e o servidor, permite que o servidor envie instantâneamente para o cliente novas mensagens que cheguem, ou então que o informe de atualizações que houveram.

Assim como o SIP o XMPP também é facilmente expansível, vide a quantidade de extensões já existentes, já que sua estrutura baseada em XML permite facilmente a criação de novas *tags* criando novas funcionalidades.

O fato de o XMPP ser totalmente baseado em XML, dificulta a implementação de alguns serviços como a transferência de arquivos, por exemplo. Arquivos XML não são bons lugares para se transmitir informações binárias, sendo a criação de um canal de comunicação *out-of-band* a melhor maneira de realizar esta transferência. Claro que existe a possibilidade de realizar este envio codificando o arquivo desejado em *base64* por exemplo, porém há um consumo maior processamento e largura de banda.

Mas o XMPP não deixa muito a desejar. Seu uso vem crescendo exponencialmente, sendo o protocolo de comunicação escolhido por grandes provedores de serviço como por exemplo o Facebook[1] e o GTalk[2], este último inclusive implementando chamadas de voz e vídeo.

Independente do protocolo escolhido, ou escolhendo os dois, a implementação de um sistema de UC não foge de um dos grandes problemas enfrentado por administradores de rede: o *NAT*. Ambos possuem maneiras de se resolver - ou melhor, se contornar - esta dificuldade, como

[1] http://blog.facebook.com/blog.php?post=297991732130

[2] http://code.google.com/intl/pt-BR/apis/talk/talk_developers_home.html

STUN ou ICE, porém é um assunto que gera muita dor de cabeça a quem está implementando uma comunicação entre duas pontas que possuem um *NAT*, ou um *firewall*, entre elas.

Como foi visto no estudo do XMPP, ele foi inicialmente desenvolvido para MI e presença, e faz isso muito bem. Assim como o SIP foi desenvolvido para negociação de mídias e a faz muito bem também. A princípio são protocolos diferentes com funções diferentes. Mas ambos possuem várias extensões que agregam novas funções tornando-os protocolos semelhantes e com os mesmo objetivos. Um exemplo é a extensão Jingle(SAINT-ANDRE; LUDWIG et al., 2009a) que possibilita a negociação de mídias utilizando o XMPP. Ou então as extensões criadas pelo Grupo de Trabalho SIMPLE, que possibilita o uso de presença no SIP.

No estudo comparativo entre os dois protocolos e suas extensões, conclui-se que há total possibilidade de convergência entre eles visto que executam as mesmas funções e possuem parâmetros parecidos, possibilitando o mapeamento de mensagens entre os dois. Inclusive verificou-se a existência de várias recomendações, que ainda são rascunhos(drafts), de como realizar este mapeamento.

Porém, ao realizar os testes, verificou-se que esta conversão não está completamente implementada e não está em um nível que permita configurá-la em um ambiente de produção. No primeiro cenário montado, utilizando o software Asterisk, foi visto que este servidor SIP em específico não possui implementação de algumas extensões do protocolo SIP, o que impossibilitou seu uso neste trabalho. Por este motivo foi perdido um tempo considerável na busca de um novo servidor para teste e no entendimento do seu funcionamento.

O servidor escolhido foi o OpenSIPS, que demonstrou ser um excelente servidor SIP. Porém sua configuração é extremamente complicada, baseado em uma linguagem de script própria e que exige um bom conhecimento do protocolo. Em contrapartida, como servidor XMPP foi escolhido o Prosody, por ser extremamente simples de configurar de modo a não ser perdido mais tempo para a execução dos testes.

As maiores dificuldades encontradas, além do fato de algumas conversões não serem implementadas, é a falta de documentação específica sobre este assunto e a falta de opções de aplicações (principalmente de servidores) para uso.

Contudo, os testes que foram realizados confirmaram a possibilidade de haver interoperabilidade entre os dois protocolos utilizando protocolos e aplicações abertas. A medida que as recomendações de mapeamento entre os protocolos deixarem de ser rascunho, novas aplicações as implementando deverão surgir, possibilitando a implementação de ambientes de Comunicações Unificadas totalmente baseados em protocolos e aplicações abertas, gerando uma econo-

mia expressiva para as empresas que as implementarem.

6.1 Trabalhos Futuros

Ficam como sugestões de trabalhos futuros:

- Desenvolvimento de gateways para tradução entre os protocolos SIP e XMPP.
- Implementação de um servidor de MI e Presença no IFSC, facilitando a comunicação entre alunos, professores e servidores.
- Estudo de aplicações que façam polling e que possam ser substituídas por soluções em XMPP
- Desenvolvimento de agenda que centralize informações de contato e o status (informação de Presença) de usuários de uma instituição

Lista de Abreviaturas

SIP *Session Initiation Protocol*

UC *Unified Communications*

MI Mensagens Instantâneas

XMPP *Extensible Messaging and Presence Protocol*

SIMPLE *SIP for Instant Messaging and Presence Leveraging Extensions*

XEP *XMPP Extension Protocol*

RFC *Request for Comment*

VoIP Voz sobre IP

TI Tecnologia da Informação

GPS *Global Positioning System*

PABX *Private Automatic Branch eXchange*

IP *Internet Protocol*

GPRS *General Packet Radio Service*

SS7 Sistema de Sinalização número 7

NGN *Next Generation Network*

IETF *Internet Engineering Task Force*

MSRP *Message Session Relay Protocol*

SMTP *Simple Mail Transfer Protocol*

SMS *Short Message Service*

TCP *Transmission Control Protocol*

UAS *User Agent Server*

UAC *USer Agent Client*

UA *User Agent*

URI *Uniform Resource Indicator*

UDP *User Datagram Protocol*

SCTP *Stream Control Transmission Protocol*

SDP *Session Description Protocol*

3GPP *3rd Generation Partnership Project*

XML *eXtensible Markup Language*

JID *Jabber ID*

TLS *Transport Layer Security*

SASL *Simple Authentication and Security Layer*

NAT *Network Address Translation*

BOSH *Bidirectional-streams Over Synchronous HTTP*

STUN *Session Traversal Utilities for NAT*

ICE *Interactive Connectivity Establishment*

sRTP *Secure Real Time Protocol*

FTP *File Transfer Protocol*

RTCP *RTP Control Protocol*

IMS *IP Multimedia Subsystem*

Referências Bibliográficas

BAUGHER, M.; MCGREW, D. et al. *The Secure Real-time Transport Protocol*. [S.l.], 2004. Disponível em: <http://www.ietf.org/rfc/rfc3711.txt>.

BRAY, T.; PAOLI, J. et al. *Extensible Markup Language*. [S.l.], 2008. Disponível em: <http://www.w3.org/TR/REC-xml/>.

CAMARILLO, G.; ROSENBERG, J. *The Alternative Network Address Types (ANAT) Semantics for the Session Description Protocol (SDP) Grouping Framework*. [S.l.], 2005. Disponível em: <http://www.ietf.org/rfc/rfc4091.txt>.

CAMPBELL, B.; ROSENBERG, J. *CPIM Mapping of SIMPLE Presence and Instant Messaging*. [S.l.], 2002. Disponível em: <http://tools.ietf.org/html/draft-ietf-simple-cpim-mapping-01>.

COLCHER, S.; GOMES, A. *Voip - Voz sobre IP*. [S.l.]: Editora Campus, 2005.

CRISPIN, M. *Internet Message Access Protocol*. [S.l.], 2004. Disponível em: <http://tools.ietf.org/html/rfc3501>.

DAY, M. et al. *RFC2779: Instant Messaging / Presence Protocol Requirements*. [S.l.], 2000. Disponível em: <http://www.ietf.org/rfc/rfc2779.txt>.

DAY, M.; ROSENBERG, J.; SUGANO, H. *RFC2778: A Model for Presence and Instant Messaging*. [S.l.], 2000. Disponível em: <http://www.ietf.org/rfc/rfc2778.txt>.

DIERKS, T.; ALLEN, C. *The TLS Protocol*. [S.l.], 1999. Disponível em: <http://www.ietf.org/rfc/rfc2246.txt>.

FALTSTROM, P.; MEALLING, M. *The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)*. [S.l.], 2004. Disponível em: <http://www.ietf.org/rfc/rfc3761.txt>.

FIELDING, R.; MASINTER, L. et al. *Uniform Resource Identifier (URI): Generic Syntax*. [S.l.], 2005. Disponível em: <http://www.ietf.org/rfc/rfc3986.txt>.

FREED, N.; BORENSTEIN, N. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. [S.l.], 1996. Disponível em: <http://www.ietf.org/rfc/rfc2045.txt>.

GARCIA-MARTIN, M.; LORETO, S. et al. *A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer*. [S.l.], 2009. Disponível em: <http://tools.ietf.org/html/rfc5547>.

GONCALVES, F. E. *Building Telephony Systems with OpenSER*. [S.l.]: Packt Publishing, 2008.

HANDLEY, M.; JACOBSON, V.; PERKINGS, C. *SDP: Session Description Protocol*. [S.l.], 2006. Disponível em: <http://www.ietf.org/rfc/rfc4566.txt>.

ITU. *E.164: The international public telecommunication numbering plan*. [S.l.], 2005. Disponível em: <http://www.itu.int/rec/T-REC-E.164/en>.

JENNINGS, C.; CAMPBELL, B.; MAHY, R. *The Message Session Relay Protocol (MSRP)*. [S.l.], 2007. Disponível em: <http://www.ietf.org/rfc/rfc4975.txt>.

JENNINGS, C.; MAHY, R.; ROACH, A. B. *Relay Extensions for the Message Session Relay Protocol (MSRP)*. [S.l.], 2007. Disponível em: <http://www.ietf.org/rfc/rfc4976.txt>.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores: uma abordagem Top-Down*. [S.l.]: Pearson Education, 2006.

MAYER, G.; POIKSELKA, M. *The IMS IP Multimedia Concepts and Services*. [S.l.]: Wiley, 2009.

MYERS, J.; ROSE, M. *Post Office Protocol - Version 3*. [S.l.], 1996. Disponível em: <http://www.ietf.org/rfc/rfc1939.txt>.

NIEMI, A. *Session Initiation Protocol (SIP) Extension for Event State Publication*. [S.l.], 2004. Disponível em: <http://www.ietf.org/rfc/rfc3903.txt>.

ONG, L.; YOAKUM, J. *An Introduction to the Stream Control Transmission Protocol*. [S.l.], 2002. Disponível em: <http://www.ietf.org/rfc/rfc3286.txt>.

PETERSON, J. *Common Profile for Presence (CPP)*. [S.l.], 2003. Disponível em: <http://tools.ietf.org/html/draft-ietf-impp-pres-04>.

POSTEL, J. B. *Simple Mail Transfer Protocol*. [S.l.], 1982. Disponível em: <http://www.ietf.org/rfc/rfc821.txt>.

ROACH, A. B. *Session Initiation Protocol (SIP)-Specific Event Notification*. [S.l.], 2002. Disponível em: <http://www.ietf.org/rfc/rfc3265.txt>.

ROSENBERG, J. *Session Traversal Utilities for NAT (STUN)*. [S.l.], 2008. Disponível em: <http://tools.ietf.org/html/rfc5389>.

ROSENBERG, J.; CAMARILLO, G. et al. *SIP: Session Initiation Protocol*. [S.l.], 2002. Disponível em: <http://www.ietf.org/rfc/rfc3261.txt>.

ROSENBERG, J.; HUITEMA, C. et al. *Session Initiation Protocol (SIP) Extension for Instant Messaging*. [S.l.], 2002. Disponível em: <http://www.ietf.org/rfc/rfc3428.txt>.

SAINT-ANDRE, P. *Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)*. [S.l.], 2003. Disponível em: <http://tools.ietf.org/html/draft-ietf-xmpp-cpim-03>.

SAINT-ANDRE, P. *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. [S.l.], 2004. Disponível em: <http://www.ietf.org/rfc/rfc3921.txt>.

SAINT-ANDRE, P. *Extensible Messaging and Presence Protocol (XMPP): Core*. [S.l.], 2004. Disponível em: <http://www.ietf.org/rfc/rfc3920.txt>.

SAINT-ANDRE, P. *Jabber Component Protocol*. [S.l.], 2005. Disponível em: <http://xmpp.org/extensions/xep-0114.html>.

SAINT-ANDRE, P. *Out of Band Data*. [S.l.], 2006. Disponível em: <http://xmpp.org/extensions/xep-0066.html>.

SAINT-ANDRE, P. *XMPP Ping*. [S.l.], 2009. Disponível em: <http://xmpp.org/extensions/xep-0199.html>.

SAINT-ANDRE, P. *Jingle File Transfer*. [S.l.], 2010. Disponível em: <http://xmpp.org/extensions/xep-0234.html>.

SAINT-ANDRE, P.; HOURI, A.; HILDEBRAND, J. *Interoperability between the Extensible Messaging and Presence Protocol (XMPP) and SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)*. [S.l.], 2004. Disponível em: <http://xmpp.org/internet-drafts/attic/draft-saintandre-xmpp-simple-00.html>.

SAINT-ANDRE, P.; HOURI, A.; HILDEBRAND, J. *Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Presence*. [S.l.], 2009. Disponível em: <http://xmpp.org/internet-drafts/draft-saintandre-sip-xmpp-presence-02.txt>.

SAINT-ANDRE, P.; KAES, C. *Flexible Offline Message Retrieval*. [S.l.], 2005. Disponível em: <http://xmpp.org/extensions/xep-0013.html>.

SAINT-ANDRE, P.; KARNEGES, J. *In-Band Bytestreams*. [S.l.], 2009. Disponível em: <http://xmpp.org/extensions/xep-0047.html>.

SAINT-ANDRE, P.; LUDWIG, S. et al. *Jingle*. [S.l.], 2009. Disponível em: <http://xmpp.org/extensions/xep-0166.html>.

SAINT-ANDRE, P.; LUDWIG, S. et al. *Jingle RTP Sessions*. [S.l.], 2009. Disponível em: <http://xmpp.org/extensions/xep-0167.html>.

SAINT-ANDRE, P.; MILLARD, P.; MEIJER, R. *Publish-Subscribe*. [S.l.], 2010. Disponível em: <http://xmpp.org/extensions/xep-0060.html>.

SCHULZRINNE, H. et al. *RTP: A Transport Protocol for Real-Time Applications*. [S.l.], 2003. Disponível em: <http://www.ietf.org/rfc/rfc3265.txt>.

SHORETEL. *Getting the Foundation Right: Unified Communications*. 2009. White Paper.

SINGH, K. *SIP vs. XMPP or SIP and XMPP?* [S.l.], 2009. Disponível em: <http://p2p-sip.blogspot.com/2009/11/sip-vs-xmpp-or-sip-and-xmpp.html>.

SOLUTIONS, N. N. W. *Unified Network Presence Management*. 2000. White Paper.

SUGANO, H.; PETERSON, J. et al. *Presence Information Data Format (PIDF)*. [S.l.], 2004. Disponível em: <http://www.ietf.org/rfc/rfc3863.txt>.

UNION, I. T. *Introduction to CCITT Signalling System No. 7*. [S.l.], 1993. Disponível em: <http://www.itu.int/rec/T-REC-Q.700/en>.

WILKINSON, N. *Next Generation Network Services: technologies and strategies*. Chichester: John Wiley & Sons, 2002.

XMPP Standards Foundation. *XMPP and Jabber*. [S.l.], 1998. Disponível em: <http://xmpp.org/about/jabber.shtml>.