DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Desenvolvimento de um Cliente para Integração de Sistemas de Comunicação Unificada com Plataformas de Telefonia

Trabalho Submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina
DAS 5511: Projeto de Fim de Curso

Lauvir Ramos Neto

Florianópolis, agosto de 2012.

Desenvolvimento de um Cliente para Integração de Sistemas de Comunicação Unificada com Plataformas de Telefonia

Lauvir Ramos Neto

Orientadores:

_____

Elisa Manfrin de Araújo, Eng.

Orientadora na Intelbras

_____

Prof. Joni da Silva Fraga

Orientador na UFSC

Este relatório foi julgado no contexto da disciplina
DAS 5511: Projeto de Fim de Curso
e aprovado na sua forma final pelo
Curso de Engenharia de Controle e Automação

Agradecimentos

Aos meus pais, Louvenir e Graça, pelo amor, amparo e, principalmente, por tudo o que fizeram por mim para que eu conseguisse ultrapassar mais essa etapa de minha vida.

À minha companheira, Renata, pelo amor, carinho e por ficar incondicionalmente ao meu lado em todos os bons e maus momentos que passamos juntos.

À minha irmã Greice, pelo amor, ajuda e conselhos que me guiaram nos caminhos que segui.

Ao meu afilhado, João Bernardo, que mesmo ainda sem saber tudo o que se passa em sua volta, consegue me dar forças com um simples sorriso.

Aos meus orientadores, Elisa e Gilson, pelos conselhos, ensinamentos e companheirismo nos momentos em que trabalhamos juntos.

Ao professor Joni, pela ajuda e orientação nesse trabalho.

Á Intelbras, mais especificamente à equipe de Comunicação Unificada, pela oportunidade, espaço e infraestrutura cedida para a realização do projeto.

À Universidade Federal de Santa Catarina, por possibilitar que esse trabalho fosse realizado.

Aos colegas e amigos em geral, por toda a ajuda e companheirismo nos momentos compartilhados.

Resumo

Cada vez mais existem recursos que possibilitam a uma pessoa se comunicar com outra. No entanto, há poucos esforços no sentido de fazer com que várias dessas novas tecnologias trabalhem com alguma integração entre si.

Normalmente, quando tecnologias de comunicação não são interligadas, ocorrem vários problemas relacionados à perda de tempo e gastos desnecessários com recursos de comunicação, o que tende a ser um comportamento nocivo para o ambiente corporativo.

Nesse contexto surge o conceito de comunicação unificada, propondo que dispositivos e serviços de comunicação trabalhem de forma conjunta, propiciando mais agilidade e assertividade no ambiente de comunicações corporativas.

A Intelbras, grande empresa brasileira no setor de telecomunicações, tem concentrado esforços para aderir à filosofia de comunicações unificadas. Para tanto, a empresa vem desenvolvendo sua própria solução para se adequar a esse conceito, e disponibilizá-la no mercado corporativo.

Este trabalho refere-se a uma parcela dessa solução de comunicações unificadas, que aborda o desenvolvimento de softwares para permitir que usuários acessem o sistema como um todo, ou seja, o objetivo é fazer com que esses softwares desenvolvidos se integrem a uma solução de comunicações unificadas, permitindo que o usuário possa interagir com outras pessoas, além de gerenciar seus recursos de comunicação.

Para tanto, será apresentada a forma como foram implementados os softwares de acesso à solução, ressaltando-se suas funcionalidades, utilidade e viabilidade de uso no sistema.

Palavras-chave: integração, perda de tempo e gastos, ambiente corporativo, comunicação unificada, solução de comunicações unificadas, software de acesso.

Abstract

There are many devices that allow a person to communicate with someone else. However, there are also few efforts to make several of these new technologies work integrated.

Normally, when communication technologies are not linked, there are several problems related to time loss and unnecessary expenses with resources for communication, which tends to be a harmful behavior to the corporate environment.

In this context the concept of unified communications arises, proposing that devices and communication services work jointly, providing more flexibility and assertiveness in the corporate communications environment.

Intelbras, one of the largest Brazilian companies in the telecommunications industry, has focused efforts to adhere to the philosophy of unified communications. Thus, the company has developed its own solution to fit this concept and make it available in the corporate market.

This report refers to a portion of this unified communications solution, in which software was developed to allow users to access the system as a whole, i.e., the goal is to have the software developed integrated with the unified communications solution allowing the user to interact with others and manage their communications devices.

To do so, the way that this software was developed will be presented, highlighting their features, utility and feasibility of use.

Keywords: integration, loss of time and expenses, corporate environment, unified communications, unified communications solution, access software.

Sumário

Capítulo 1: Introdução

Com o advento de várias tecnologias voltadas para o setor de comunicações, surgem novos, e cada vez mais sofisticados dispositivos que promovem a interação entre pessoas no sentido de trocar informações e dados.

No entanto, a grande maioria desses aparelhos e serviços funcionam de forma isolada, sem poderem atuar conjuntamente com outras ferramentas de comunicação. Um bom exemplo que pode ser citado é a quase inexistência de integração entre serviços de telefonia e e-mail. Esse caso, na verdade, demonstra apenas um de muitos outros empecilhos trazidos com a multiplicidade de recursos de comunicação, como problemas de replicação de dados em vários dispositivos e demora em encontrar a pessoa com que se deseja trocar informações.

No meio corporativo, essa situação possui ainda mais agravantes, pois com um mercado cada vez mais dinâmico e exigente, faz-se necessário que as empresas tenham meios de conseguirem agilizar os seus procedimentos de contatar clientes, fornecedores e os próprios colegas de trabalho. Normalmente, quando esses processos não são bem formalizados dentro de um negócio, ocorrem problemas como gastos desnecessários e perda de tempo, o que tende a ser fatal para a continuidade de uma empresa.

No sentido de solucionar esses problemas que interferem significativamente no meio corporativo, surge a filosofia de comunicações unificadas, a qual propõe uma integração entre os mais diversos meios de comunicação que uma pessoa possa utilizar no ambiente corporativo.

Geralmente empresas que adotam soluções de comunicação unificada conseguem reduzir de maneira expressiva seus problemas relacionados à comunicação, pois propiciam aos seus clientes maior rapidez no fechamento de negócios, e aos funcionários maior agilidade em tarefas cotidianas.

A Intelbras, uma das grandes empresas nacionais de telecomunicações, tem por objetivo seguir o paradigma de comunicações unificadas, propondo uma solução para integração de dispositivos e meios de comunicação.

Essa solução vem somar às empresas uma forma bastante versátil de intensificar e acelerar o modo como elas promovem troca de dados e informações, pois englobará a maioria dos recursos de comunicação em apenas uma solução, que fará o gerenciamento de todas as mídias e aparelhos utilizados.

Para cumprir com esse objetivo, o sistema de comunicação unificada da Intelbras deverá contar com um conjunto de softwares e hardwares que comporão a solução e permitirão que se integrem serviços como, por exemplo, telefonia, e-mail e mensagens instantâneas.

Uma parte desse conjunto são os softwares que permitem o acesso ao serviço de comunicação unificada, sendo esses os objetivos de estudo desse trabalho. Esses softwares, tratados ao longo de todo o texto como cliente de acesso à solução, permitirão ao usuário gerenciar e utilizar todos os seus dispositivos de comunicação integrados ao sistema, fazendo um intermédio entre ele e o restante da solução. Em outras palavras, os softwares do cliente de acesso terão a incumbência de informar ao usuário dados sobre seus aparelhos e serviços de comunicação, além de levar à solução as requisições feitas pelo usuário.

Um fato interessante a se acrescentar é que a solução de comunicações unificadas da Intelbras vem sendo desenvolvida durante os dois últimos anos, ganhando o amadurecimento necessário para cumprir suas metas. Entretanto, o cliente de acesso ao sistema teve seu início somente em janeiro de 2012, que é o período que marca o início do projeto a ser discutido nesse trabalho.

1.1: Objetivo do trabalho

Há de se considerar que o cliente de acesso ao sistema de comunicação unificada não consegue sozinho integrar os dispositivos de um usuário. Isso porque a solução de comunicação unificada é um sistema que vai além de um software: ela é um conjunto de módulos que trabalham integrados a fim de reunir informações sobre vários recursos e dispositivos de comunicação.

Por esse motivo, pode-se considerar como objetivo geral do trabalho a criação de um cliente que será integrado a uma solução de comunicação unificada, agrupando mídias e ferramentas que proveem comunicação.

Dentro da meta de gerar o cliente de acesso à solução, ainda devem ser ressaltados alguns outros pontos que servirão como etapas para que o objetivo geral seja alcançado:

- Estudo e entendimento das ferramentas que compõem o sistema de comunicação unificada da Intelbras;
- Avaliação e padronização de protocolos de comunicação;
- Desenvolvimento de módulos para servidor de comunicações unificadas;
- Desenvolvimento de elementos externos ao servidor de comunicações unificadas;
- Integração da solução cliente com o servidor de comunicações unificadas e com plataformas de telefonia;
- Testes de viabilidade e usabilidade do cliente desenvolvido.

1.2: Estrutura do trabalho

O presente trabalho será organizado de modo que o leitor primeiramente seja inserido no contexto de comunicações unificadas, entendendo o modo como ela atinge as empresas que a adotam, e como a Intelbras vem buscando se inserir nesse conceito.

Em um segundo momento, será possível compreender de maneira geral o funcionamento do serviço de comunicações unificadas da Intelbras, abordando-se mais a fundo a implementação do cliente de acesso à solução, que é o objetivo de estudo desse trabalho.

Para que essas ideias fossem ordenadas como descrito, o trabalho contará com sete capítulos que abordarão o tema relativo ao projeto.

O capítulo dois trará como objetivo principal evidenciar ao leitor uma melhor abordagem sobre o conceito de comunicações unificadas, apontando problemas

que as empresas enfrentam atualmente para gerenciar seus serviços de comunicação, e descrevendo que melhorias podem ser alcançadas quando se adotam tecnologias que auxiliam a administrar complexos sistemas de comunicação. Um breve histórico sobre o surgimento da filosofia de unificação de comunicações também será mostrado para fundamentar a formalização desse conceito.

Os esforços que a Intelbras vem concentrando para se inserir no ambiente de comunicações unificadas será o tema do terceiro capítulo. Nele serão apresentados os serviços de comunicações que a empresa deseja integrar, bem como o recente sistema de comunicação unificada que está em desenvolvimento, abordando-se a arquitetura e funções gerais da solução.

Os capítulos quatro e cinco tratam especificamente do cliente a ser integrado na solução de comunicações unificadas. Nesses capítulos mostram-se ferramentas utilizadas para a criação do cliente de acesso, definição de protocolos de comunicação e, principalmente, o modo como o cliente atua no sistema de comunicações unificadas, apresentando-se fluxos de funcionamento e aspectos de implementação.

O sexto capítulo mostra alguns testes, resultados e melhorias que foram aplicadas ao cliente implementado. Nesse capítulo são mostrados cenários em que o software desenvolvido foi inserido a fim de verificar sua usabilidade e corrigir possíveis erros.

O trabalho se encerra no capítulo sete com algumas considerações a respeito do assunto discutido, e apresentam-se também perspectivas para novas etapas da solução de comunicações unificadas.

Capítulo 2: Comunicações Unificadas (UC)

O cenário contemporâneo de muitas empresas traz os mais diversos meios de comunicação, os quais são disponibilizados aos seus colaboradores, como telefone, correio de voz, FAX, e-mail, mensagens instantâneas, áudio e videoconferência. Esses geralmente desempenham um bom papel individualmente, no entanto, não oferecem uma experiência completa ao usuário por não possuírem, em grande parte, recursos que promovam uma troca de informações e dados de forma rápida e assertiva[1].

Com o objetivo de suprir algumas ineficácias que esses meios de comunicação possam trazer, os usuários ainda integram seus dispositivos pessoais, como celulares, e-mail pessoal e FAX via internet, adicionando mais complexidade na troca de dados[1].

Aparentemente, pode-se acreditar que uma pessoa com esse volume de recursos disponíveis conseguirá prover e receber informação de forma eficaz. No entanto, geralmente ocorre o oposto nessa situação: são tantos os modos pelos quais uma pessoa pode ser encontrada que se diminui a probabilidade de achá-la em um curto espaço de tempo, assim como é difícil buscar a informação correta armazenada em um dos muitos dispositivos possuídos pelo usuário[1][2].

A Figura 2.1 mostra uma média de tempo gasto por colaboradores de empresas em atividades cotidianas que envolvem comunicação em um dia de trabalho de oito horas.

É possível perceber, por meio da Figura 2.1, que um colaborador toma em média 50% de sua jornada de trabalho se vinculando a tarefas não operacionais, afetando consideravelmente seu tempo produtivo e elevando os custos para a empresa.

Nesse sentido, a UC entra como grande alternativa para resolver problemas de agilidade e praticidade para colaboradores de empresas em trocas de informações diárias, ou seja, a UC pretende fazer com que serviços utilizados para comunicação, os quais antes dispunham de padrões, canais e equipamentos diferentes, sejam integrados em uma única infraestrutura capaz de gerenciar as

informações entre esses dispositivos e oferecê-las ao usuário em qualquer lugar em que ele as acesse[3].

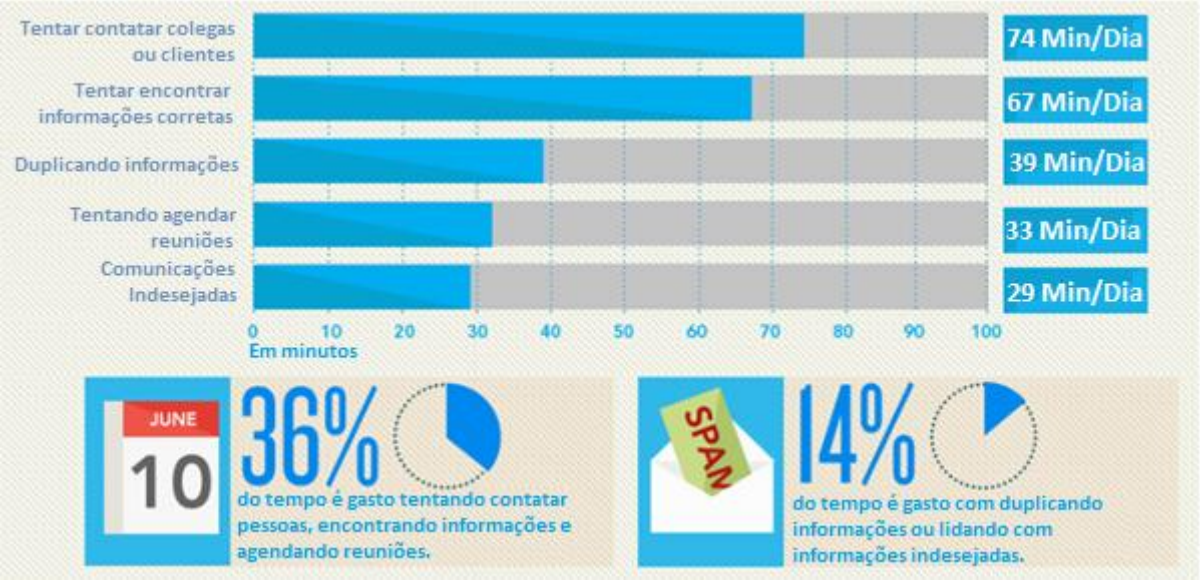

Figura 2.1 - Tempo gasto em atividades relacionadas á comunicação em um dia de trabalho de oito horas[2].

2.1: Evolução das Comunicações Unificadas

A origem da UC está ligada à evolução da tecnologia de suporte. No início dos anos 80 as empresas contavam com sistemas telefônicos concedidos por uma operadora local ou tinham um PABX (Private Automatic Branch eXchange). Esses tipos de sistemas usavam circuitos analógicos ou digitais para executar chamadas do telefone de um escritório central para o cliente. O PABX então aceitava a chamada e fazia o direcionamento correto da mesma[4].

Ainda na mesma época começaram a emergir alguns recursos para comunicação como correio de voz e e-mail, os quais eram adotados por funcionários que necessitavam se deslocar com frequência do local de trabalho[5].

No início da década de 90 surgiu a ideia de Mensagem Unificada (UM), foi nessa época que se realizou uma soma de esforços para unir correio de voz e e-mail em um único telefone celular ou na tela de um computador no escritório de trabalho[5].

Com o advento dos telefones celulares, o recurso de UM foi aprimorado para o que se conhece como sistemas “find me/follow me” (no Brasil recebeu o nome de “siga-me”), os quais ofereciam a possibilidade de um usuário encontrar um cliente no número em que ele estivesse disponível para atender a chamada[6].

Outros tipos de recursos relacionados à comunicação também foram surgindo nessa época na tentativa de promover ao usuário uma interação mais completa entre troca de informações e integração de ambientes. Interfaces controladas por voz também aumentaram sua importância no sentido de propiciar funcionalidades de acesso rápido como realizar chamadas para contatos apenas dizendo o seu nome ou número[6].

Foi nessa década ainda que surgiu pela primeira vez o conceito de UC, com a adjeção entre mensagens e comunicações em tempo real. Esse conceito foi um precursor do que se entende como UC atualmente, embora essas tecnologias citadas não sirvam sozinhas para representar essa ideia hoje[5].

Já no início da década seguinte, muitas empresas desenvolveram outras formas de fornecer comunicação aos usuários. A telefonia IP foi uma das grandes novidades e, assim que foi integrada com UM, mobilidade e desktops, trouxe o advento da UC.

O trunfo da telefonia IP está no fato de o aparelho não ser apenas mais um dispositivo digital ligado a um PABX, ou seja, ele passa a estar conectado na rede como qualquer outro computador, e o áudio é transportado por um codificador/decodificador e não mais por variações de tensão e modulação de frequência[4].

O fato de o telefone estar ligado à rede torna possível oferecer funcionalidades avançadas, ou seja, aplicativos do computador podem se comunicar com o aparelho ou até mesmo se pode atualizar ou instalar programas neste. (5)

No entanto, muitos consumidores que já tinham feito seus investimentos em antigos PABX com correios de voz e e-mail não conseguiram amortizar esse novo jeito de prover comunicação, sendo que a adoção dessas novas tecnologias tem se formalizado de maneira bastante lenta[5].

A tentativa mais recente das companhias fornecedoras de UC é se concentrar no foco de integrar um número maior de ferramentas de comunicação em suas soluções com o objetivo de torná-las mais atrativas aos consumidores, os quais buscam um produto que reduza custos e otimize os processos dentro do meio corporativo.

A Figura 2.2 ilustra de maneira simples a intenção da UC em integrar vários dispositivos liberando o acesso dos dados fornecidos em uma única interface de preferência do usuário. Certamente, a evolução de muitos meios de comunicação auxiliou na formalização do presente conceito de UC.

Figura 2.2 - Integração de vários dispositivos em uma única interface[7].

2.2: Adoção de UC no meio Corporativo

As empresas em geral têm como hábito adotar novas práticas e tecnologias quando as julgam necessárias para otimizar processos ou trazer ganhos para o negócio. Nenhum presidente ou diretor de uma organização adotará aquilo que não se aplica à empresa ou que não traz grande ajuda para impulsionar os lucros.

Entretanto, algumas decisões são tomadas de maneira precipitada, ou não se estima corretamente o que uma nova mudança poderá alterar na realidade da empresa. Como consequências surgem gastos desnecessários, perda de tempo, agravamento de situações adversas em processos já mal formalizados e, em casos mais extremos, demissões e falências.

A opção por um serviço de UC também se enquadra nessa regra: deve-se analisar os benefícios trazidos com a adoção de um novo sistema de comunicação, como também a infraestrutura necessária para colocá-lo no meio corporativo sem que haja impactos negativos aos processos e políticas do negócio.

Boa parte das empresas acredita que um software de UC será capaz de solucionar todas as suas adversidades relacionadas à comunicação. Uma parcela significativa dessas organizações também se depara com um cenário bastante contraditório: após aplicarem um serviço de UC, muitos gestores percebem que a empresa não possuía suporte ou treinamento adequado para o uso de um software bastante poderoso[8].

Muitas das redes atuais têm problemas para sustentar os recursos que a UC oferece, principalmente com relação à videoconferência, que exige grande tráfego de dados. Ainda nesse cenário são inseridos muitos dispositivos, padrões e protocolos, aumentando a complexidade do sistema e a dificuldade de mantê-lo em seu melhor desempenho[9].

A Figura 2.3 ilustra o quão multíplice pode ser uma solução de UC, capaz de oferecer uma série de serviços, o que, por consequência, implica um sistema cada vez mais complexo e, em casos em que não há suporte adequado, difícil de manter.

Figura 2.3 - Serviços que podem ser integrados a um sistema de UC[7].

Com a introdução da UC nas empresas será necessário avaliar continuamente o impacto corporativo de falhas de infraestrutura para se levantar os investimentos cabíveis a fim de manter esse tipo de sistema operando de forma satisfatória. Para tanto, torna-se pertinente a manutenção de uma equipe de TI devidamente capacitada e a adoção de novas soluções para assegurar um projeto, implantação e operação eficiente de sistemas de UC[9].

Por outro lado, quando se conhecem as necessidades de se ter uma equipe capacitada e infraestrutura da empresa, e se trabalha para contornar possíveis problemas, a UC entra como forte aliada para os colaboradores e principalmente para os negócios.

Tendo em vista esses fatores, e sabendo das exigências do mercado, o qual precisa cada vez mais de agilidade na obtenção de resultados em tarefas sempre mais complexas, podem-se estimar pelo menos quatro contundentes motivos para a adoção de UC:

- Acesso remoto ao sistema;
- Colaboração;
- Retorno sobre o investimento;
- Facilidade de uso e agilidade no trabalho.

2.2.1: Acesso Remoto ao Sistema

A flexibilidade de acesso ao sistema é um dos grandes trunfos da UC. Basta que um usuário se conecte à rede da empresa e ele poderá ter acesso a seus dados de qualquer dispositivo que queira, seja de um computador, smartphone ou PDA.

Ainda é possível que uma pessoa consiga, usufruindo do mesmo acesso, contatar outras em diversos meios de comunicação que estas estejam utilizando. Dessa forma, torna-se possível a criação de postos de trabalho remotos sem que haja perdas de tempo e dados[11].

Outro benefício visível do acesso remoto é o fato de um colaborador que esteja viajando ou em outros compromissos que impeçam sua presença física na empresa não ter sua produtividade e a do grupo ao qual pertence afetadas, pois poderá participar de decisões e processos de qualquer lugar desde que tenha um dispositivo conectado aos demais colaboradores[10].

2.2.2: Colaboração

A colaboração entra como uma grande vantagem no sentido de que integrantes de uma determinada empresa, parceiros ou clientes possam participar do desenvolvimento de soluções e produtos juntos, compartilhando dados e arquivos por meio de seus dispositivos de comunicação[11].

A troca de ideias, planos e estratégias de negócio pode acontecer de maneira instantânea e de qualquer local, facilitando a interação entre os usuários e permitindo que os resultados esperados sejam atingidos mais rapidamente[11].

2.2.3: Retorno sobre o investimento

Segundo pesquisa realizada pela revista CIO em 2009, a qual questionou 413 executivos de diversas empresas responsáveis pelo o orçamento de TI sobre a

adoção de UC no meio corporativo, a segunda maior preocupação dos entrevistados quando decidem implantar esse tipo de solução é o retorno sobre o investimento, sendo citado por 57% da amostra[12].

Para que haja uma correta implantação de um serviço de UC é necessário, como citado anteriormente, planejamento, investimento em novas tecnologias e preparação de uma equipe para lidar com o sistema. Por outro lado, o capital empregado retorna nos custos relacionados à gestão e serviços de comunicação como fax, ligações e envio de mensagens, pois todos esses recursos estarão interligados através de uma única rede[10].

2.2.4: Facilidade de uso e agilidade no trabalho

Dentre as vantagens que a UC é capaz de trazer a um grupo ou empresa, o ganho de produtividade pode ser considerado a mais significativa segundo a pesquisa citada no item 2.2.3, a qual afirmou que 79% dos entrevistados assinalaram esse benefício como sendo o mais decisivo para a implantação da tecnologia[12].

Por meio de um sistema de UC é possível conseguir um maior fluxo de trabalho, agilizar tarefas e evitar perda de informações. (5) A empresa pode estar pronta para o cliente a qualquer momento, podendo atendê-lo o mais rápido possível e da forma mais eficiente[1].

Os colaboradores conseguem atender a solicitações de qualquer local, direcionando-as para o dispositivo de sua preferência. O fato de esse tipo de software proporcionar o monitoramento de chamadas e dados por meio de apenas uma interface faz com que o usuário consiga a informação necessária de maneira assertiva, reduzindo o tempo para buscá-la[1].

Eliminação de informações inapropriadas e organização da agenda de compromissos também liberam o funcionário de tarefas indesejadas e permitem que se organize de maneira rápida.

A partir desses recursos se consegue reduzir consideravelmente o tempo gasto por colaboradores em tarefas cotidianas relacionadas à comunicação, como mostrado na Figura 2.1.

Abaixo, na Figura 2.4, pode-se observar uma média desse tempo poupado com a adoção de comunicação unificada.

Figura 2.4 - Economia de tempo médio com adoção de UC[2].

Por meio da imagem acima, é possível perceber que um colaborador consegue economizar cerca de 48% do tempo médio perdido antes da adoção da tecnologia. Dentro dessas quase duas horas salvas, o colaborador pode se dedicar a outras funções dentro da empresa, aumentando seu tempo produtivo.

Além de todos os benefícios citados anteriormente, ainda se pode dizer que a adoção de UC pode trazer outras melhorias tangíveis ou intangíveis, como melhor alocação de funcionários dentro da empresa e satisfação dos colaboradores, os quais não precisam despender seu tempo em tarefas exaustivas e em muitos casos frustrantes.

2.3: Considerações finais

Tendo em vista a necessidade do mercado, o qual necessita cada vez mais de rapidez e agilidade na realização de negócios e o contraponto a esse fator que é o gasto excessivo de tempo em atividades não operacionais, a UC entra como grande aliada aos gestores de empresas na tentativa de elevar a produtividade, flexibilidade e reduzir os custos de maneira eficaz, desde que tomadas as devidas precauções como uma análise correta sobre o sistema.

A evolução do conceito de UC, com a integração de vários recursos em um único software, impulsionou a criação de muitos sistemas dessa espécie. Atualmente, com a telefonia IP, diversos recursos de vídeo, troca de dados e dispositivos móveis, são inúmeras as soluções e serviços que podem ser oferecidos por uma solução de UC.

No entanto, no Brasil, ainda são escassos os esforços na tentativa de colocar no mercado um software de UC comercializável e que possa atender necessidades de pequenas a grandes empresas. Nesse sentido, o próximo capitulo tem por objetivo mostrar a solução adotada na Intelbras com o objetivo de fornecer ao mercado um software confiável e genuinamente brasileiro.

Capítulo 3: Solução de UC Intelbras

No decorrer dos últimos anos a Intelbras tem concentrado forças em ampliar os seus mercados e negócios. A empresa, que tinha por política oferecer aos clientes soluções sem muita flexibilidade e produtos de manufatura, passou a oferecer em seu portfólio serviços customizáveis de comunicação ao ambiente corporativo.

Com essa nova ideia, a empresa tem não só conseguido manter seu antigo mercado de consumidores de varejo, mas também propor a implantação de tecnologias e soluções para companhias de portes variados, mostrando-se como uma das principais empresas de comunicação no cenário nacional.

Nesse contexto, no decorrer dos dois últimos anos, formalizou-se a intenção de colocar no mercado uma solução de UC com tecnologia inteiramente desenvolvida dentro da empresa e capaz de integrar diversos dispositivos de comunicação com plataformas de telefonia.

Como citado no fim do capítulo anterior, ainda são poucas as tentativas no país de oferecer tecnologias de UC genuinamente brasileiras. A Intelbras vem trabalhando com o objetivo de colocar a solução mais completa e competitiva no mercado nacional, hoje monopolizado por algumas empresas multinacionais.

Além do aspecto mercadológico, há ainda a preocupação em utilizar outras tecnologias já existentes dentro da empresa, a qual conta com recursos de telefonia IP e suas próprias plataformas de telefonia. Esse fator proporciona à Intelbras uma vantagem no sentido de implementar uma solução compatível com seus dispositivos, além do fato de poder contar com todo o aparato tecnológico dentro da empresa.

Somado a isso, pode-se dizer que a solução de UC trará à própria empresa uma série de benefícios como os citados na seção 2.2, pois um dos fatores que levaram à implementação de uma solução de UC foi o de inseri-la na Intelbras como auxílio para o problema de integração de muitos dispositivos e melhoria nos processos de comunicação.

3.1: Integração de serviços de comunicação

O objetivo central de uma solução de UC é a convergência entre diversos serviços de comunicação. Entretanto, não há um engessamento no conceito de UC, não sendo possível afirmar quais e quantas tecnologias são necessárias para se concretizar uma solução dessa espécie.

Há os que consideram como UC apenas o gerenciamento de presença. Por outro lado, em alguns casos se trata UC como um software capaz de colocar todos os dispositivos de comunicação do usuário em uma única interface.

A Gartner, maior empresa do mundo em análise e pesquisa na área de tecnologia, define UC como produtos que “melhoram a produtividade de indivíduos, de grupos de trabalho e das organizações, facilitando e habilitando o controle, gerenciamento, integração e uso de múltiplos métodos de comunicação corporativos”[13].

A solução de UC da Intelbras entra em conformidade com o conceito estabelecido pela Gartner, ou seja, a empresa tem por meta colocar no mercado um produto capaz de otimizar a produtividade convergindo não somente dispositivos de comunicação, mas também serviços como[14]:

- Comunicação em tempo real;
- Inserção de múltiplas mídias;
- Integração de múltiplos dispositivos;
- Gerenciamento de presença;
- Independência de localidade.

3.1.1: Comunicação em tempo real

O serviço de comunicação em tempo real pode ser oferecido por pelo menos três vias: áudio, vídeo ou texto. Cabe ao usuário decidir como e em que ocasião deve usá-las caso sejam disponibilizadas.

A solução de UC proposta tem por meta oferecer recursos para cada um dos três meios de comunicação em tempo real. A razão de propor esses tipos de recursos é a de serem grandes aliados no ambiente corporativo no contexto de praticidade e rapidez na comunicação.

Por meio de pelo menos um dos três métodos citados é possível se conectar a qualquer dispositivo e contatar a pessoa com quem se deseja falar desde que ambas as partes estejam conectadas na rede. Na verdade, áudio, vídeo e texto são serviços complementares e devem ser usados para comunicação em tempo real nas ocasiões mais propícias para cada recurso.

- Recursos de áudio: são utilizados geralmente através de telefonia e audioconferência. A primeira é bastante útil no sentido de contatar rapidamente um consumidor ou colega mesmo que este esteja em uma localidade desconhecida. Já a audioconferência mostra-se importante na redução de tempo e custos bem como no aumento da praticidade para se comunicar. É normalmente usada para promover reuniões entre parceiros e gestores todos em conjunto numa mesma conversa. Na solução de UC da Intelbras está prevista a integração desses dois recursos permitindo que o usuário tenha uma experiência bastante completa na utilização de áudio.
- Recursos de vídeo: A solução de UC proposta tem por objetivo integrar-se ao serviço de videoconferência para prover ao usuário o recurso de vídeo. No entanto, esse tipo de tecnologia ainda não estará disponível na primeira versão, apesar de já estarem sendo realizados esforços no sentido de agregá-lo à solução. Serviços de videoconferência são geralmente usados para reuniões rápidas e rotineiras em que os participantes se encontram em lugares diferentes. No meio corporativo muitos colaboradores usufruem dessa tecnologia por permitir que se consiga visualizar dados expostos, fazer explicações demonstrativas ou mesmo por proporcionar um ambiente mais agradável se comparado a conferências apenas com recurso de áudio.
- Recursos de texto: são usualmente utilizados por colaboradores para contatar outros colaboradores. Normalmente esses se encontram em seus postos de trabalho e necessitam tomar uma decisão rápida e/ou rotineira. Para tanto, utilizam de serviços que oferecem a troca de mensagens instantâneas. Os

populares chats têm se mostrado bastante eficientes para solucionar pequenos problemas internos promovendo uma forma de comunicação ágil e rápida. Na solução de UC o serviço de mensagens instantâneas estará presente tanto para troca de informações entre dois usuários conectados ao servidor, quanto para um grupo formado por mais de duas pessoas através de salas de chat, as quais permitem que um grupo de usuários troque mensagens de texto entre si.

3.1.2 Inserção de múltiplas mídias

Como mostrado na seção 3.1.1 é possível que comunicação em tempo real seja oferecida por pelo menos três meios: áudio, vídeo e texto. É a partir deles que se consegue também promover formas de comunicação que não sejam instantâneas. Eles permitem a inserção de outras mídias na solução de UC, contribuindo para que o usuário consiga utilizá-la de maneira mais completa.

Dentre muitas mídias possíveis, a solução de UC da Intelbras permitirá incluir SMS, recursos de áudio, vídeo e e-mail, todos integrados para facilitar a utilização do produto.

As mensagens de texto via celular (SMS) serão úteis para enviar e receber alguma informação escrita e também para o aviso sobre notificações a respeito da conta do usuário, como, por exemplo, ser comunicado sobre a chegada de uma nova mensagem no correio de voz.

Os recursos de voz e vídeo são utilizados novamente agora em serviços como o de mensagem unificada. Esse serviço oferece ao usuário a possibilidade de, por exemplo, escutar mensagens deixadas em sua caixa de recados a qualquer momento, e em qualquer dispositivo que possua recursos de áudio, eliminando a necessidade de discar para seu correio de voz. Além disso, o usuário pode ser notificado em qualquer aparelho sobre a chegada de novas mensagens.

O serviço de mensagem unificada é bastante utilizado em soluções de UC, permitindo que mensagens de texto, áudio e vídeo sejam armazenadas em uma única caixa de recados. Dessa forma o usuário poderá ler, escutar ou ver essas

mensagens de qualquer dispositivo, desde que este ofereça o recurso compatível com o formato da mídia do recado.

O serviço de e-mail ficará de acordo com a preferência do usuário, o qual poderá configurá-lo na solução para recebimento de mensagens unificadas no endereço apontado. Dessa maneira a perda de tempo na procura por informações em diversas caixas de entrada será minimizada.

3.1.3: Integração de múltiplos dispositivos

UC tem como uma de suas principais características a integração entre vários dispositivos de comunicação. Telefones, celulares e computadores podem todos servir como meios para troca de informações.

Na solução de UC da Intelbras, os celulares dos usuários poderão ser integrados permitindo a utilização de várias das mídias oferecidas. Os atuais smartphones e PDAs entram como fortes aliados para mobilidade, visto que o usuário pode usufruir da grande maioria dos serviços. Já os computadores normalmente possuem compatibilidade com a grande maioria das mídias oferecidas no serviço UC, sendo estes importantes meios para a utilização da solução.

Mas um dos grandes diferenciais do serviço UC da Intelbras é o fato de este ter suporte para terminais analógicos, digitais ou IP. Isso acontece pelo fato de a empresa possuir plataformas de telefonia que suportam todos esses recursos.

Dessa maneira, as empresas que adotarem a solução não terão que eliminar seus antigos terminais analógicos para trocar por telefonia IP, por exemplo. E as companhias que desejarem evoluir para plataformas mais avançadas terão em mãos tanto plataformas híbridas (podem suportar telefonia IP, digital e/ou analógica em somente uma plataforma) quanto um serviço UC compatível com elas.

Caberá aos gestores decidirem que tipo de plataforma será usada. Normalmente telefonias analógica e digital são caracterizadas por uma melhor transferência de áudio e ainda são muito utilizadas no meio corporativo.

Já a telefonia IP tem como fator importante a economia, já que se utiliza da rede de dados para realizar a transferência de áudio, permitindo que os usuários não tenham custos com várias ligações, mas somente com o gasto para manter a rede. Outra vantagem é o fato de a telefonia IP permitir integração com serviços de vídeo e e-mail, por exemplo[15].

É comum haver equívocos sobre telefonia IP e VoIP (Voice over IP), no entanto, VoIP faz uso da digitalização e codificação de voz e o empacotamento de dados IP para a transmissão em uma rede, podendo ser utilizada em um computador comum. Ou seja, ela é uma parte da telefonia IP, a qual também se utiliza de terminais telefônicos especiais conectados diretamente a rede de dados permitindo a integração de várias mídias e serviços[16].

3.1.4: Gerenciamento de presença

Um dos objetivos de um serviço UC é fazer com que se diminua o tempo para encontrar clientes e colegas de trabalho. O gerenciamento de presença entra como recurso responsável por minimizar esse problema a ainda auxiliar o usuário no recebimento de dados e informações.

O gerenciamento de presença na solução de UC da Intelbras pode ser entendido como uma adjeção entre disponibilidade para troca de mensagens instantâneas e acessibilidade para receber chamadas em dispositivos ligados à telefonia. Somando-se esses dois modos para gerenciamento de presença, o usuário estará disponibilizando aos seus contatos o que se chama de Rich Presence, ou presença estendida, que nada mais é que um gerenciamento de presença de vários dispositivos e serviços nos quais poderá ser encontrado.

O gerenciamento de presença ligado a mensagens instantâneas permite que o usuário mostre sua disponibilidade para iniciar uma conversa. Ele pode expor também sua situação e localidade no momento. Esse tipo de recurso é bastante utilizado em programas de chat comumente encontrados na Internet e foi muito bem inserido no meio corporativo para indicar disponibilidade e localização dos colaboradores.

No entanto, o serviço de UC será formado também com a integração de terminais telefônicos, e, visto que o usuário pode integrar à solução vários desses dispositivos bem como seus celulares e telefones pessoais, faz-se necessário um gerenciamento da disponibilidade para receber ligações em algum desses aparelhos.

Para isso, serão oferecidas algumas configurações a fim de que o usuário gerencie sua presença ligada a recursos de telefonia:

- Disponibilidade para receber chamadas;
- Configuração de números alternativos;
- Chamada a Múltiplos Dispositivos (CMD);
- Desvio de chamada;
- Não perturbe (DND);
- Mensagem Unificada.

3.1.4.1: Disponibilidade para receber chamadas

Esse recurso será o principal meio de informar o estado de um ramal do usuário na empresa. A partir dele se consegue mostrar aos contatos a acessibilidade do ramal na empresa, podendo essa atingir três modos: disponível, ocupado e indisponível. No primeiro caso o ramal estará livre para receber chamadas. No segundo o usuário estará com uma chamada em curso e na terceira situação não terá nenhum ramal acessível.

Ainda com esse recurso de presença é possível informar aos contatos sobre a disponibilidade de correio de voz e vídeo chamada. Dessa forma o usuário consegue saber se poderá deixar algum recado para um de seus colegas caso este não tenha atendido à ligação e ainda se será possível efetuar chamadas com vídeo quando terminais adequados à mídia estiverem acessíveis.

3.1.4.2: Configuração de números alternativos

Esse recurso de presença permitirá a quem utilizar a solução que se configurem outros números nos quais deseje ser encontrado. Assim, caso não se consiga efetuar uma ligação para o ramal principal configurado, outras formas de contatar o usuário poderão existir de acordo com a quantidade de números alternativos disponíveis.

Esse tipo de configuração é bastante útil no sentido de permitir que se adicionem à solução outros ramais, telefones fixos e celulares, os quais serão contatados de acordo com a prioridade estabelecida pelo usuário[17].

3.1.4.3: Chamada a Múltiplos Dispositivos (CMD)

CMD é um recurso bastante poderoso no contexto de presença. Ele permite que todos os números configurados pelo usuário sejam acionados ao mesmo tempo caso uma chamada seja efetuada. Além disso, quando se atende a ligação em algum dos números acionados, todos os demais dispositivos deixam de notificar o usuário sobre a chamada. Dessa forma, quando algum contato desejar realizar uma ligação, aumentará a probabilidade de sucesso em encontrar a pessoa com quem deseja falar.

3.1.4.4: Desvio de chamada

O desvio de chamada possibilita configurar algum direcionamento para ligações recebidas. Com ele o usuário pode desviar chamadas tanto para seu correio de voz e escutá-las quando puder, quanto para um número configurado nos números alternativos.

Além disso, esse recurso pode ser alterado para desviar sempre as ligações feitas para o ramal do usuário, nunca desviar as ligações, desviar caso o ramal

esteja em uso no momento da ligação, ou ainda desviar caso a ligação não seja atendida após uma quantidade preestabelecida de toques.

3.1.4.5: Não perturbe

Essa configuração para estados dos ramais permite ao usuário informar que não deseja receber ligações. Assim nenhum ramal ou número configurado será acionado.

3.1.4.6: Mensagem Unificada

Como mencionado na seção 3.1.2, o serviço de mensagem unificada tem a função de agrupar todas as mensagens, sejam de texto, voz ou vídeo em uma única caixa de recado. Esse recurso também estará presente na solução de UC e trará grande versatilidade para o gerenciamento de presença relacionado à telefonia, pois permite que o usuário tenha uma caixa de recados única e mais acessível do que os serviços de correio de voz normalmente utilizados.

3.1.5: Independência de localidade

Esse serviço será bastante importante dentro da solução de UC da Intelbras. Ele permitirá o acesso remoto ao sistema por usuários em qualquer lugar que estejam, oferecendo a possibilidade de criação de postos de trabalho remotos e colaboração.

Com o fato de a localização do usuário ser indiferente, ganha-se também em agilidade, rapidez e praticidade de comunicação com colaboradores e clientes.

A independência da localidade surge a partir da arquitetura distribuída da solução de UC de Intelbras. Com os módulos do sistema desacoplados o usuário poderá ter acesso a seus dados e configurações tanto de seu local de trabalho

quanto de qualquer lugar fora da empresa por meio de algum computador, smartphone ou PDA.

3.2: Arquitetura da Solução

A proposta para a arquitetura da solução de UC da Intelbras é a de construir um sistema com o máximo de desacoplamento entre seus módulos. Cada um deles terá um software responsável por executar algumas tarefas mantendo sua parcela de processamento independente dos outros módulos.

Cada parte, no entanto, deverá requerer e repassar dados para outras conforme necessidade. Assim, a informação sobre cada usuário poderá circular dentro do sistema sem sobrecarregar ou inutilizar nenhum dos módulos.

Todos os elementos da solução podem também ser replicados permitindo que ela seja escalável conforme necessidade. Por esse motivo o serviço de UC poderá atender a empresas de diversos portes.

Outras vantagens inerentes a uma arquitetura distribuída também poderão ser vistas no sistema. A transparência para o usuário final em relação ao funcionamento é uma delas. O cliente somente terá a interface de acesso através da qual poderá interagir com toda a solução, sem precisar se preocupar como ela processará e armazenará as informações.

A facilidade de manutenção também pode ser citada como uma característica da solução de UC da Intelbras. Isso acontece por ela ter sua arquitetura dividida em módulos, logo, caso algum destes falhe, o problema ficará isolado e mais fácil de identificar.

É importante ressaltar que o serviço de UC da Intelbras vem sendo desenvolvido nos dois últimos anos. No entanto, nem todos os seus módulos estão prontos ou implementados. A meta é que uma primeira versão da solução de UC seja pré-lançada em outubro de 2012 na maior feira de telecomunicações da América Latina, a Futurecom[18].

A Figura 3.1 mostra um esboço de como será organizada a arquitetura do sistema. É possível ver que contará com cinco módulos: Componente XMPP, LibIntUC, Client-Core, Client-d e Client-UI. Os dois primeiros citados já estão implementados e em fase de integração e testes. O Client-UI está sendo desenvolvido, porém ainda não está concluído. Client-Core e Client-d serão os objetivos de discussão desse trabalho. Cada módulo tem sua responsabilidade sobre o sistema como será discutido a seguir.

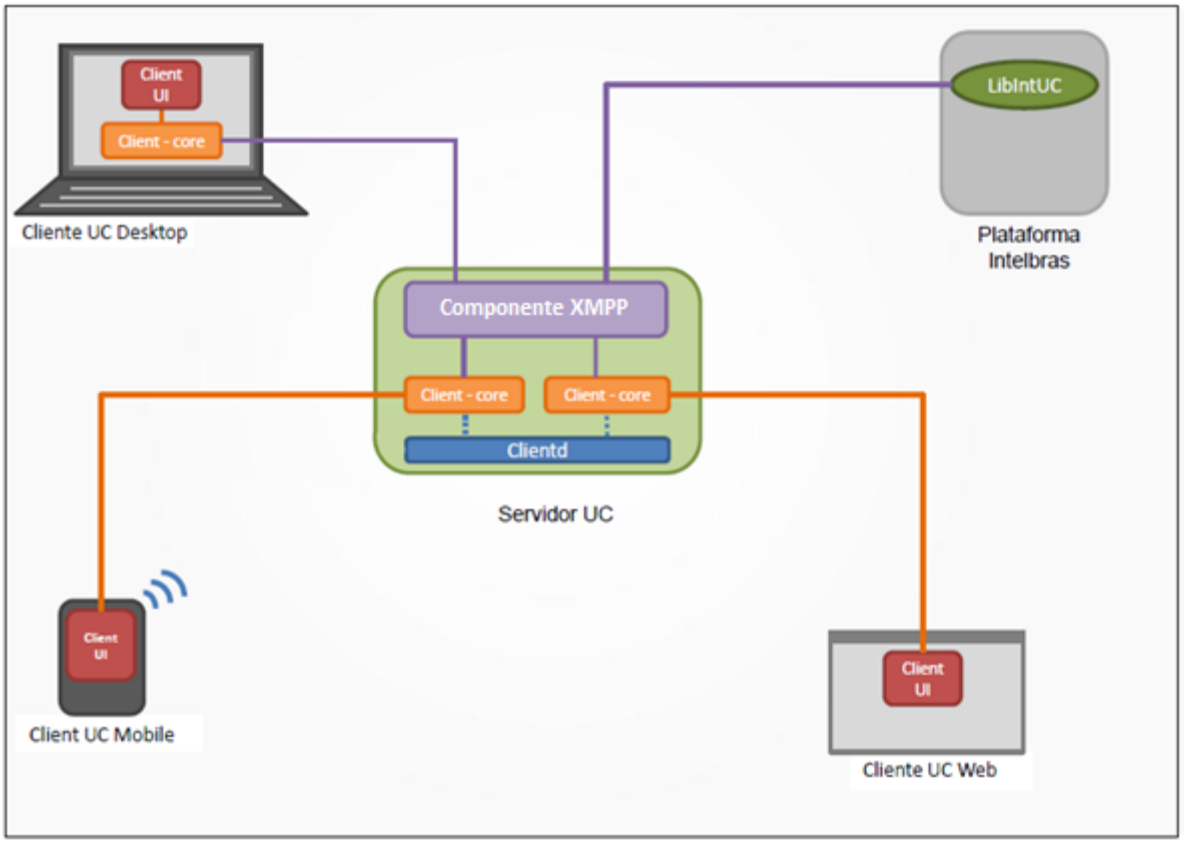

Figura 3.1 - Arquitetura geral da Solução de UC Intelbras[14].

3.2.1: Componente XMPP

Esse módulo será o responsável por trocar mensagens e informações entre os clientes que acessarem o serviço. Assim, é possível que usuários se comuniquem entre si por meio de mensagens instantâneas, por exemplo. Além disso, informações referentes a cada cliente ficarão armazenadas nesse componente para serem disponibilizadas aos clientes.

O componente XMPP é capaz ainda de promover a troca de dados entre as plataformas de telefonia da Intelbras e os clientes. Dessa forma, informações sobre estados dos ramais podem ser oferecidas aos usuários.

No diagrama da Figura 3.1, é possível visualizar que o componente XMPP pode se comunicar com cliente e plataformas a partir de um fluxo de informações representado na imagem pelas linhas contínuas em cor roxa.

Como o próprio nome do módulo diz, ele é feito sob a tecnologia XMPP (Extensible Messaging and Presence Protocol), que é um protocolo aberto, extensível, baseado em XML (Extensible Markup Language) implementado inicialmente para troca de mensagens e presença[4].

O protocolo XMPP será a base de comunicação entre LibIntUC e Client-Core com o componente XMPP, sendo necessário abordá-lo mais a fundo.

3.2.1.1: Surgimento do protocolo XMPP

O protocolo XMPP surgiu em 1998 criado por Jeremie Miller, o qual estava insatisfeito com os serviços de troca de mensagens instantâneas na época e optou por implementar um servidor para troca de mensagens instantâneas com código aberto nomeado de Jabberd. Com a popularização do protocolo a partir da criação de muitos clientes compatíveis com o Jabberd, em 2001 se criou a organização sem fins lucrativos Jabber Software Foundation, a qual prosseguiu no desenvolvimento e documentou o protocolo, inclusive o renomeando para a atual nomenclatura Extensible Messaging and Presence Protocol[19].

3.2.1.2: Arquitetura XMPP

A tecnologia XMPP usa uma arquitetura descentralizada do tipo cliente-servidor, bastante semelhante à utilizada pela World Wide Web e redes de e-mail. O fato de usar esse tipo de arquitetura traz vantagens como facilidade de

gerenciamento, se comparado com uma rede par-a-par, e robustez de ser ter um sistema no qual não existe somente um ponto de falha[20].

Apesar da semelhança com as arquiteturas citadas, é pertinente ressaltar que a arquitetura XMPP possui algumas diferenças em relação a elas. Quando se realiza um acesso a algum Website o navegador se conecta a um servidor Web, entretanto, servidores Web não se conectam entre si para concluir a transação. Já na arquitetura XMPP, quando um cliente tenta uma ligação com um contato que tenha outro domínio, o pacote é enviado para o seu "home server", o qual se encarregará de enviar a mensagem para o destino, podendo passar por outros servidores[20]. A Figura 3.2 ilustra genericamente uma arquitetura XMPP:

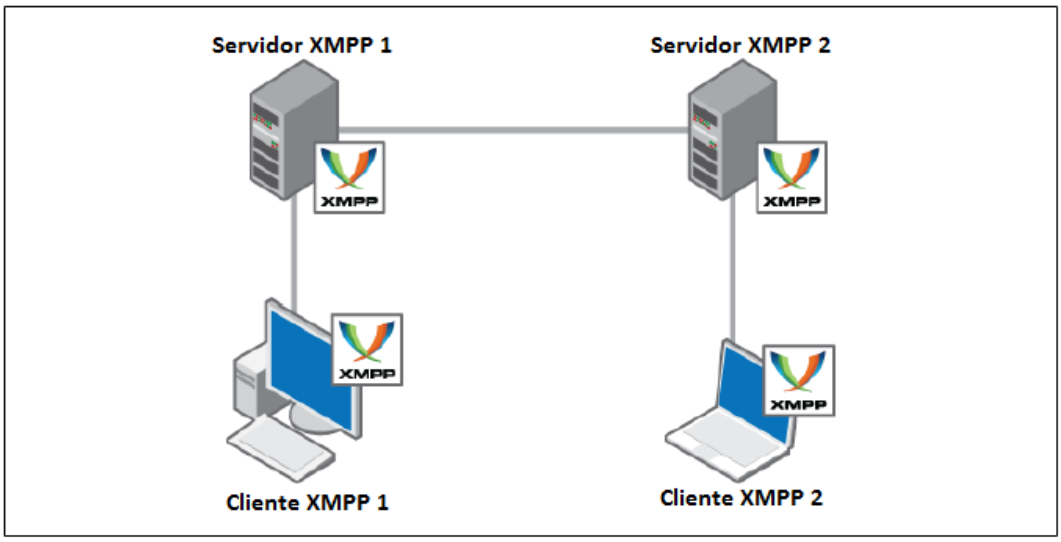

Figura 3.2 - Arquitetura genérica de um serviço XMPP[19][20].

3.2.1.3: Componentes básicos do protocolo XMPP

As mensagens XML trocadas entre clientes e servidores baseados em XMPP são compostas por alguns componentes necessários para identificação e informação dessas mensagens. Esses componentes serão usados durante a implementação da solução UC para que todos os módulos consigam se comunicar.

Os componentes básicos de uma arquitetura UC são JID (Jabber Identifier), Roster e Stanza (Stanzas podem ser de tipo Mensagem, Presença ou IQ), cada qual com sua utilidade e importância dentro do protocolo XMPP:

JID

Todo usuário de um servidor baseado em XMPP necessita ter um JID. Esse elemento é o que identifica unicamente todas as entidades fazendo o seu endereçamento no servidor[21]. É formado por [usuário@]domínio[/recurso], podendo ser dividido em duas categorias: Bare JID e Full JID. O primeiro não contém o elemento [/recurso], já o segundo é composto por todos os identificadores [20]. Esses identificadores têm as seguintes funções[4]:

- Domínio: é o identificador primário do JID. Representa um servidor válido no qual a entidade pode se conectar.
- Usuário: identificador secundário que tem validade somente no servidor do domínio especificado.
- Recurso: identificador terciário. É utilizado para apontar o local ou dispositivo do usuário.

A Tabela 1 mostra exemplos de JIDs em uma rede XMPP:

Tabela 1 - Exemplos de JIDs[4].

JID	Identificador Primário	Identificador Secundário	Identificador Terciário
marcos@casa.com/família	casa	marcos	família
felipe@video.com/locadora	vídeo	felipe	locadora
alice@carro.com	carro	alice	

Roster

Os contatos de um usuário de um servidor baseado em XMPP ficam todos contidos em uma lista chamada de Roster. Ela é armazenada no próprio servidor e permite a organização dos contatos por grupos[21].

Stanza

Em um protocolo implementado sobre o padrão XMPP a Stanza pode ser considerada a unidade básica de comunicação. Podem ser recebidas e enviadas de forma assíncrona, diferenciando-se de uma requisição HTTP, por exemplo, na qual cada resposta só pode ser enviada depois de feita uma requisição. A Stanza é

semelhante a um pacote ou mensagem em outros protocolos[20]. São pequenas porções de XML trocadas entre o servidor e clientes, podendo ser de três tipos:

- **Message:** Pacotes desse tipo são utilizados para troca de mensagens entre usuários, como também para envio de alertas, notificações, conferências e outros tipos de aplicações. Esse tipo de mensagem não recebe confirmação de envio[20]. A Figura 3.3 ilustra uma Stanza do tipo Message.

```
<message from="queen@wonderland.lit"
         to="madhatter@wonderland.lit">
  <body>Off with his head!</body>
</message>
```

Figura 3.3 - Exemplo de Stanza do tipo Message[19].

- **Presence:** Esses pacotes servem para gerenciar e informar a disponibilidade para troca de mensagens instantâneas de usuários. Podem ser úteis também para negociar solicitações de aceitação para outras entidades[20]. A Figura 3.4 mostra uma Stanza do tipo Presence.

```
<presence from="alice@wonderland.lit/pda">
  <show>xa</show>
  <status>down the rabbit hole!</status>
</presence>
```

Figura 3.4 - Exemplo de Stanza tipo Presence[19].

- **IQ:** utilizada para envio e respostas de solicitações em uma comunicação XMPP. Uma requisição IQ pode ser do tipo Set, em que se atribui novas informações a itens de configuração no servidor, ou do tipo Get, na qual há a solicitação para consulta de dados. Cada requisição IQ deverá receber uma resposta de tipo Result, quando bem sucedida, ou Error, quando não se consegue processar a informação solicitada. A Figura 3.5 ilustra as interações entre Stanzas do tipo IQ, enquanto a Figura 3.6 exibe um exemplo de Stanza do tipo IQ[20].

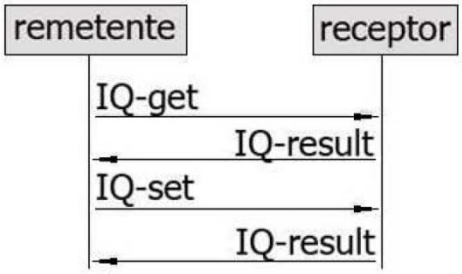

Figura 3.5 - Exemplo de fluxo de troca de Stanzas tipo IQ[21].

```
<iq type="get">
  <query xmlns="jabber:iq:roster"/>
</iq>

<iq type="result">
  <query xmlns="jabber:iq:roster">
    <item jid="alice@wonderland.lit"/>
    <item jid="madhatter@wonderland.lit"/>
    <item jid="whiterabbit@wonderland.lit"/>
  </query>
</iq>
```

Figura 3.6 - Exemplo de Stanza do tipo IQ[19].

3.2.2: LibIntUC

Entre os objetivos da implementação da solução de UC da Intelbras está o de aproveitar a tecnologia já existente dentro da empresa. A integração das plataformas de telefonia ao serviço de UC facilitando o acesso a funcionalidades relacionadas à telefonia já criadas nesses dispositivos é uma dessas metas. Para tanto, é necessário criar elementos que possibilitem inserir as plataformas no sistema de UC.

Com esse objetivo foi implementada a LibIntUC. Esse módulo da solução é o framework responsável por promover a integração entre plataformas e a solução de UC, abstraindo delas as trocas de mensagens necessárias ao sistema[22].

A Figura 3.7 representa como a LibIntUC ficará localizada no sistema. Pode-se notar que o framework será comum a todas as plataformas de telefonia, na imagem, exemplificadas por Impacta, Infinity e CIP92200 (essas são algumas das

plataformas existentes na Intelbras). Isso é feito com o objetivo de manter e respeitar um padrão na transmissão de informações. Certamente, a integração de uma plataforma somente ocorrerá se esta estiver licenciada no servidor UC[22].

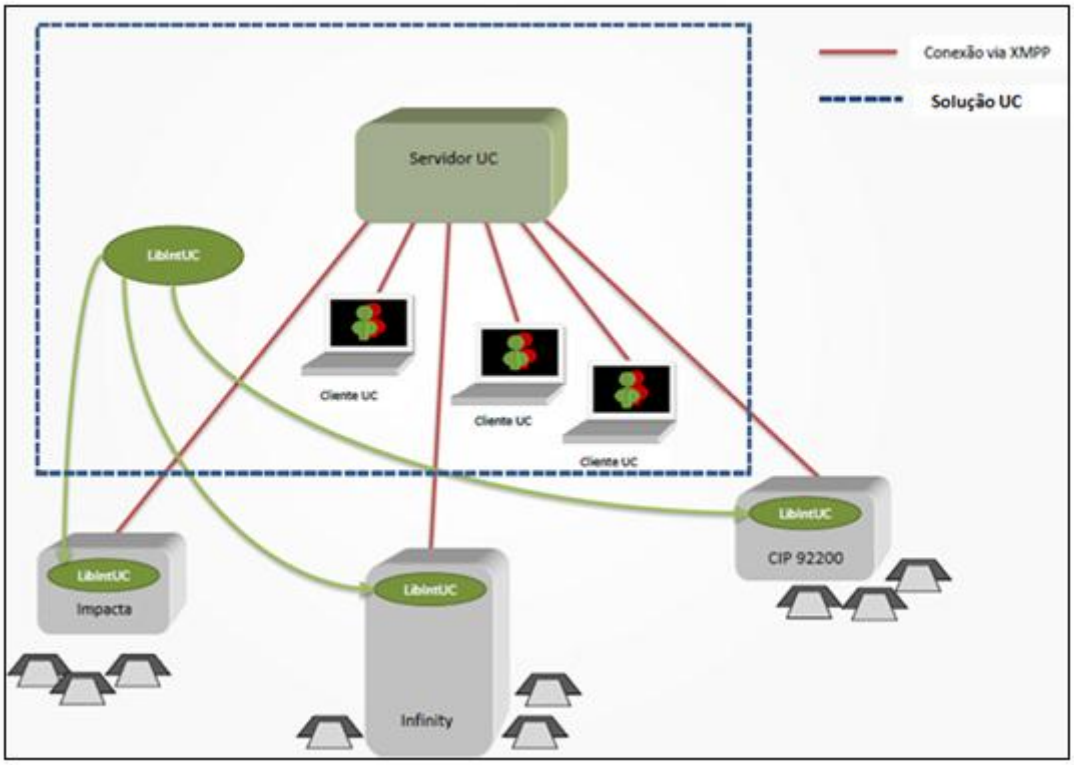

Figura 3.7 - Localização da LibIntUC no sistema[22].

Para funcionar, a LibIntUC deve fornecer informações ao Servidor UC (representado na Figura 3.1 como a união entre Componente XMPP, Client-Core e Client-d e processos de suporte), como também cadastrar call-backs para notificações de ocorrência de alguns eventos[22].

3.2.2.1: Informações enviadas pela plataforma

As informações a serem enviadas para o servidor UC têm como origem as plataformas de telefonia, as quais devem enviar dados sobre[22]:

- Ramais licenciados no sistema de UC;
- Estado dos ramais licenciados (livre, ocupado ou indisponível);

- Disponibilidade para correio de voz;
- Notificações sobre novas mensagens no correio de voz;
- Dados para recebimento de chamadas SIP (Session Initiation Protocol).

Para informar ao servidor UC sobre os ramais inseridos na plataforma, a LibIntUC deve enviar uma Stanza IQ ao Servidor UC contendo esses ramais, que devem ser únicos nesta plataforma[23].

Como toda Stanza IQ deve obter uma resposta, o Servidor UC envia uma nova mensagem à LibIntUC notificando sobre os ramais licenciados nele.

Após receber essa resposta, a LibIntUC enviará ao servidor dados para chamada SIP, disponibilidade para correio de voz e vídeo chamada por meio de um pacote de disponibilidade para receber chamadas como discutido na seção 3.1.4.1[23]. A Figura 3.8 exibe o fluxo descrito acima.

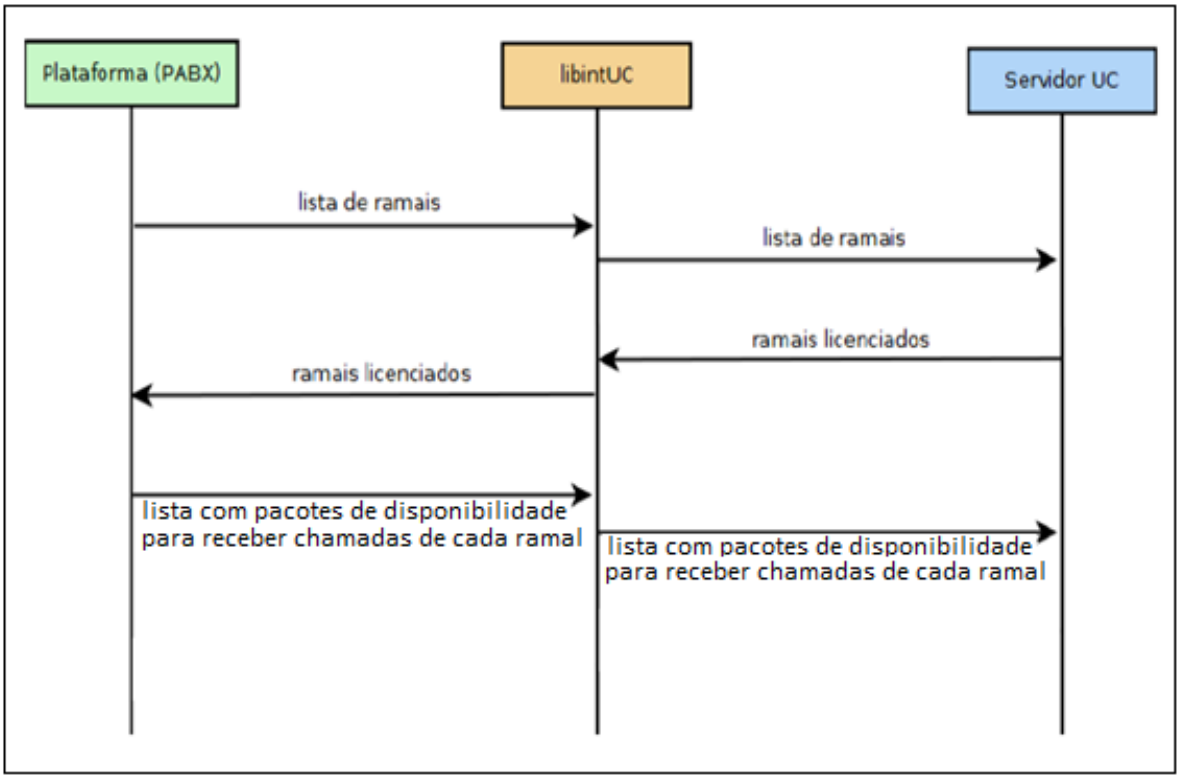

Figura 3.8 – Fluxo de mensagens trocadas para informar ao servidor sobre os ramais licenciado em uma plataforma[23].

Para notificar o Servidor UC sobre mudança no estado dos ramais o fluxo de informações mostrado na Figura 3.9 deve ser seguido[23].

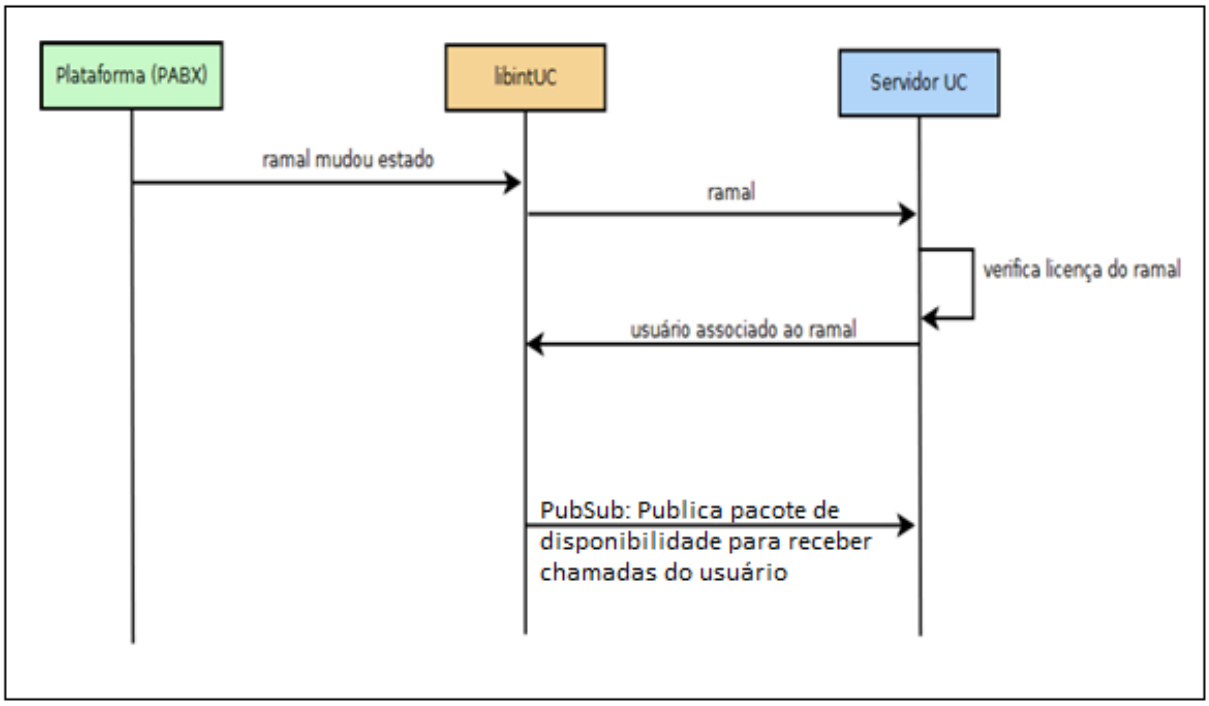

Figura 3.9 - Fluxo de troca de mensagens para aviso sobre mudança nos estados dos ramais[23].

Quando ocorrer a mudança no estado de um ramal, a plataforma informa a alteração à LibIntUC. Essa, por sua vez, requisita ao Servidor UC qual o JID do usuário associado ao ramal. Quando a LibIntUC recebe a resposta, esta publica a alteração no ramal através de um serviço de Publish-Subscribe.

Mecanismos Publish-Subscribe, ou PubSub, como são comumente chamados, permitem que um usuário ou aplicação publique informações (Publishers) que são recebidas por vários outros usuários (Subscribers). O ponto principal desse serviço é a criação dos chamados “nós”, em que os Publishers publicam seus dados e os Subscribers cadastrados nos nós são notificados de algum evento ocorrido[24].

Normalmente esse tipo de mecanismo é mediado por um serviço que recebe as informações publicadas e as transmite aos usuários cadastrados no nó[24].

3.2.2.2: Informações enviadas à plataforma

As informações a serem enviadas à plataforma estão ligadas ao gerenciamento de presença e realização de chamadas:

- Configuração de desvios;
- Configuração de Não Perturbe;
- Configuração de CMD;
- Click-to-call;
- Realização de conferências (áudio/video);
- Envio de SMS;
- Configurações relativas a correio de voz.

A maior parte dessas funcionalidades foi apresentada na seção 3.1.4, fazendo parte das configurações do usuário relativas à disponibilidade dos recursos de telefonia. O serviço de Click-to-Call realiza a ligação entre dois usuários da plataforma e será tratado na seção 5.3.5.2.

Todas as requisições dessas funcionalidades devem partir dos clientes da solução seguindo o fluxo mostrado na Figura 3.10.

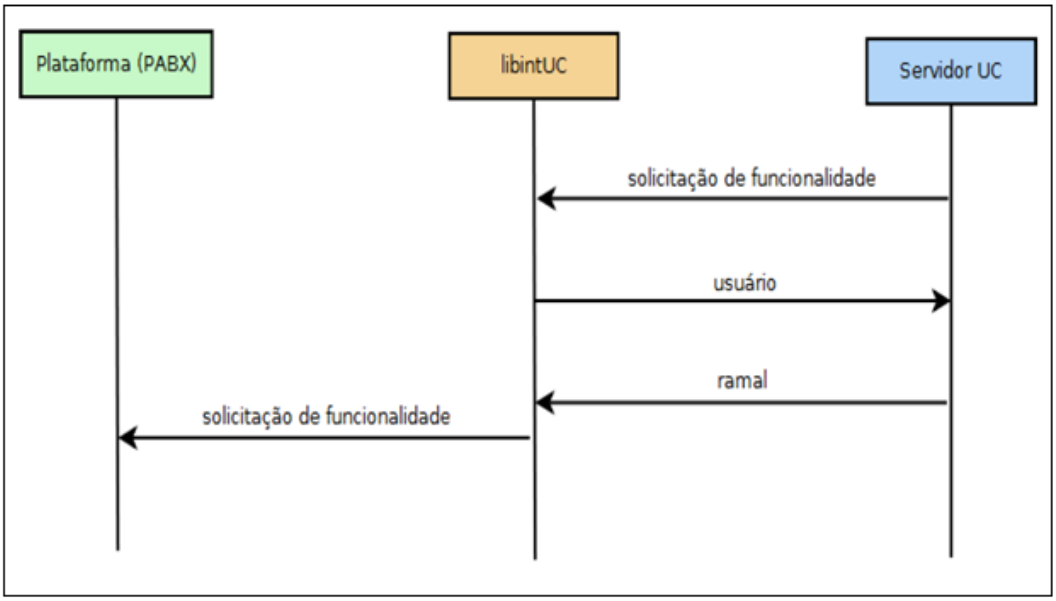

Figura 3.10 - Fluxo de troca de mensagens para solicitações de funcionalidades à plataforma de telefonia[23].

As solicitações vindas dos usuários e recebidas pelo componente XMPP serão repassadas à LibIntUC. Essa por sua vez requisita ao servidor qual o ramal associado ao JID do usuário. Quando a LibIntUC recebe a resposta ela encaminha à plataforma a solicitação de funcionalidade recebida no início do fluxo juntamente com o ramal associado.

3.2.3: Cliente para acesso ao sistema

Todos os módulos do sistema têm sua importância dentro da solução de UC. No entanto, os usuários precisam acessá-los de alguma forma. Com esse propósito é necessário que se crie um meio para interagir com o serviço.

Dessa forma, a solução de UC contará com um conjunto de módulos que serão responsáveis por manter o contato entre os usuários e as funcionalidades do sistema.

Uma das metas da solução, como citado no início da seção 3.2, é criar um sistema distribuído para evitar a centralização do processamento em algum dos módulos. Outra vantagem do serviço de UC é oferecer o acesso ao sistema independentemente da localidade do usuário. Por esse motivo o cliente para acesso ao sistema será dividido em três módulos: Client-UI, Client-Core e Client-d.

3.2.3.1: Client-UI

Esse módulo da solução UC será a interface propriamente dita. É a partir dele que o usuário poderá acessar o sistema e fazer a requisição de alguma funcionalidade. No entanto, o Client-UI não será implementado sob o protocolo XMPP, ou seja, ele sozinho não pode acessar o sistema de UC, precisando de um Client-Core (vide seção 3.2.3.2). Esse módulo servirá somente para que o usuário consiga apontar qual evento será disparado no sistema, não sendo sua responsabilidade se comunicar e processar as informações vindas do Componente XMPP.

Essa característica do Client-UI lhe confere duas vantagens bastante importantes: independência de localidade e distribuição no processamento de informações, indo ao encontro dos objetivos da solução.

A primeira vantagem se verifica pelo fato de o Client-UI ser implementado para três vias de acesso: aplicativo instalado em um computador pessoal, aplicativo para smartphones e PDAs ou por acesso web por meio de um navegador. Dessa forma, mesmo que um colaborador não esteja utilizando seu computador localizado no posto de trabalho, ele poderá utilizar seus aparelhos móveis pessoais para realizar o acesso à solução, ou ainda utilizá-la de qualquer dispositivo que tenha acesso à Internet por meio do Client-UI Web.

A Figura 3.11 mostra a arquitetura do sistema de maneira simplificada, apresentado os três tipos de Client-UI possíveis: Client-UI Desktop, Mobile e Web.

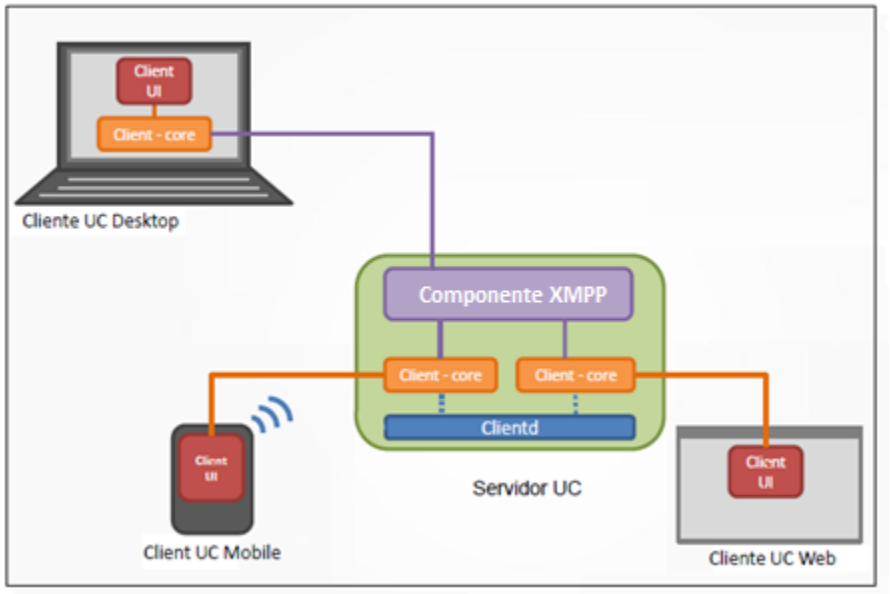

Figura 3.11 - Exemplos de Client-UI. Podem ser do tipo Desktop, Mobile ou Web.

A segunda vantagem citada se verifica no desacoplamento entre Client-UI e Client-Core, ou seja, o processamento de informações trocadas entre o Componente XMPP e o cliente ficará a cargo do Client-Core. No entanto, como se pode observar na figura 3.11, esse módulo poderá estar contido tanto no servidor UC quanto junto ao Client-UI.

Essa característica permite uma melhor distribuição das atividades realizadas pela solução de UC, pois, em dispositivos em que geralmente não há maiores problemas com processamento e alocação de memória como em computadores pessoais, o Client-Core ficará junto do Client-UI, dispensando o Servidor UC de realizar os eventos solicitados.

Já em aplicativos de dispositivos móveis e por acesso via navegador, nos quais se tem limitações de hardware, o Client-Core ficará alocado no Servidor UC, poupando o dispositivo do usuário.

3.2.3.2: Client-Core

O Client-Core, como o próprio nome sugere, é a parte principal do cliente. É ele que tratará da comunicação com o Componente XMPP, fazendo uma ponte entre este e o Client-UI.

O Client-Core será implementado também sob o protocolo XMPP e, por esse motivo, é ele que consegue transmitir todas as ações do usuário ao Componente XMPP.

Esse módulo, juntamente com o Client-d, será o objetivo de discussão desse trabalho, e, por esse motivo, maiores detalhes sobre ferramentas e modo de implementação serão discutidos no capítulo seguinte.

3.2.3.3: Client-d

Como foi falado anteriormente, todo Client-UI necessita de um Client-Core para poder fazer requisições ao Componente XMPP. Entretanto, em aplicações para clientes com acesso web, por exemplo, o Client-Core ficará contido na máquina do Servidor UC.

Não faria sentido manter instâncias do Client-Core previamente prontas no Servidor UC esperando por todos os possíveis clientes, pois seria muito custoso, em

termos de memória e processamento, alocar todos esses elementos criados dentro do Servidor UC sem uma utilidade prévia.

Para resolver esse problema foi criado o Client-d. Esse módulo tem a simples, mas importante função de instanciar novos processos do Client-Core. Toda vez que um usuário fizer uso de um Client-UI Mobile ou Web, os quais não contam com um Client-Core “embutido”, ele fará uma primeira requisição ao Client-d, que criará dentro do Servidor UC uma instância responsável pelo processamento dos dados XMPP.

Como mencionado na seção anterior, esse módulo da solução de UC da Intelbras também será objeto de debate do trabalho e, em razão disso, terá uma abordagem mais minuciosa sobre sua implementação no próximo capítulo.

3.3: Considerações finais

O serviço de UC da Intelbras tende a ser um grande desafio tanto no aspecto mercadológico quanto para desenvolvimento. Isso porque o sistema desenvolvido pode ajudar a quebrar paradigmas dentro da empresa e na venda de soluções de UC dentro do país, atualmente monopolizada por grandes empresas multinacionais.

No contexto de implementação, o projeto também é bastante ambicioso. Manter uma arquitetura distribuída traz grandes benefícios como os citados no início da seção 3.2: processamento distribuído, escalabilidade, transparência, facilidade de manutenção e compartilhamento de recursos.

No entanto, sistemas distribuídos requerem cuidados especiais em suas implementações. Problemas de rede e definição de protocolos de comunicação devem ser minuciosamente contornados a fim de manter um sistema robusto e seguro.

No caso da solução em questão, são cinco módulos diferentes, cada qual com seu próprio software. Todos os cinco programas têm autonomia em seu processamento sem depender dos demais. No entanto, nenhum deles pode operar sozinho se não tiver informações partindo de algum módulo da solução.

Esse interessante paradoxo é o que motiva o desenvolvimento de uma arquitetura na qual haja um compromisso entre interoperabilidade e desempenho individual de seus módulos para garantir um sistema com o melhor funcionamento possível.

Com todo o serviço de UC da Intelbras apresentado, justificando-se sua utilidade, operabilidade e elementos constituintes do projeto, pode-se partir para a discussão dos objetos de estudo desse trabalho: Client-Core e Client-d. Esses dois módulos da solução serão detalhados no capítulo seguinte, mostrando-se aspectos específicos de funcionamento e implementação.

Capítulo 4: Gerenciador de Conexões – Client-d

No capítulo anterior foi apresentada a arquitetura da solução de UC desenvolvida. Um dos pontos mais interessantes dessa estrutura é o fato de ter todos os módulos de sua composição desacoplados, incluindo o cliente de acesso ao serviço.

Não é comum encontrar softwares de clientes que tenham suas interfaces gráficas desvinculadas de seus núcleos de processamento, sendo esse um dos importantes diferenciais da solução de UC da Intelbras. Como mostrado também no capítulo anterior, esse tipo de arquitetura traz ganhos como o de acesso remoto do usuário por uma interface Web ou Mobile, por exemplo.

No entanto, não seria uma boa escolha colocar Client-Cores (núcleos de processamento) previamente alocados no Servidor UC para esperar por possíveis conexões de Clients-UI (interfaces), pois seriam utilizados memória e processamento sem necessidade. Para contornar esse problema, foi necessário criar um módulo capaz de instanciar novos Client-Cores à medida que requisições de conexões ao serviço de UC fossem recebidas.

Antes de apresentar o modo como o Client-d executa suas funcionalidades, se faz necessário levantar as ferramentas de desenvolvimento e recursos para troca de informações como:

- Ambiente de desenvolvimento e execução;
- Linguagem de programação utilizada;
- Troca de dados;
- Protocolo de comunicação.

4.1: Ferramentas de desenvolvimento

O Client-d, dentre os módulos componentes do cliente de acesso (Client-Core, Client-UI e Client-d), é o que necessita ser mais estável. É altamente desejável que os outros dois componentes do cliente também não apresentem

problemas em sua execução, no entanto, caso o Client-d apresente alguma falha ou ocorra um comportamento inesperado, vários usuários serão afetados. Isso acontece pelo fato de o Client-d gerenciar os acessos ao sistema, ou seja, se não houver um módulo que faça a mediação da conexão entre Client-UI e Client-Core, nenhum usuário utilizando Client-Cores remotos poderá ter acesso ao sistema de UC.

Portanto, é recomendável que os recursos utilizados na implementação desse módulo sejam os mais estáveis e rápidos possíveis para garantir o seu melhor desempenho.

Nesse sentido as seções seguintes apresentarão informações a respeito do desenvolvimento do software em questão, sendo apresentados os recursos tecnológicos utilizados na implementação.

Com o objetivo de justificar a utilização desses recursos é importante apresentar também as ferramentas utilizadas e seus benefícios para o projeto.

4.1.1: Ambiente de desenvolvimento e execução

O desenvolvimento do Client-d foi realizado com o objetivo de ser executado em um sistema operacional Linux. A escolha por uma distribuição Linux ocorre por uma série de fatores, sendo os seguintes os mais relevantes para a solução:

- Estabilidade: sistemas baseados em Linux normalmente não apresentam problemas com estabilidade, podendo ficar por um longo tempo sem serem reinicializados. Isso ocorre principalmente pelo fato de ser um sistema operacional de código aberto, contando com a colaboração de diversos desenvolvedores para a liberação de versões cada vez mais robustas.
- Velocidade de processamento: Por gerenciar melhor seus recursos de hardware, kernels baseados em Linux possuem uma capacidade maior de processamento se comparado a sistemas operacionais baseados em Windows.

- Segurança: Apesar de não ser impossível, é bastante raro encontrar vírus em sistemas operacionais desse tipo, o que garante uma maior confiabilidade ao sistema.

4.1.2: Linguagem de programação utilizada

Para o desenvolvimento do código do gerente de conexões foi escolhida a linguagem de programação C. Essa opção foi tomada devido a essa linguagem possuir alguns aspectos importantes em relação aos objetivos da implementação do Client-d, como os seguintes:

- Controle sobre alocação de memória: Esse fator se torna importante no sentido de tornar o código bastante estável. Com o controle sobre alocação e liberação de memória é possível otimizar a utilização desta, o que costuma ser uma tarefa mais complexa em linguagens que não possuem esse recurso explicitamente.
- Velocidade de execução dos programas: C é uma linguagem estruturalmente simples e possui alta portabilidade. O compilador C é capaz de gerar códigos enxutos e mais velozes que outras linguagens.
- Funcionalidade: Embora seja estruturalmente simples, a linguagem C permite a criação de várias rotinas além de contar com uma vasta quantidade de rotinas pré-compiladas que auxiliam no desenvolvimento de códigos.

4.2: Transmissão de informações

No contexto de transmissão de dados entre o Client-d e os três outros módulos com os quais ele se comunica (Client-UI, Client-Core e Componente XMPP), é importante utilizar alternativas que promovam uma troca de dados segura e veloz, a fim de que dados não se percam entre interações, ao mesmo tempo em que sejam rapidamente processados e despachados aos seus destinos.

Com esse propósito, se faz necessário definir os recursos para trocas de dados bem como o protocolo de comunicação adotado nas interações.

4.2.1: Recurso para troca de dados

Os dados fornecidos ao Client-d, bem como as informações enviadas por ele, devem ter um meio de transmissão. Para isso se fez uso de sockets com o objetivo de promover essa interação entre os módulos do cliente de acesso à solução de UC.

Sockets são muito utilizados em ambientes de telecomunicações para transmissão de dados por serem uma maneira bastante eficaz e segura de cumprir essa tarefa. Além disso, diversas aplicações fazem uso de sockets, o que permite que esse tipo de recurso seja bastante flexível. São bastante úteis também no sentido de não promoverem grande tráfego na rede se usados eficientemente.

Apesar de o sistema necessitar de uma transmissão rápida de dados, o modelo do socket implementado foi baseado no protocolo TCP (Transmission Control Protocol) ao invés do UDP (User Datagram Protocol), normalmente utilizado nesses casos. Essa opção foi tomada por se julgar o requisito de garantia de entrega dos dados como fator mais relevante nessa escolha.

4.2.2: Protocolo de comunicação

Para que o Client-d conseguisse se comunicar com os outros módulos foi necessário definir o protocolo de comunicação entre eles.

O processamento das informações, como citado anteriormente, deve ser o mais rápido possível, e, por esse motivo, optou-se por um protocolo bastante simples para cumprir esse critério.

O Client-d deverá receber dados do Client-Core, Client-UI e Componente XMPP de acordo com a Figura 4.1:

[comando] + [|] + [informações complementares]

Figura 4.1 - Sequência para protocolo de comunicação.

Em que [comando] será representado por um número inteiro e indicará qual ação o Client-d deverá executar no processo. O caractere “ | ” tem a função de separador entre o comando e as informações complementares. Essas últimas, por sua vez, têm a utilidade de oferecer dados para a ação requerida no elemento [comando]. A Tabela 2 apresenta alguns exemplos de comandos recebidos pelo Client-d.

Tabela 2 - Exemplos de comandos recebidos pelo Client-d

Protocolo	Comando	Separador	Informações
1|9000	1	|	9000
0|	0	|	-
4|lauvir@server+8000	4	|	lauvir@server+8000

Vale lembrar que o Client-d poderá executar cinco tipos de ações diferentes conforme os comandos recebidos. Essas ações especificam o comportamento do Client-d e serão tratadas na seção 4.5.2.

Alguns dos comandos enviados pelos três módulos que se comunicam com o Client-d têm uma resposta dele. No entanto, como as respostas são síncronas em relação às requisições, elas não precisam de identificadores de comando. Em outras palavras, toda vez que Client-Core, Client-UI ou Componente XMPP enviam um comando ao Client-d, esse somente repassa um array de caracteres contendo as informações que foram requisitadas. A Figura 4.2 ilustra essa situação.

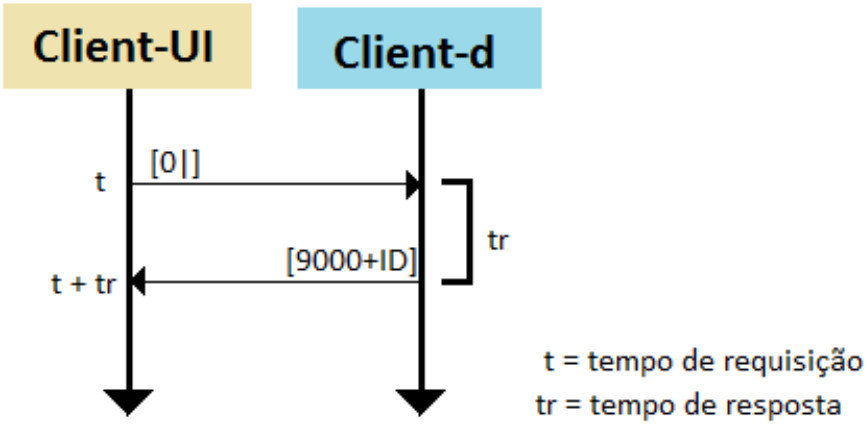

Figura 4.2- Protocolo de requisição e resposta no Client-d

Como se pode observar pela imagem, o comando enviado ao Client-d requisitou informações que foram repassadas em um tempo tr depois do pedido. Assim, a instância do Client-UI somente poderá fazer outro tipo de requisição quando receber essa resposta, impedindo que haja equívocos ou erros na entrega dos dados solicitados.

Com esse comportamento é possível perceber também que a instância do Client-UI fica bloqueada até o recebimento da resposta, o que poderia gerar um comportamento indesejável caso a mensagem não fosse entregue, pois o Client-UI não sairia da condição de bloqueio. No entanto, existe um timeout nesse módulo que o permite saber se uma resposta não foi entregue, deixando que ele faça nova requisição. Esse tipo de recurso também é encontrado no Client-Core e será abordado mais especificamente na seção 4.5.

Por outro lado, nada impede que outras instâncias do Client-UI, Client-Core ou Componente XMPP façam requisições ao Client-d a qualquer tempo, ou seja, mesmo que um módulo tenha enviado um comando ao Client-d em um tempo t, outras instâncias desses módulos podem fazer requisições em um tempo entre t e t+tr. A Figura 4.3 mostra essa situação.

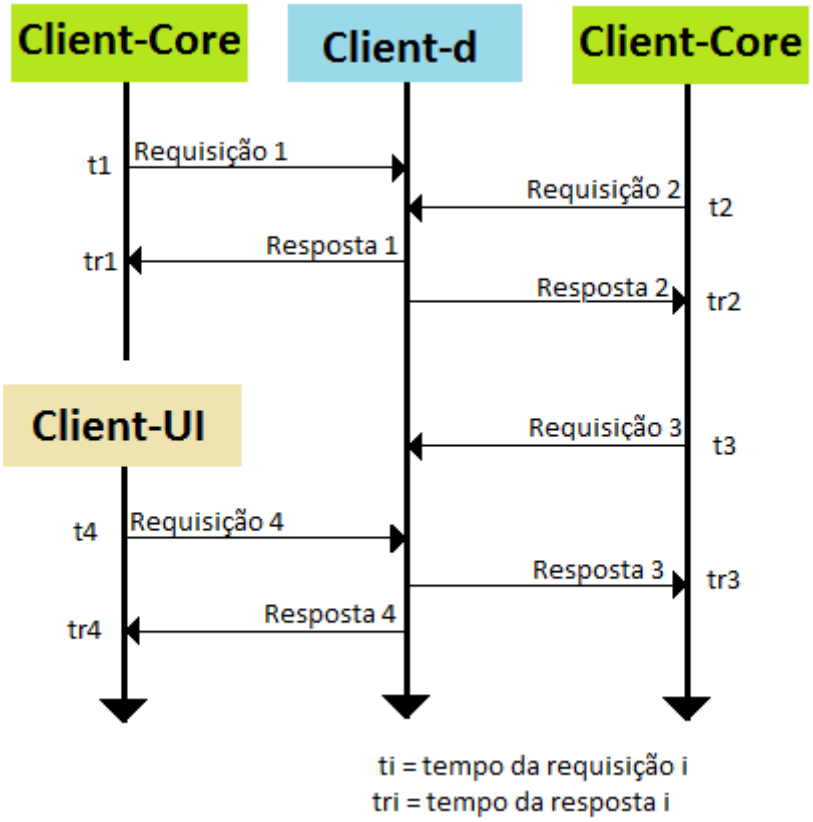

Figura 4.3 - Requisições assíncronas no Client-d.

Como se pode ver na Figura 4.3, uma segunda requisição somente terá sua resposta atendida depois que o Client-d já entregou os dados do pedido anterior. Isso acontece porque o Client-d possui uma “fila” de requisições, na qual a primeira requisição a chegar será a primeira a ser respondida, caracterizando um algoritmo FIFO (First In, First Out).

Com esse recurso do Client-d se consegue receber requisições de maneira assíncrona, além de evitar problemas como o de postergação indefinida, na qual uma requisição jamais receberia resposta caso o processo atendesse primeiro todos os outros pedidos que chegassem entre o tempo de requisição e resposta.

Apresentadas as ferramentas e definidos os métodos para transmissão de dados entre os módulos do cliente de acesso à solução de UC, pode-se partir agora para um detalhamento maior de como o Client-d fica inserido no projeto. Para tanto,

nas seções seguintes serão apresentados aspectos direcionados ao funcionamento desse módulo bem como de sua implementação.

4.3: Dados relevantes ao processo

Para que um Client-UI consiga se ligar a um Client-Core é preciso que algumas etapas sejam cumpridas até que a conexão entre os dois fique realmente estabelecida. Além disso, é necessário monitorar outros aspectos dentro do sistema, como requisições de desconexão e falha de autenticação entre Client-UI e Client-Core.

Entre essas etapas, porém, o Client-d precisa saber algumas informações tanto do Client-UI, quanto do Client-Core, ou ainda do Componente XMPP para conseguir gerenciar e manter um controle sobre as requisições de conexão.

Então, antes de poder apresentar fluxos de funcionamento do Client-d, faz-se necessário saber quais são essas informações que circulam dentro do processo.

São apenas três os tipos de dados que o Client-d precisa ter para poder gerenciar as conexões:

- Porta de conexão;
- ID de Sessão;
- Full JID.

4.3.1: Porta de Conexão

Como será descrito posteriormente, Client-UI e Client-Core também se comunicam por meio de sockets. Entretanto, o primeiro não sabe em qual porta deverá se conectar ao segundo. Quem deverá ditar essa informação é o Client-d. Obviamente, em um serviço baseado em sockets, o estabelecimento de uma conexão deve conter, além da porta, o endereço IP do *server socket*, mas, nessa primeira versão da solução de UC da Intelbras, os Client-Cores deverão ser

instanciados na mesma máquina que o Client-d, ou seja, quando um Client-UI requere a conexão com um Client-Core, ele precisa antes se conectar ao Client-d e, por esse motivo, já sabe qual o IP.

4.3.2: ID de Sessão

Esse parâmetro é uma segurança auxiliar na conexão entre Client-Core e Client-UI. O ID de sessão funciona como se fosse uma senha para garantir que um Client-UI somente consiga se conectar ao Client-Core instanciado para ele. Ele é formado por uma sequência aleatória de dez dígitos seguidos de dez letras para garantir que a probabilidade de uma senha se repetir seja ínfima dando mais segurança contra tentativas de corromper o sistema.

4.3.3: Full JID

Quando uma conexão é efetivada entre Client-Core e Client-UI, o gerenciador de conexões precisa saber que a porta a qual ele cedeu está sendo usada em uma determinada sessão. Para obter essa informação o Client-d vincula o Full JID (user@server/resource) do usuário à porta em questão. Esse artifício é imprescindível para recuperar a porta usada depois que uma sessão é encerrada.

4.4: Fluxos de Funcionamento

Apresentando-se os dados que circularão na troca de mensagens entre o Client-d e os demais elementos que se comunicam com ele, é possível partir para a definição do fluxo de funcionamento, no qual serão mostradas, de maneira genérica, as etapas a serem cumpridas para que se efetive uma conexão entre Client-Core e Client-UI, além de outros comportamentos inerentes ao sistema.

Essas definições terão utilidade para que seja possível compreender o comportamento do Client-d e, posteriormente, os aspectos de implementação desse software.

Como o Client-d deve atender a múltiplas requisições de conexão, ele não pode apresentar um algoritmo muito complexo a ponto de se tornar um gargalo dentro do sistema. Por esse motivo ele tem poucas, mas importantes funcionalidades, as quais podem ser resumidas em:

- Recebimento de pedido de conexão e autenticação entre módulos;
- Notificações de falhas de autenticação;
- Alertas sobre processos encerrados.

4.4.1: Fluxo de conexão e autenticação

Para que Client-UI e Client-Core efetuem uma conexão entre si é necessário que o Client-d intervenha para que ela ocorra, manejando dados e iniciando processos de acordo com o caso.

O fluxo de conexão e autenticação nada mais é do que as etapas as quais os elementos do cliente de acesso devem cumprir até que consigam ficar interconectados. A Figura 4.4 mostra, de modo organizado, como essa série de fases deve acontecer, e quais parâmetros, que foram discutidos na seção 4.3, estarão circulando em cada estágio até que se confirme a conexão.

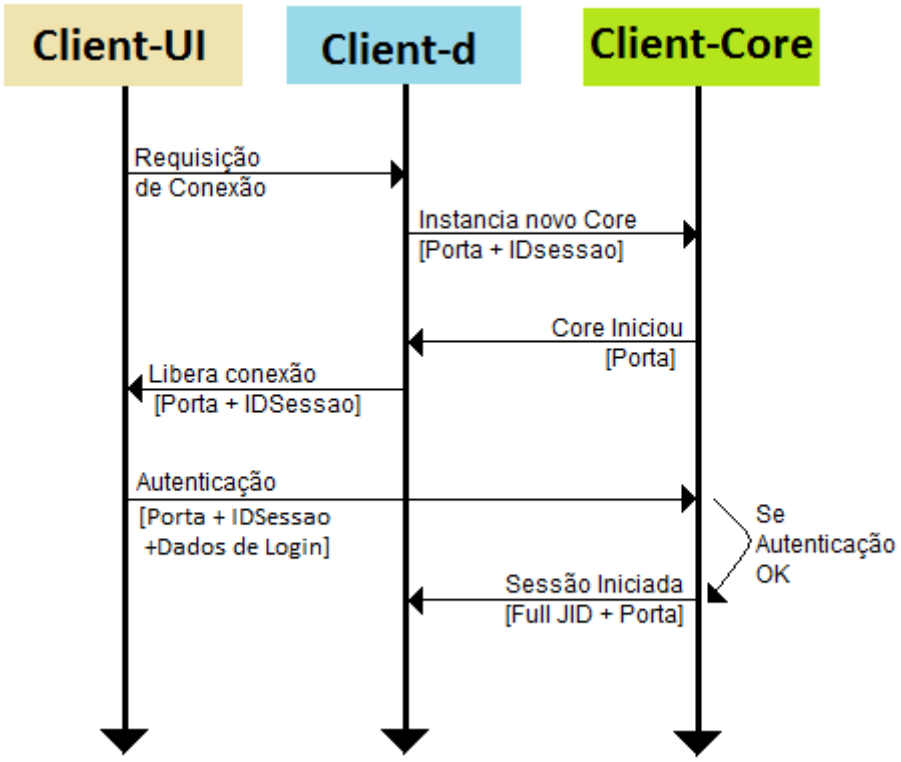

Figura 4.4- Fluxo de conexão e autenticação do cliente de acesso.

4.4.1.1: Requisição de conexão

O fluxo de acesso de um Client-UI ao sistema começa quando um usuário abre uma interface de login do sistema, coloca seus dados de autenticação, e tenta se conectar à solução de UC.

Todas as operações intermediárias até que o usuário consiga visualizar se conseguiu ou não se conectar ao sistema ficam transparentes a ele, ou seja, ao tentar se conectar ao sistema, nenhuma dessas etapas mostradas na Figura 4.4 aparecerá para o usuário, pois ele somente verá o resultado final.

4.4.1.2: Instanciação de um novo Client-Core

Quando o Client-d recebe uma notificação de que alguém deseja se conectar ao sistema, ele parte para a segunda etapa do fluxo, que é a instanciação de um

novo Client-Core. Para tanto, são passados a esse processo recém-criado uma porta e um ID de sessão.

Como mencionado na seção 4.3, a porta servirá para que o Client-Core inicie um *server socket* e espere a conexão do Client-UI. Já o ID será a senha para que esta conexão se efetive.

4.4.1.3: Notificação de início do processo Client-Core

A terceira etapa do processo inicia quando o novo Client-Core envia ao Client-d uma notificação de que já está pronto para receber uma conexão do Client-UI. Nesse estágio do processo é passado o parâmetro [Porta] novamente ao Client-d. Esse procedimento tem a utilidade de identificar para qual Client-UI deverá ser realizada a próxima etapa. No entanto, esse procedimento somente ficará mais claro quando forem discutidos os aspectos de implementação na seção 4.5.

4.4.1.4: Liberação para conexão entre Client-Core e Client-UI

A fase seguinte se realiza quando o Client-d finalmente envia a liberação para que o Client-UI se conecte ao Client-Core. Nesse estágio, são passados os parâmetros [Porta + ID de Sessão], os quais serão necessários para a conexão e autenticação propriamente ditas.

4.4.1.5: Conexão e Autenticação

A quinta etapa do fluxo não tem participação do Client-d. Na verdade, esse estágio fica a cargo de Client-Core e Client-UI, pois serão eles que farão o processo de se conectarem.

O Client-UI tem a função de criar um socket cliente que se se conectará ao *server socket* previamente criado pelo Client-Core. Os parâmetros passados são [ID

de Sessão + Dados de Login], sendo o primeiro o responsável pela autenticação da conexão via socket e o segundo pela conexão ao sistema de UC.

Vale lembrar que os dados de login não foram tratados de forma direta na seção 4.3, isso porque esses dados são formados por um JID, senha e domínio, no entanto, somente a informação do JID será útil ao Client-d.

4.4.1.6: Notificação de sessão iniciada

Quando bem sucedida a etapa anterior, a última fase do processo é um aviso ao Client-d de que Client-Core e Client-UI já estão devidamente conectados e a autenticação ocorreu com sucesso.

Para tanto, a porta e o Full JID do usuário que se conectou são passados ao Client-d para que ele mantenha dentro de sua estrutura informações a respeito de sessões que estão em andamento entre Client-Core e Client-UI. Como já foi citado na seção 4.3.3, isso acontece porque o Client-d vincula a porta usada no processo ao Full JID do usuário, entretanto, novamente essa função somente ficará mais clara quando discutidos os aspectos de implementação na seção 4.5.

O fluxo apresentado nesta sessão é o mais importante dentro do processo, pois é ele que dita o acesso de cliente ao sistema. Entretanto, ainda há os outros comportamentos do sistema que são notificações a respeito de autenticações mal sucedidas e encerramento de sessões.

4.4.2: Falhas de autenticação

Falhas de autenticação são comportamentos mais difíceis de acontecer se comparados a autenticações bem sucedidas. No entanto, ainda assim é necessário tratar esse tipo de ação dentro do Client-d.

Esse comportamento normalmente ocorre quando ou o ID de sessão ou os dados de login inseridos pelo usuário não estão corretos. Nesse caso a conexão entre o Client-Core e Client-UI não será efetivada e uma notificação é enviada pelo

Client-Core ao Client-d confirmando que a operação de autenticação não ocorreu de forma correta. Nesse caso é enviado somente o parâmetro [Porta] para que o Client-d disponibilize novamente a porta para outras sessões.

A Figura 4.5 mostra de forma simplificada essa operação.

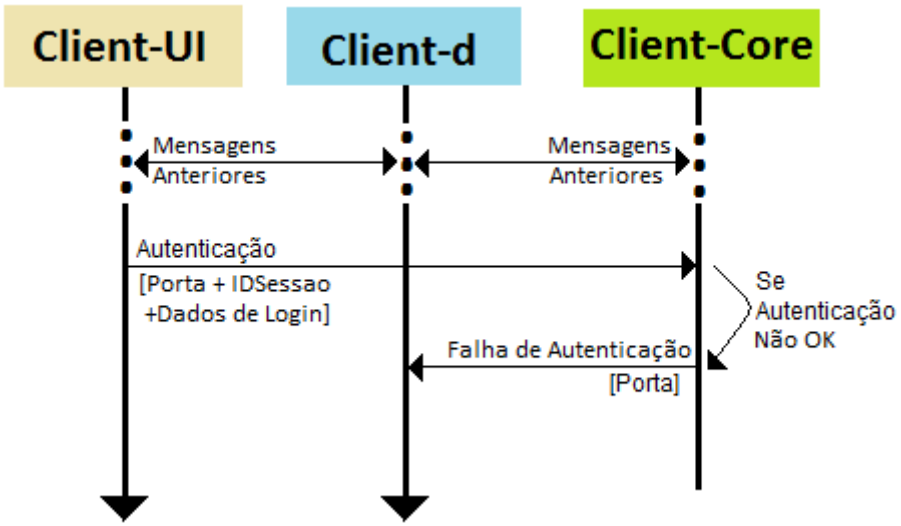

Figura 4.5 - Notificação de falha de autenticação.

Por meio da imagem, pode-se perceber também que, após enviar a notificação de falha de autenticação, a linha de representação do Client-Core se encerra, indicando que esse processo não existe mais. Assim, caso o Client-UI tente novamente se conectar à solução de UC, ele deverá realizar outra vez as etapas de autenticação descritas na seção 4.4.1.

4.4.3: Encerramento de sessões

O último tipo de notificação que o Client-d pode receber é o aviso de que uma sessão entre Client-Core e Client-UI foi encerrada. Nesse momento, quem oferece a informação necessária para o Client-d é o Componente XMPP. Ele passará ao Client-d o comando de encerramento de sessão juntamente com o Full JID do usuário que estava nessa sessão. Dessa forma, o Client-d consegue saber qual porta estava sendo utilizada e a coloca disponível para reuso. Além disso, ele

também desvincula o Full JID da porta e o elimina de seu processo. A Figura 4.6 mostra a notificação de fim de sessão.

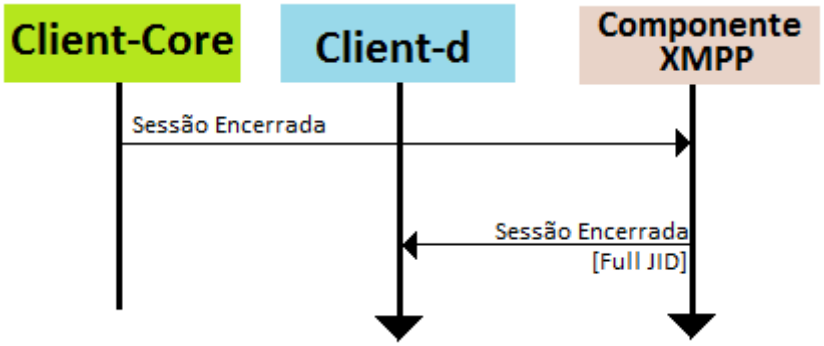

Figura 4.6 - Notificação de encerramento de sessão.

O fato de o Componente XMPP ser o escolhido para enviar essa mensagem é uma questão de segurança na transmissão de informações ao Client-d, isso porque, caso Client-UI ou Client-Core interrompam seus processos de uma forma anormal, existirá outro módulo capaz de passar a informação correta ao Client-d, nesse caso, o Componente XMPP.

É natural que se questione o fato de o Componente XMPP também interromper seu funcionamento, mas, nesse caso, isso seria uma falha mais grave, a qual afetaria todo o sistema de UC.

Outra interrogação pertinente seria a situação em que um usuário tenta se conectar ao sistema duas vezes com o mesmo Full JID, o que geraria um conflito para identificar as sessões. No entanto, o Componente XMPP tem um mecanismo que não permite que um usuário faça isso, ou seja, caso esse fato aconteça, o Componente XMPP enviará primeiro a notificação de encerramento de uma das sessões e, dessa forma, o usuário poderá se conectar ao sistema mais uma vez.

Essa situação é bastante comum quando um usuário se conecta, por exemplo, através de um Web Client-UI abrindo mais de um navegador. A Figura 4.7 identifica esse mecanismo de segurança na solução de UC.

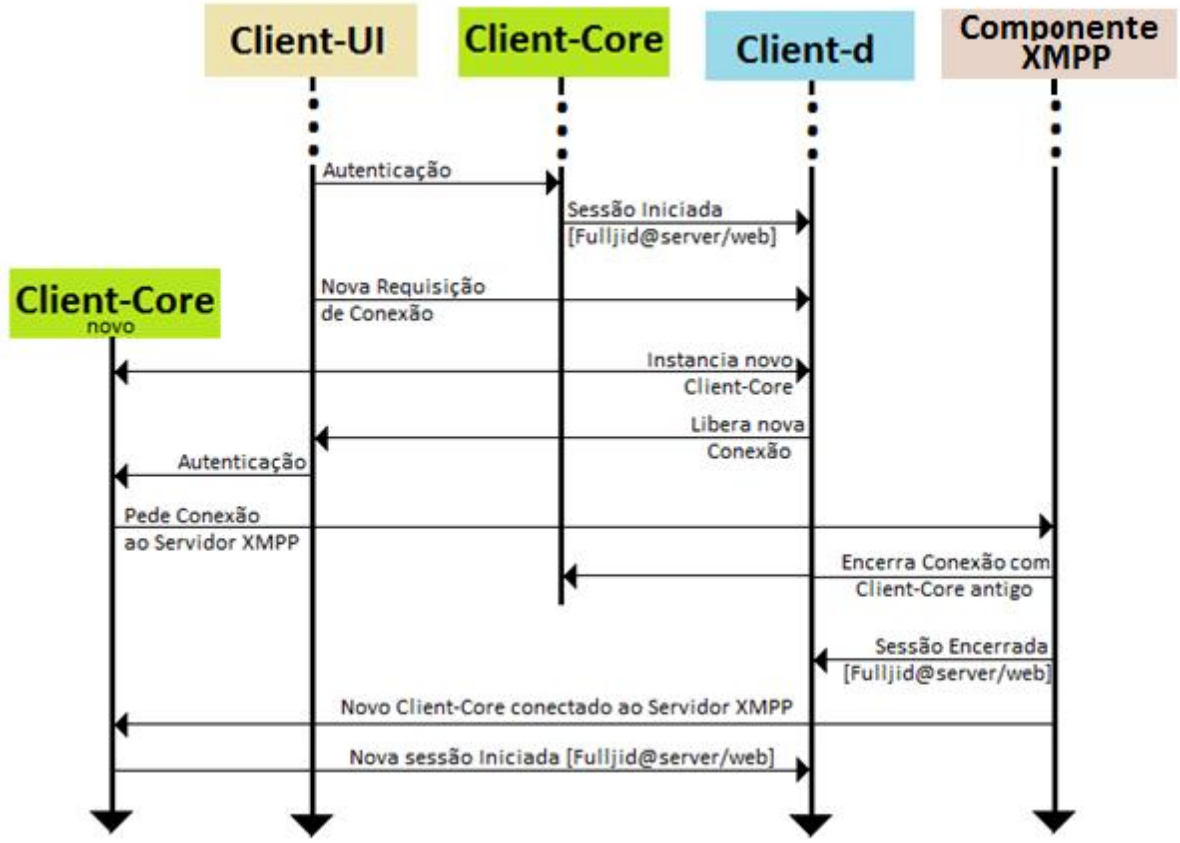

Figura 4.7 - Mecanismo para evitar duplicação de Full JID no Client-d.

Obviamente alguns parâmetros e etapas de inicialização de novas sessões foram suprimidos na imagem para facilitar o entendimento do fluxo. No entanto, é possível verificar que para o Client-d não há a duplicação de nenhum Full JID, já que quando recebe a notificação de encerramento de sessão, o Client-d usa o Full JID para identificar a porta do antigo processo e, posteriormente, elimina esse dado. Assim, quando o novo Client-Core enviar novamente o Full JID, o Client-d já não terá mais armazenada uma informação similar a essa.

4.5: Implementação do gerente de conexões

Nessa parte do trabalho serão discutidos aspectos de implementação do Client-d, na qual se apresentarão como e porque esse software é capaz de realizar os comportamentos descritos em toda a seção 4.4.

Como citado na seção 4.2.1, a comunicação entre o Client-d e os módulos da solução com os quais interage será feita a partir de sockets. Além disso, o Client-d deve ter suporte a várias requisições ou transmissão de informações que podem chegar de maneira assíncrona.

Somado a isso, como foi especificado no protocolo de comunicação na seção 4.2.2, o Client-d terá que executar uma série de ações dentro do próprio software quando receber algum dos cinco comandos possíveis.

Para esclarecer todos esses procedimentos serão discutidos tanto os aspectos de conexão ao Client-d quanto as rotinas que esse possui.

4.5.1 Socket multicliente

Normalmente implementações de sockets são feitas a partir de um modelo cliente-servidor em que o server socket é capaz de atender somente um cliente por vez. Em outras palavras, para que um novo cliente se conecte ao server socket, o primeiro terá que se desconectar e, só então, o segundo efetuará a conexão.

Entretanto, é imprescindível que o Client-d consiga atender múltiplas conexões que chegam de maneira totalmente assíncrona, sendo impossível prever quando uma nova solicitação de conexão será criada.

Para atender esse propósito, o *server socket* do Client-d é capaz de manejar mais de uma requisição de conexão simultaneamente, e o modo como ele cumpre essa tarefa pode ser descrito em dois sub-processos:

- Recebimento de novas conexões;
- Espera por ações de sockets clientes já conectados.

4.5.1.1: Recebimento de novas conexões

Primeiramente, ao ser iniciado, o Client-d cria o socket servidor para atender múltiplas conexões. Após ser criado, esse socket passa a esperar por possíveis sockets clientes, como quaisquer outros modelos de serviços cliente-servidor.

A diferença começa, no entanto, quando ele recebe uma requisição de conexão de um socket cliente. Isso porque, ao receber um pedido desse tipo, o socket servidor do Client-d aceita a conexão, atribui um identificador único ao socket cliente, e o coloca em uma lista na qual estão contidos os sockets que ainda possuem conexão ativa. A Figura 4.8 ilustra esse processo.

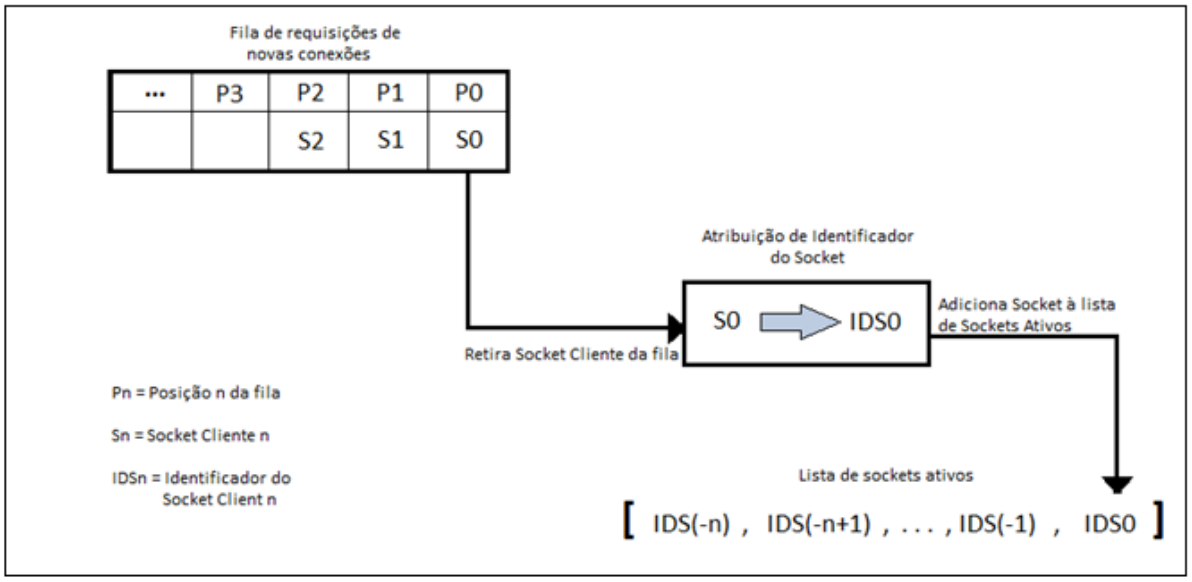

Figura 4.8 - Atendimento de requisições de conexão ao socket servidor do Client-d.

O identificador que é atribuído aos sockets serve como parâmetro auxiliar para que o socket servidor saiba para qual cliente deve enviar respostas de possíveis ações.

É possível perceber também, através da imagem, que o socket servidor possui uma fila de requisições para conexão. Sendo assim, todos os pedidos são atendidos, um de cada vez, sem que se deixe de receber nenhuma das conexões, desde que não atinjam o tamanho máximo da fila, situação em que seriam descartados.

O tamanho da fila, por sua vez, pode ser configurado. Hoje, na situação atual de desenvolvimento da solução de UC, que é uma condição de testes, essa fila de conexões pode atingir até dez requisições, o que é ainda um número plausível, devido a velocidade com que o Client-d atende a novos chamados de sockets clientes.

Naturalmente, em uma situação na qual haja mais usuários, e os acessos à solução sejam mais frequentes, a dimensão da fila precisaria ser reconfigurada para suportar mais pendências de conexões.

Não seria, entretanto, uma boa prática colocar um tamanho não configurável para esse parâmetro, pois poderia haver situações de subutilização ou falta de memória alocada.

Seria possível pensar em uma alocação dinâmica para essa fila de pendências, no entanto, por uma questão de segurança, se optou por uma alocação estática para evitar que múltiplas conexões pudessem congestionar e afetar o desempenho de todo o sistema.

4.5.1.2: Monitoramento de ações de sockets clientes conectados

Nessa situação o socket do Client-d fica esperando por algum evento nos sockets já conectados. Assim, quando algum deles transmite dados ao socket servidor, o Client-d tira o socket cliente da lista, atende sua requisição, e depois o coloca novamente na lista de sockets conectados. A Figura 4.9 apresenta essa ação.

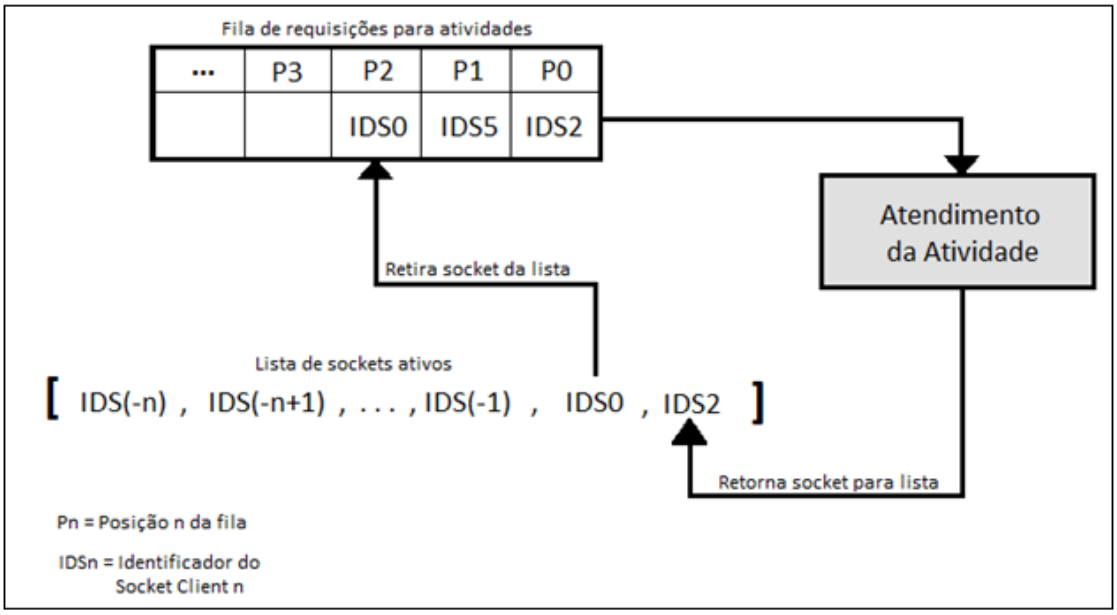

Figura 4.9 – Atendimento de atividades de sockets já conectados.

Também nesses casos é preciso manter um fila de requisições a serem atendidas. Ela é necessária para que, como citado na seção 4.2.2, não haja postergação indefinida, já que todas as requisições serão devidamente atendidas quando atingirem a primeira posição da fila, mesmo que novos pedidos sejam recebidos pelo Client-d.

No caso de a ação do socket cliente ser um pedido de desconexão, o *server socket* faz a mesma ação da Figura 4.9. No entanto, o cliente desconectado não retorna para a lista de sockets ativos.

4.5.2: Comandos enviados ao Client-d

Todas as vezes que um socket cliente que já esteja conectado ao socket servidor faz uma requisição de atividade, o Client-d faz o procedimento descrito na Figura 4.9. No entanto, na mesma imagem é possível ver que o bloco “Atendimento de Atividade” representa a parte do processo na qual o Client-d executa a ação requerida segundo o comando enviado pelos sockets clientes.

Cada um desses comandos que chegam ao Client-d causará a execução de uma ou mais das etapas dos fluxos de funcionamento apresentados na seção 4.4. Além disso, esses pedidos devem obedecer ao protocolo de comunicação adotado e podem, como mencionado na seção 4.2.2, ser de cinco diferentes tipos:

- REQUEST_CONNECTION;
- CORE_ALIVE;
- CONNECTED;
- DISCONNECTED;
- AUTHENTICATION_FAILED.

4.5.2.1: Comando REQUEST_CONNECTION [0\|]

O comando REQUEST_CONNECTION, ou [0|] segundo o protocolo de comunicação, é o pedido de uma nova conexão entre Client-UI e Client-Core. Vale

lembrar que essa requisição somente pode ser enviada pelo Client-UI ao Client-d. No entanto, como descrito na seção 4.4.1, a resposta para a liberação dessa conexão somente acontecerá se outras etapas intermediárias forem cumpridas.

O Client-d, por sua vez, ao receber esse tipo de comando, deve executar uma rotina dentro de sua estrutura de acordo com a Figura 4.10.

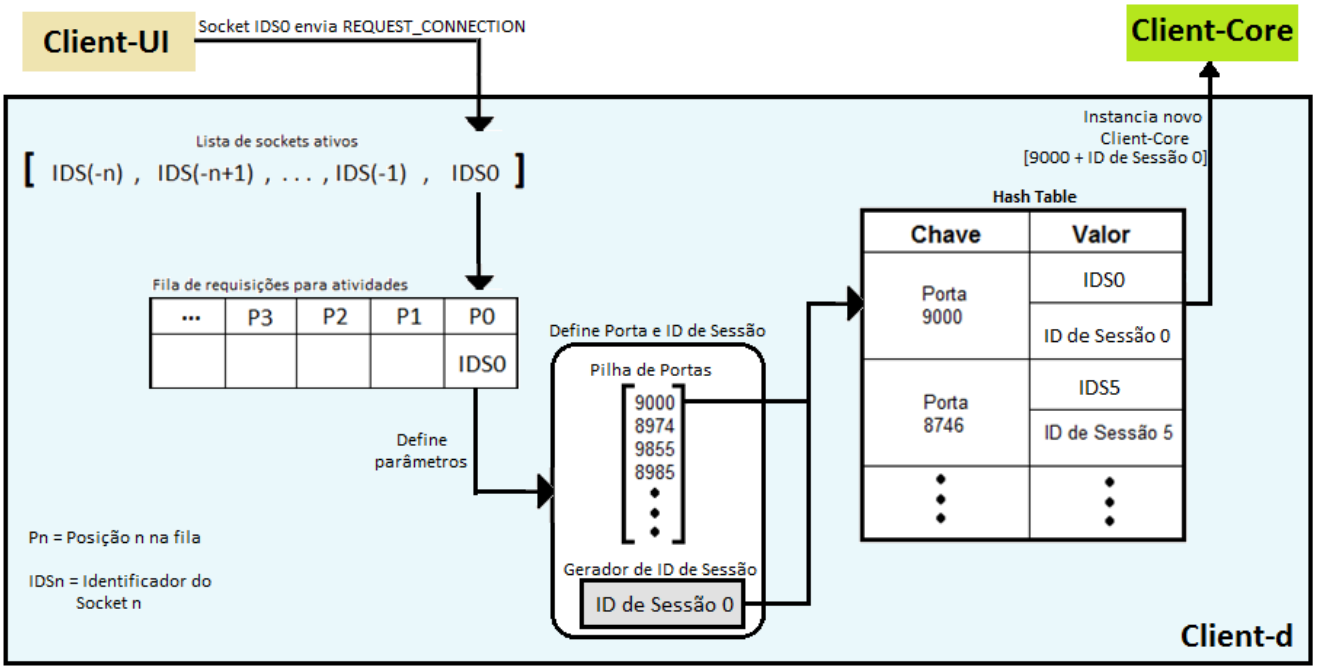

Figura 4.10 - Rotina para comando REQUEST_CONNECTION.

Antes de executar qualquer outra ação, o Client-d faz o procedimento da seção 4.5.1.2: verifica a fila de ações, retira a ação pendente e então parte para a execução da tarefa em si.

Como se pode observar pela imagem, ao receber o comando [0|] o Client-d, o qual mantém em sua estrutura uma pilha contendo todas as portas possíveis de receberem conexões, retira dessa pilha a porta que está no topo, e, em seguida, gera um ID de sessão para a nova conexão entre Client-UI e Client-Core.

Feito isso, a próxima atividade a ser realizada pelo Client-d é o armazenamento de informações, ou seja, o gerente de conexões possui também uma tabela hash na sua estrutura para manter esse tipo de dado. Assim, o Client-d coloca como chave de busca da tabela a porta a qual ele retirou da pilha, e os

valores armazenados são o ID de sessão e o identificador do socket que serão utilizados posteriormente.

A última ação que o Client-d deve executar nesse comando é a instanciação do Client-Core. Para isso, o gerente de conexões possui um mecanismo capaz de instanciar sub-processos (Client-Cores) passando como parâmetros a porta e o ID de sessão gerado.

4.5.2.2: Comando CORE_ALIVE [1\|Porta]

Esse comando é o aviso de que o Client-Core recém-instanciado está pronto para receber a conexão do Client-UI.

O Client-Core passa o comando [1|] além do parâmetro [Porta] para o Client-d. Esse, por sua vez, executa a rotina relativa ao comando segundo a Figura 4.11.

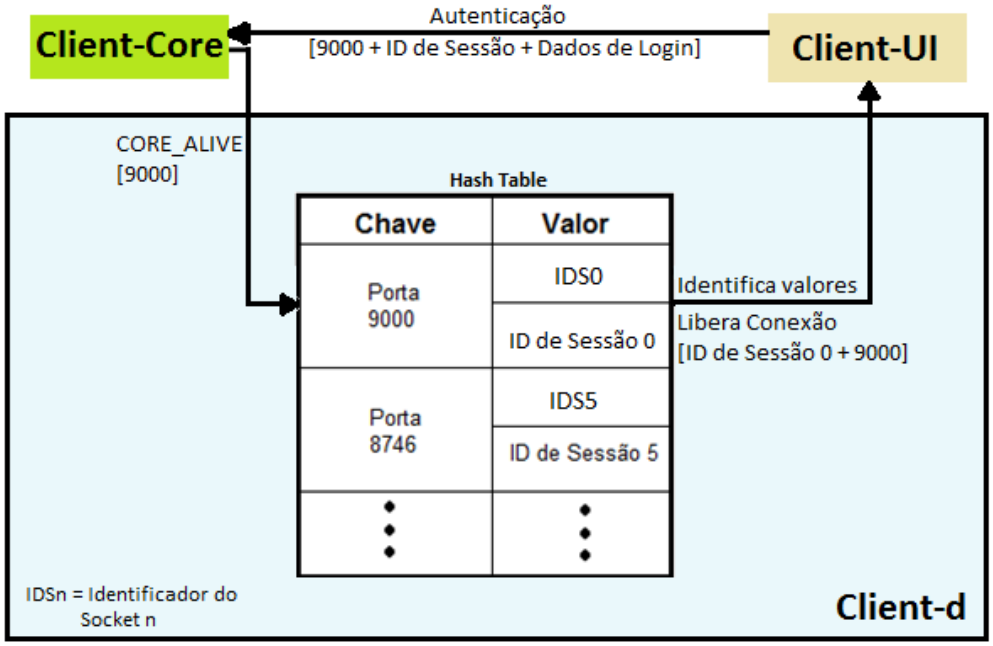

Figura 4.11 - Rotina para comando CORE_ALIVE.

Primeiramente, o Client-d recebe a porta enviada pelo Client-Core. Tendo essa informação, o Client-d consegue fazer uma consulta na tabela hash que

contém as informações sobre o ID de Sessão e o identificador do socket para o qual ele deve liberar a conexão.

Quando o Client-d já possui todas as informações (Porta, ID de Sessão e Identificador do socket) ele libera o Client-UI para se conectar ao Client-Core passando todos os dados necessários.

A última etapa que deve ser cumprida com esse comando foi descrita na seção 4.4.1.5, a qual diz que o Client-UI se conecta via socket ao Client-Core, situação em que o gerente de conexões não tem participação.

4.5.2.3: Comando CONNECTED [2| Full JID + Porta]

O comando CONNECTED também é enviado pelo Client-Core ao Client-d e significa que a conexão entre aquele e o Client-UI foi efetivada com sucesso.

Nesse comando o Client-d deve receber os parâmetros passados e armazená-los em outra tabela hash. A chave nesse caso será o Full JID do usuário e o valor armazenado é a porta em que a sessão entre Client-Core e Client-UI está acontecendo. A Figura 4.12 mostra esse comportamento.

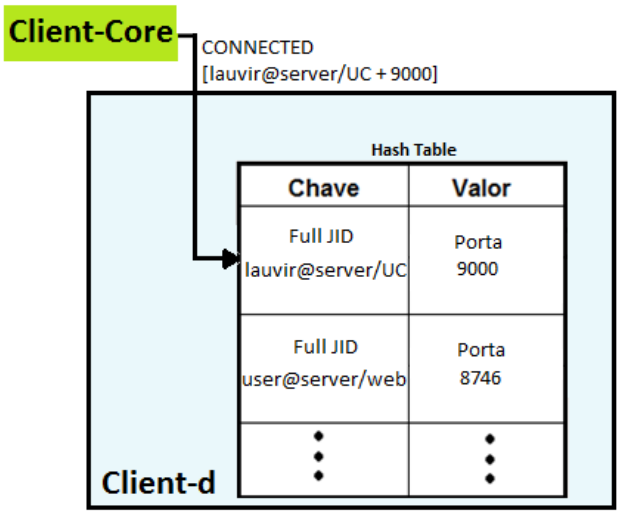

Figura 4.12 - Rotina do comando CONNECTED.

Vinculando o Full JID à porta da sessão, o Client-d consegue manter um controle sobre quais portas estão sendo usadas em sessões que ainda estão acontecendo.

4.5.2.4: Comando DISCONNECTED [4| Full JID]

Nesse comando é o Componente XMPP que avisa ao Client-d que uma sessão foi encerrada, passando como parâmetro o Full JID do usuário que estava usando o serviço de UC.

Quando o Client-d recebe esse comando, ele consulta a tabela hash que contém o Full JID como chave, e “descobre” qual a porta que estava vinculada e ele.

A última etapa desse comando é colocar a porta novamente na pilha para que possa ser reutilizada, completando um ciclo entre o início e o encerramento de uma sessão. A Figura 4.13 ilustra um exemplo do comando de desconexão.

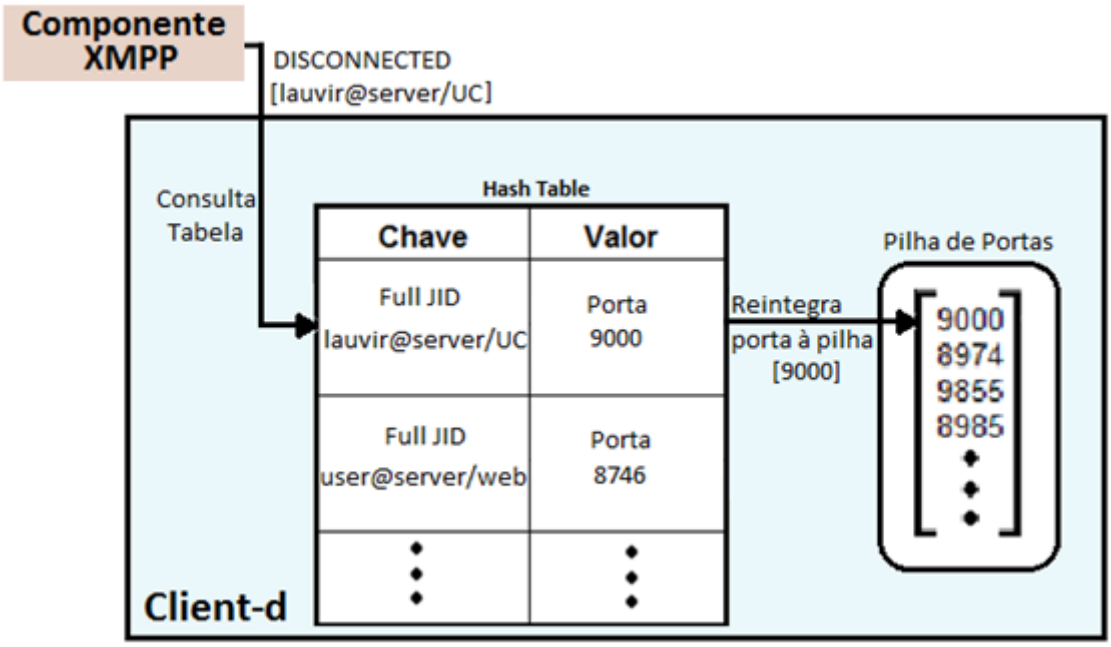

Figura 4.13 - Rotina do comando DISCONNECTED.

4.5.2.5: Comando AUTHENTICATION_FAILED [5| Porta]

Esse comando é, na verdade, um comando de exceção. Se o ID de sessão repassado ao Client-UI não é compatível com o do Client-Core associado, ou se os dados de login inseridos pelo usuário para se conectar ao serviço de UC da Intelbras estiverem incorretos, o comando de falha de autenticação é enviado pelo Client-Core ao Client-d.

Nesse caso, como a sessão ainda não foi iniciada, não há como vincular o Full JID do usuário à porta. E nesse caso nem faria sentido fazer essa associação, pois quando o Client-Core notifica o problema de conexão e passa o parâmetro [Porta], o Client-d simplesmente pega a porta e a reintegra à pilha para reuso.

A Figura 4.14 mostra graficamente todo o Client-d, mostrando todos os comandos e recursos utilizados como descritos nessa seção.

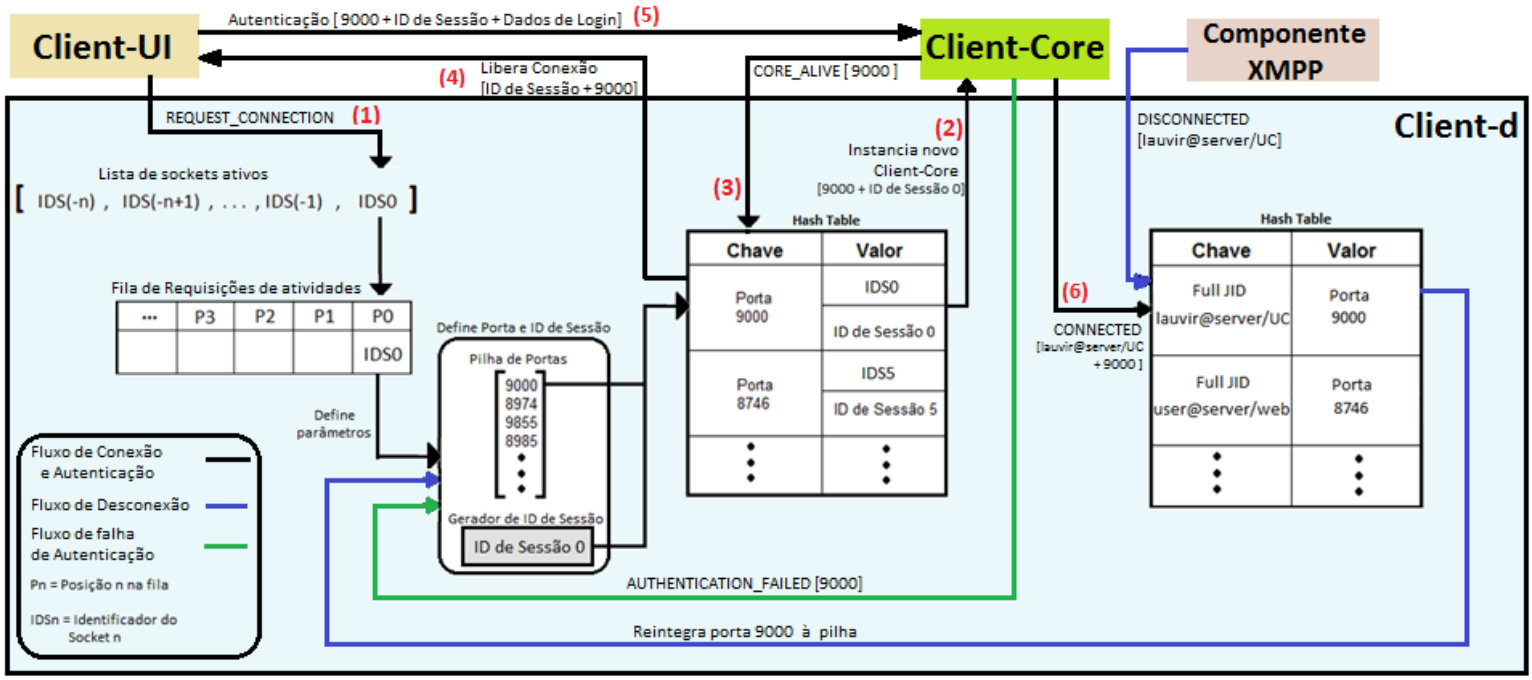

Figura 4.14 - Estrutura completa das rotinas do Client-d.

Os números indicados em algumas das movimentações fazem um paralelo com a Figura 4.4, a qual mostra o principal fluxo de funcionamento do Client-d (fluxo de

conexão e autenticação), mostrando como o gerente de conexões executa todas as etapas em sua estrutura interna.

É possível que se questionem alguns fatores de segurança do processo, como o fato de algum dos módulos do cliente que fizerem requisições de comandos não receberem respostas.

No entanto, existem timeouts contidos tanto no Client-UI quanto no Client-Core que identificam quando uma resposta não recebe retorno. Como exemplo, pode-se citar a situação na qual o Client-UI pede uma requisição de conexão e por algum motivo (seja de rede ou do próprio processo) não recebe a liberação para se ligar ao Client-Core. Nesse caso o timeout inserido no Client-UI apontará que a resposta não foi recebida e notificará o usuário para tentar uma nova conexão.

Com o Client-d definido, pode-se agora apresentar o outro objeto de discussão desse trabalho: o Client-Core. Para isso, no próximo capítulo serão apresentados todos os aspectos relacionados aos requisitos de desenvolvimento do Client-Core, bem como as funções que executa dentro da solução de UC, além de todas as diretrizes de implementação desse módulo.

Capítulo 5: Módulo principal do Cliente – Client-Core

Como vem sendo reforçado nos dois últimos capítulos, a solução de UC da Intelbras tem como uma de suas características principais o fato de ter todos os módulos desacoplados, cada qual com sua responsabilidade sobre o sistema como um todo.

O cliente de acesso ao serviço, no entanto, é o mais segmentado, sendo dividido em três diferentes partes: a interface para o usuário final (Client-UI), o gerenciador de conexões discutido no capítulo anterior (Client-d), e o Client-Core, que será abordado nesta parte do trabalho.

O Client-UI foi apresentado de forma breve na seção 3.2.3.1, a qual menciona o fato de esse módulo da solução ser a ponte entre o usuário e o sistema de UC da Intelbras. É a partir dele que se consegue interagir com o serviço por meio de uma interface gráfica.

No entanto, o que o usuário não consegue perceber visivelmente é que, abaixo dessa interface, existe uma camada nomeada de Client-Core, a qual fará todo o tratamento, envio e recebimento de dados do Componente XMPP. Em outras palavras, o Client-UI não terá que realizar processamento algum dos dados trocados com o Componente XMPP, pois essa tarefa ficará a cargo do Client-Core. A Figura 5.1 tem por objetivo mostrar essa arquitetura.

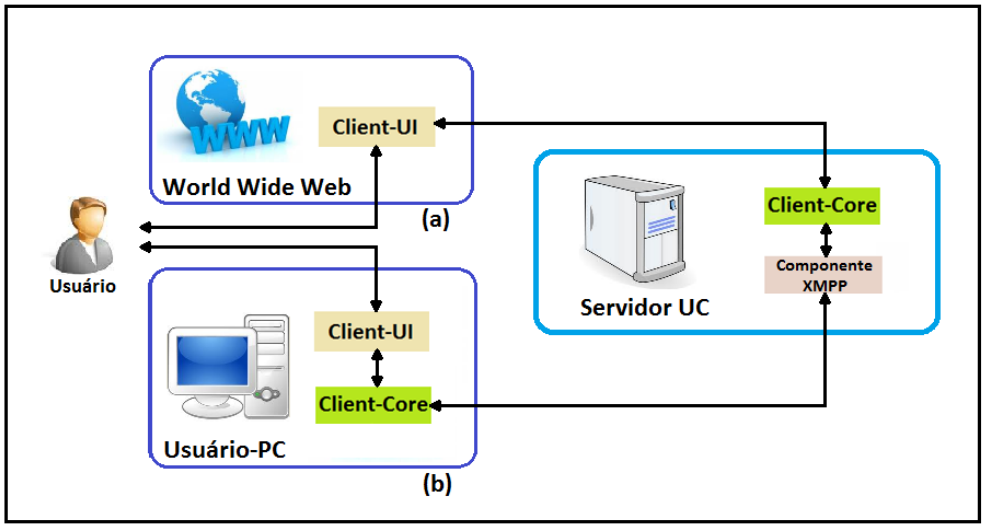

Figura 5.1 - Acesso remoto e local ao Client-Core.

Como pode ser visto na Figura 5.1, toda a troca de dados ficará transparente ao usuário final, mesmo que ele use um Client-Core em seu computador, ou de maneira remota (via web e mobile).

Esse tipo de arquitetura, como citado também na seção 3.2.3.1 traz boas vantagens no sentido de processamento distribuído e independência de localidade. O usuário poderá optar por um acesso de um Client-UI como na situação (a) da Figura 5.1, no qual exonera sua máquina do processamento de informações, deixando esse trabalho para o Servidor UC, ou poderá ter em seu computador o Client-Core instalado, como no caso (b). A vantagem de se ter um Client-Core local é então a maior velocidade na obtenção de dados, pois as informações que sairão do Client-UI não precisarão trafegar na rede para chegar até o Client-Core.

Outro ganho importante dessa arquitetura é o fato de ela permitir que não se repliquem ou criem códigos novos sem necessidade. Isso porque o código do Client-Core pode ser aproveitado do mesmo modo em qualquer tipo de acesso a solução, não sendo preciso, por exemplo, implementar um código para acesso via web e outro para um aplicativo de um computador pessoal. A mudança de código ficaria somente para a interface, a qual poderá ser implementada em várias linguagens de programação e para diferentes modos de acesso.

Por ter esse desacoplamento com o Client-UI, ao mesmo tempo em que é o responsável por tratar e transportar dados partindo tanto do Componente XMPP quanto da interface, o Client-Core pode ser considerado o principal elemento constituinte do cliente de acesso à solução de UC.

Contudo, antes da apresentação de suas funcionalidades e implementação, é pertinente que se identifiquem os recursos e tecnologias utilizadas definindo aspectos como:

- Linguagem utilizada e bibliotecas auxiliares;
- Via para troca de dados;
- Protocolos de comunicação.

5.1: Recursos para desenvolvimento

Como mencionado anteriormente, o Client-Core não estará especificamente localizado em um servidor. Ele poderá também ser utilizado em computadores pessoais, e até mesmo em outra máquina qualquer, diferente das citadas agora.

Por esse motivo, esse módulo da solução de UC deve ser compatível com qualquer tipo de sistema operacional, não havendo preferência por algum específico, como no caso do Client-d, o qual será implementado para executar em um sistema operacional Linux por uma série de fatores como os citados na seção 4.1.1.

Entretanto, há outros aspectos dentro dos recursos e tecnologias utilizados que necessitam ser abordados, como a linguagem de programação e as bibliotecas auxiliares.

5.1.1: Linguagem de programação do Client-Core

O Client-Core deve tratar de muitas atividades dentro da solução de UC. Além disso, ele deve ser capaz de se comunicar com outros três elementos do sistema: Client-UI, Client-d e Componente XMPP.

Para cumprir esses requisitos, foi necessário pensar em uma linguagem de programação bastante versátil e que pudesse contar com bastantes recursos auxiliares.

A opção adotada foi, então, a de implementar o Client-Core em linguagem Java. Essa escolha se deu por Java trazer várias vantagens que são úteis à solução como:

- Orientação a objetos (OO): O fato de ser uma linguagem que segue o paradigma de OO traz alguns benefícios como o de escalabilidade, reutilização de código e mantenabilidade. Essas características são úteis no sentido de permitir uma otimização do código da aplicação, além de reduzirem consideravelmente o tempo de implementação.

- Processamento paralelo: Java tem como recurso bastante importante a capacidade de poder executar várias threads, o que para o Client-Core tem grande utilidade, visto a alta variedade de tarefas que devem ser executadas.
- Portabilidade: uma aplicação em Java pode rodar em vários ambientes sem que seja necessária uma alteração no código. Com isso é possível migrar o Client-Core entre servidores da solução de UC ou mesmo entre os computadores pessoais utilizados pelos usuários.
- Bibliotecas direcionadas a processos específicos: Por ser uma tecnologia livre, há uma vasta quantidade de desenvolvedores que aperfeiçoam continuamente o uso de Java. São muitas as bibliotecas criadas, as quais auxiliam o desenvolvimento de aplicações de diversos tipos. No entanto, há bibliotecas que são úteis para tarefas bastante específicas, como a Smack e JSON library, responsáveis por implementar os protocolos de comunicação adotados no Client-Core.

Essas bibliotecas são de fundamental importância para o Client-Core. Sem elas o trabalho para implementar os protocolos de comunicação adotados seria consideravelmente maior. Para tanto, se faz necessário apresentar também essas ferramentas.

5.1.2: Biblioteca para comunicação via XMPP – Smack

A Smack é uma biblioteca de código aberto, desenvolvida pela companhia de software JIVE Software e direcionada para a comunicação via XMPP. É uma das mais utilizadas para o desenvolvimento de clientes que envolvam esse tipo de protocolo[25].

Por ser open-source e bastante madura em relação ao seu desenvolvimento, essa biblioteca é um recurso muito poderoso para a implementação do Client-Core, no sentido de propiciar uma maior agilidade na implementação. Isso acontece porque ela possui classes e métodos capazes de montar pacotes sobre o Protocolo XMPP, baseado na troca de mensagens XML, como descrito na seção 3.2.1. Em outras palavras, o tratamento dos dados trocados entre o Componente XMPP e o

Client-Core não precisa envolver, em todos os casos, a criação e manipulação direta de arquivos XML, sendo necessário se concentrar somente no tratamento das informações sobre os clientes que acessam a solução.

5.1.3: Biblioteca para comunicação com Client-UI – JSON Library

Como mencionado na seção 3.2.3.1, o Client-UI não é implementado sobre o protocolo XMPP, logo, esse elemento não tem a capacidade de se comunicar diretamente como o Componente XMPP, necessitando utilizar o Client-Core para essa função.

Por esse motivo, foi adotado outro protocolo para a troca de mensagens entre esses dois módulos do cliente da solução de UC, baseado em um padrão chamado JSON (Java Script Object Notation). O motivo para a adoção desse protocolo será discutido na seção 5.2.1. No entanto, um dos fatores importantes é o fato de existir uma biblioteca baseada no formato JSON que entrará como facilitadora no tratamento da comunicação entre Client-Core e Client-UI, facilitando a troca de mensagens entre esses elementos.

5.2: Troca de Dados

A troca de dados entre o Client-Core e os módulos com os quais se comunica é feita também a partir de sockets. Os motivos para a utilização desse recurso são os mesmos citados no capítulo anterior, na seção 4.2.1. Já com relação aos protocolos adotados, faz-se necessário esclarecer alguns pontos, visto o modo como a arquitetura da solução de UC foi implementada.

Como mencionado no início desse capítulo, o Client-Core se comunica com o Componente XMPP, Client-UI e Client-d. Cada um desses módulos é implementado sob a ótica de um protocolo de comunicação.

A troca de mensagens entre o Client-Core e o Client-d foi realizada segundo o protocolo especificado na seção 4.2.2.

Já o padrão para envio e recebimento de mensagens entre o Componente XMPP e o Client-Core é feito a partir do protocolo XMPP, descrito na seção 3.2.1, implementado por meio de troca de mensagens XML entre esses módulos.

O último dos três módulos com os quais o Client-Core troca informações é o Client-UI. O protocolo escolhido para troca de mensagens entre esses elementos foi baseado em JSON, o qual possui uma série de vantagens como descrito a seguir.

5.2.1: O padrão Java Script Object Notation – JSON

O formato JSON é um padrão para troca de dados completamente baseado em texto, sendo um subconjunto da notação de objetos em Java Script. Esse formato vem sendo utilizado por ter características positivas como[26]:

- Independência de linguagem de programação: é um padrão totalmente baseado em texto que usa convenções compatíveis com linguagens bastante utilizadas como C, C++, Python e Java.
- Fácil interpretação: JSON pode ser facilmente lido e escrito pelo programador, facilitando sua implementação. Contudo, as máquinas conseguem interpretá-lo de maneira bastante veloz por ser um formato de estrutura simples, garantido geração e tratamento de pacotes com alta eficiência.
- Baixo *overhead:* Se comparado ao XML, que também é um formato para troca de dados por meio de texto, o JSON permite um alto fluxo de troca de dados sem ocasionar um grande congestionamento na rede, sendo bastante utilizado em situações como a da solução de UC apresentada, na qual há um elevado número de troca de pacotes.

Como mencionado, o padrão possui uma implementação bastante simples, podendo ser formado por apenas duas estruturas: objetos e arrays.

Os objetos são formados sempre por um par nome/valor, no qual o nome será o identificador do valor. A Figura 5.2 ilustra como devem ser formados os objetos em JSON.

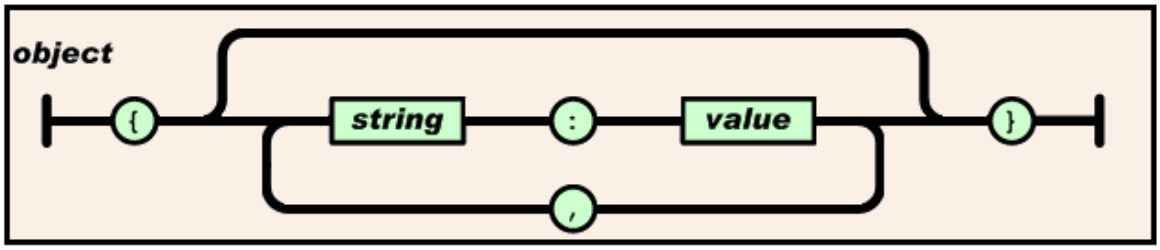

Figura 5.2 - Estrutura de um objeto segundo protocolo JSON[26].

O campo valor pode conter elementos de tipos variados, inclusive outros objetos. Já a estrutura de arrays é formada segundo a Figura 5.3.

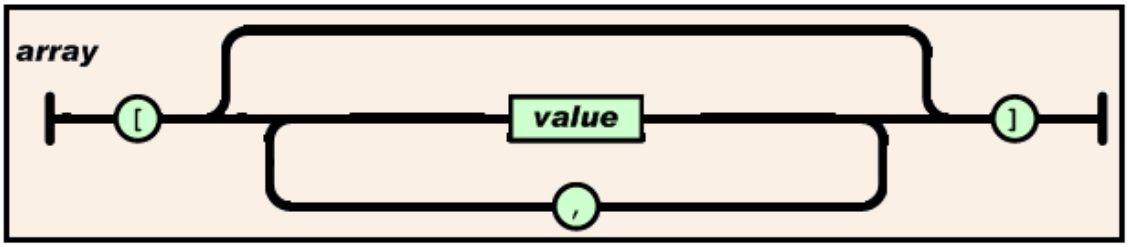

Figura 5.3 - Estrutura de Array segundo o protocolo JSON[26].

Como no caso dos objetos, o campo valor também pode conter vários tipos de elementos, sejam caracteres, números, booleanos, objetos ou mesmo outros arrays.

Tanto JSON, que será utilizado na comunicação entre Client-Core e Client-UI, quanto os formatos utilizados para a troca de dados com os outros módulos deverão estar contidos no Client-Core e possuírem uma ligação bem próxima entre si, pois os dados recebidos de um módulo deverão ser entregues ao seu destino por meio da conversão entre protocolos. Assim, todos os protocolos escolhidos são baseados em texto facilitando o manejo dos dados. Como exemplo dessa situação, pode-se citar as informações enviadas pelo Componente XMPP (por meio do protocolo XMPP) ao Client-Core, as quais deverão ser tratadas e convertidas em pacotes JSON para serem enviadas ao Client-UI. A Figura 5.4 ilustra a arquitetura envolvendo o Client-Core e os demais elementos da solução com os protocolos implementados para cada caso.

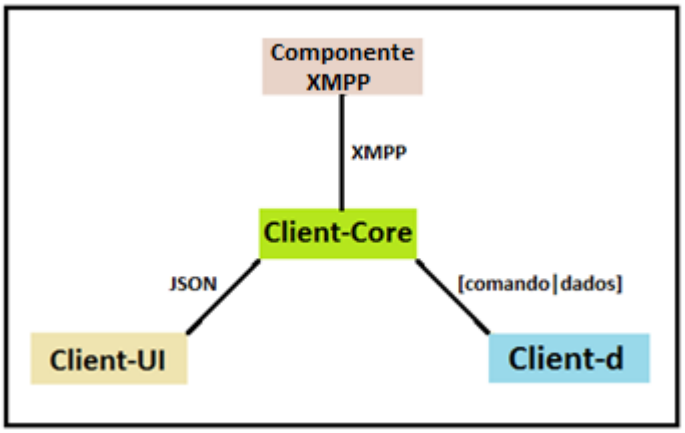

Figura 5.4 - Protocolos implementados no Client-Core.

Definidos todos os recursos utilizados na implementação do Client-Core é possível agora partir para um aprofundamento nas funcionalidades implementadas nesse módulo da solução de UC, apresentando aspectos relacionados à implementação.

5.3: Funcionalidades do Client-Core

O sistema de UC da Intelbras deve oferecer uma série de funcionalidades inerentes a servidores baseados em XMPP, além de vários recursos oriundos das plataformas de telefonia. Todas essas funções serão integradas e repassadas ao usuário final para que ele consiga acessá-las por meio do Client-UI (interface).

Sendo assim, parte dessa união de funcionalidades deve passar pelo Client-Core, pois é ele que colhe as informações disponibilizadas pelos outros elementos da solução, trata os dados, e os repassa aos seus destinos.

As funções que o Client-Core deve ser capaz de realizar foram divididas em cinco categorias:

- Acesso ao sistema;
- Informações iniciais;
- Notificações;

- Configurações de serviços.
- Ações sobre contatos do usuário;

Cada uma dessas categorias será abordada com a finalidade de se apresentar aspectos relacionados à implementação do Client-Core.

5.3.1: Acesso ao sistema

A primeira função que pode ser executada no Client-Core é a de acesso ao sistema. Nessa tarefa o usuário deve fornecer dados (nome, senha e domínio do servidor UC) para a sua autenticação na solução de UC e, então, o Client-Core se encarrega de fazer a validação das informações disponibilizadas com o Componente XMPP.

No entanto, até que a autenticação seja de fato realizada, outras etapas intermediárias que envolvem também o Client-d são necessárias como discutido na seção 4.4.1. A Figura 5.5 mostra como funciona o fluxo de autenticação do usuário no sistema de UC.

A imagem a seguir pode ser entendida como uma extensão da Figura 4.4 apresentada também na seção 4.4.1, isso porque a Figura 5.5 mostra como é feita a autenticação do usuário depois que o Client-d já instanciou o Client-Core e liberou a conexão do Client-UI.

Como pode ser observado ainda na Figura 5.5, depois que o Client-Core recebe o ID de Sessão e os dados de login do Client-UI, a autenticação do usuário no serviço ainda depende de mais duas etapas: verificação do ID de Sessão e autenticação no Componente XMPP.

Na verificação do ID de Sessão o Client-Core recebe como parâmetro do Client-UI um identificador. Dessa forma, o Client-Core compara esse ID de Sessão com a sequência de caracteres obtida assim que foi instanciado pelo Client-d.

Caso os identificadores sejam iguais, a primeira parte da autenticação é bem sucedida e, então, parte-se para a segunda etapa, na qual os dados de login são repassados ao Componente XMPP.

Se o usuário estiver cadastrado no Servidor UC e tiver passado corretamente seus dados de login, então a conexão à solução de UC da Intelbras é efetivada, e uma notificação é enviada ao Client-d e Client-UI por meio do Client-Core.

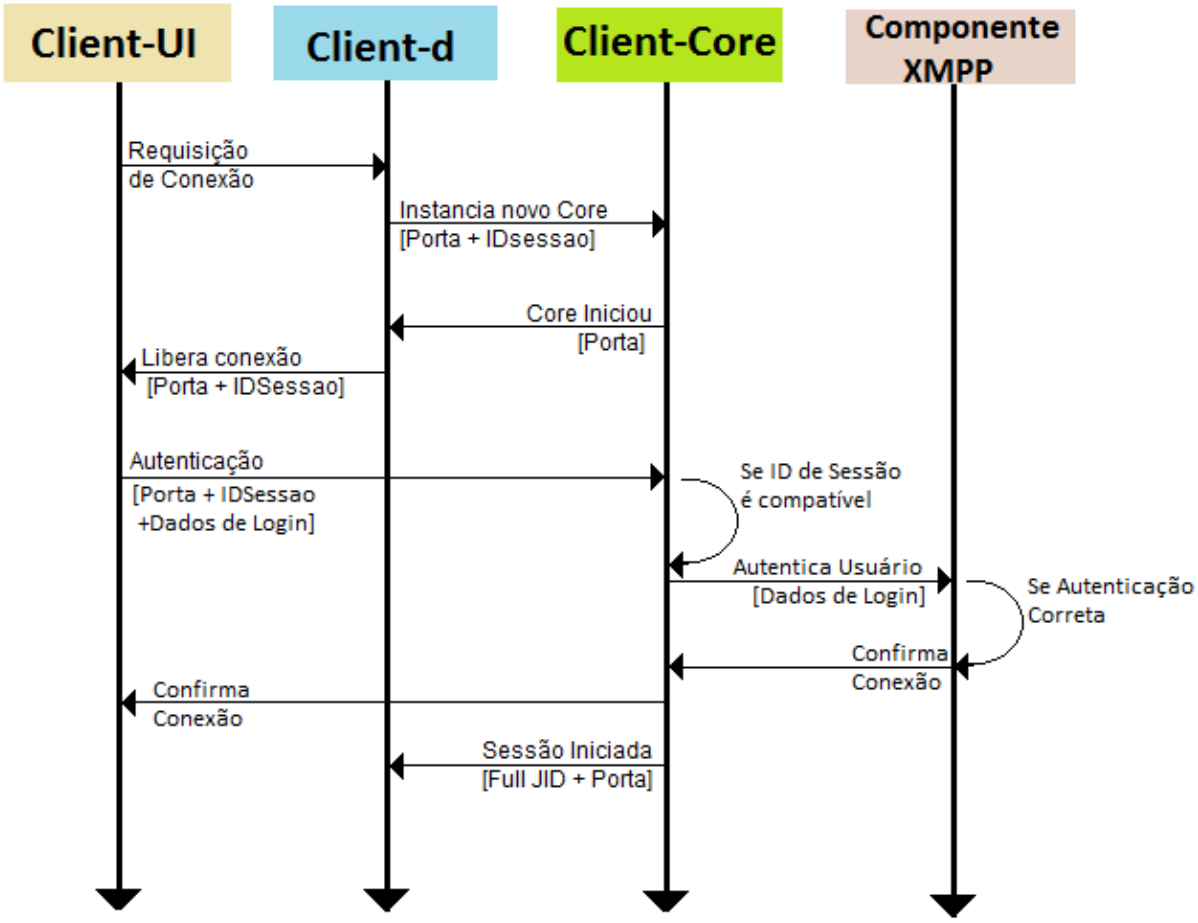

Figura 5.5 - Fluxo de autenticação de usuário no sistema de UC.

No entanto, se os dados passados pelo usuário no momento em que requisitou a conexão ou se o ID de sessão gerado pelo Client-d não forem compatíveis, uma notificação de falha de autenticação é passada pelo Client-Core ao Client-d e Client-UI. Dessa forma, caso o usuário ainda queira se conectar ao sistema, deverá pedir uma nova conexão, refazendo todo o processo da Figura 5.5.

5.3.2: Informações iniciais

Após conseguir se autenticar, o usuário passa a se deparar com a interface principal contendo uma série de informações. Basicamente, essas informações

dizem respeito a alguns serviços que a solução de UC irá oferecer, relacionados à troca de mensagens instantâneas e recursos de telefonia.

O que o usuário não consegue perceber é que, ao efetuar o login no sistema, uma série de mensagens foi trocada entre o Client-UI e o Client-Core para mostrar essas informações ao usuário.

O início do ciclo para a troca de mensagens mencionada se dá quando o Client-Core avisa ao Client-UI que ele está devidamente conectado ao serviço de UC. Isso permite que o Client-UI dispare um pedido ao Client-Core para que ele efetue mais alguma operação, que será respondida logo em seguida.

Esse ciclo se repete até que todas as informações gerais para a interface principal do Client-UI sejam obtidas e mostradas ao usuário. Todo esse processo faz parte da inicialização do sistema e é mostrado na Figura 5.6.

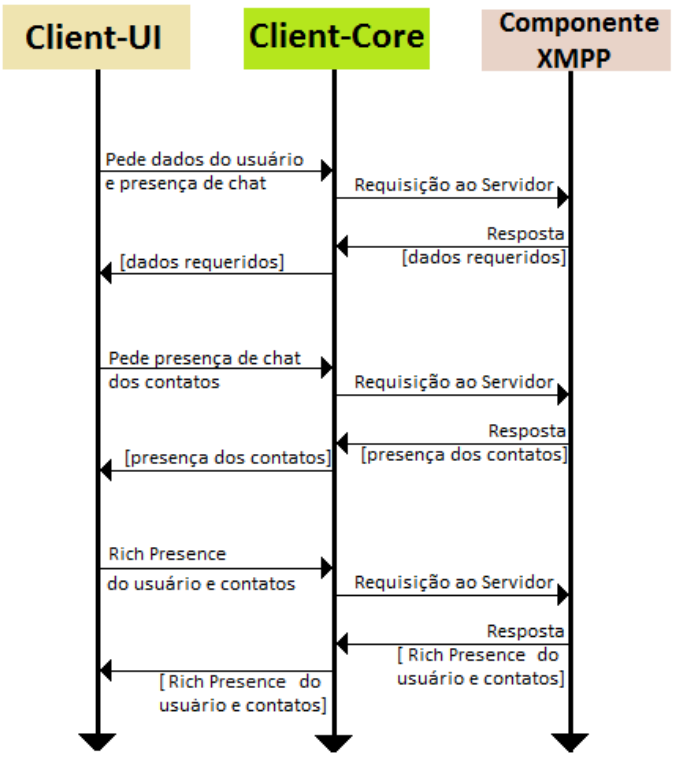

Figura 5.6 - Ciclos de requisições de tarefas iniciais.

Como se pode observar pela imagem, as informações gerais começam a ser requisitadas a partir do pedido de dados do usuário. Nesse ciclo, o Client-Core recebe apenas uma identificação de qual ação deve tomar, ou seja, o Client-UI somente envia ao Client-Core um objeto JSON contendo um identificador de tarefa

e o Client-Core deve então pedir ao Componente XMPP os dados do usuário, que são:

- Nome do usuário;
- JID;
- Presença de chat (Disponibilidade para troca de mensagens instantâneas);
- Imagem do usuário.

A Figura 5.7 mostra um exemplo das mensagens trocadas a partir do protocolo baseado em JSON.

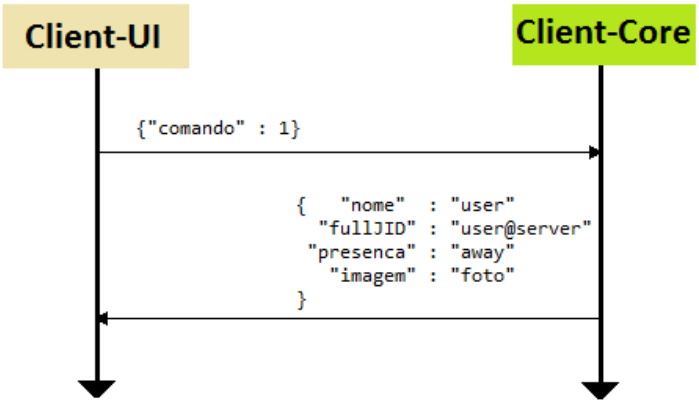

Figura 5.7 - Requisição de informações do usuário.

Na segunda parte dos ciclos de requisição de tarefas iniciais é pedida a lista de contatos com suas disponibilidades para troca de mensagens instantâneas. Basicamente, esse processo é feito de forma semelhante ao apresentado na Figura 5.7; no entanto, a mensagem que o Client-Core envia ao Client-UI é composta por um array JSON contendo objetos com as informações de cada contato.

As últimas informações que o Client-UI precisa saber logo na inicialização do sistema são os dados de disponibilidade para receber chamadas do usuário e de seus contatos. Esses dados vêm em pacotes que foram denominados de pacotes Rich Presence, no entanto, eles não trazem informações sobre presença para troca de mensagens instantâneas (já foram pedidas na requisição anterior), mas somente sobre disponibilidade para recebimento de chamadas, disponibilizando informações como discutido na seção 3.1.4, que são:

- Estados dos ramais dos usuários (livre, ocupado ou indisponível);
- Desvios de chamada ativos;
- Disponibilidade para videoconferência;
- Disponibilidade para correio de voz.

Vale lembrar que os pacotes de Rich Presence são enviados ao Servidor UC pela LibIntUC, como discutido na seção 3.2.2.1, para que sejam disponibilizados aos demais usuários.

Dessa forma, quando o Client-UI envia o comando para pedido de Rich Presence, o Client-Core requisita essas informações ao Componente XMPP e as retorna à interface. No entanto, todas as informações de Rich Presence também ficam armazenadas no Client-Core, por meio de uma tabela hash para que sejam reutilizadas em ações que serão discutidas na seção 5.3.5.

A Figura 5.8 mostra como acontece essa ação.

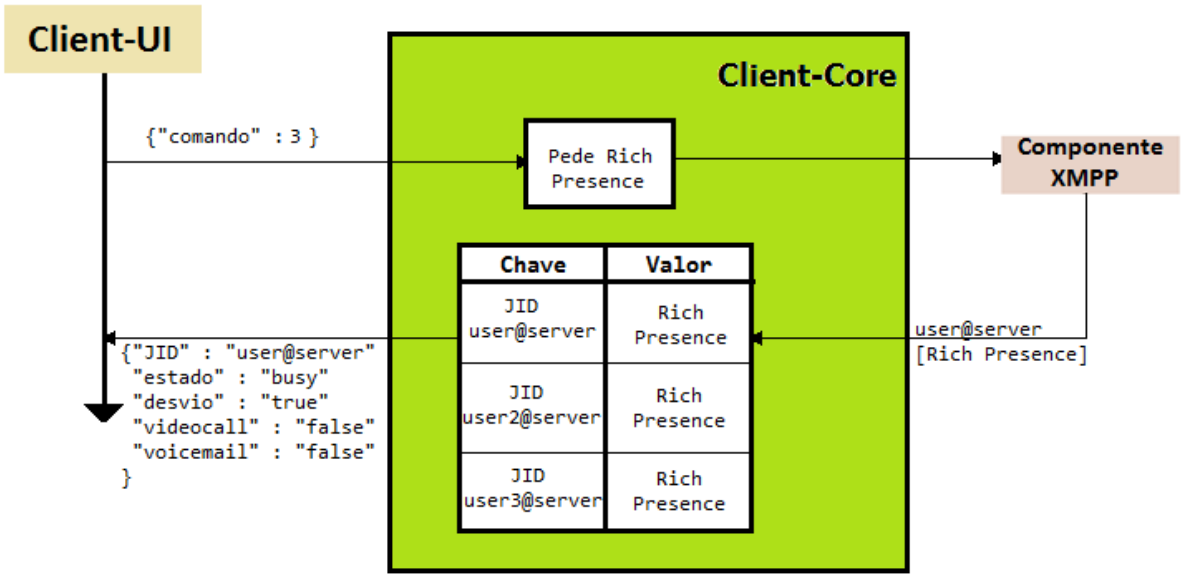

Figura 5.8 - Requisição de pacotes de Rich Presence.

O quadro “Pede Rich Presence” na imagem significa que o Client-Core deve pedir informações sobre disponibilidade para recebimento de chamadas do usuário e de seus contatos. Os pacotes com essas informações são então enviados um a um para o Client-UI.

Quando essa tarefa é concluída, encerram-se as requisições de inicialização, e o que o usuário consegue ver é uma interface semelhante ao esboço mostrado na Figura 5.9.

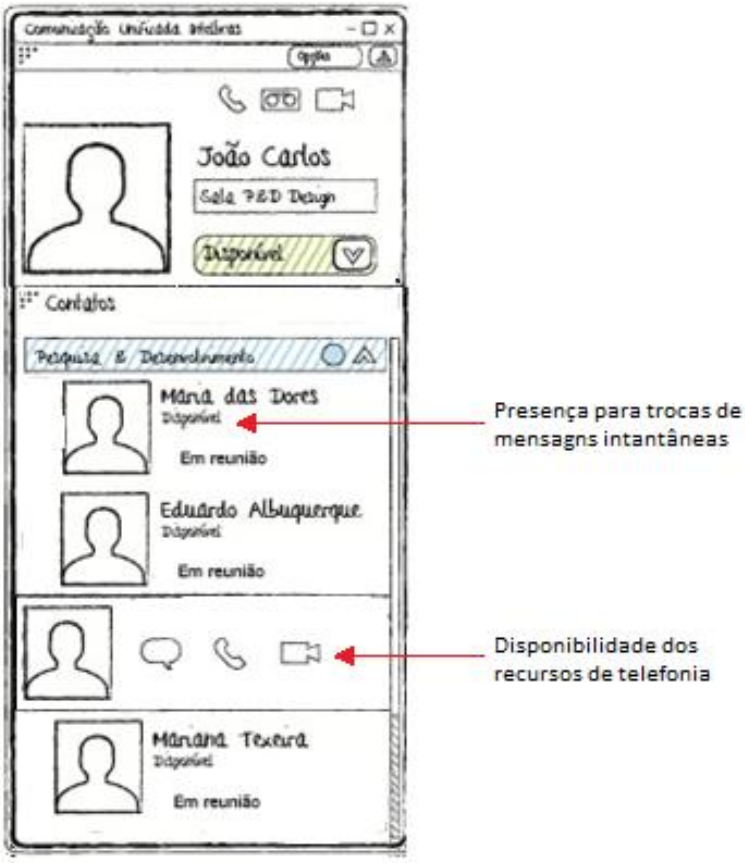

Figura 5.9 - Interface principal do Client-UI.

A Figura 5.9 é, na verdade, o Client-UI que está sendo desenvolvido na Intelbras. No entanto, essa interface ainda não está concluída e ainda não foi lançada nenhuma versão preliminar, dessa forma, o esboço tenta somente ilustrar como ficarão dispostas as informações de presença.

5.3.3: Notificações

Essas informações passam a ser recebidas depois que o Client-UI já concluiu as requisições de informações iniciais, discutidas na seção anterior. Ou seja, as informações relacionadas às notificações têm por objetivo atualizar dados ou

informar o usuário sobre interações que membros de seu Roster queiram fazer com ele.

As notificações podem, então, trazer dados como:

- Atualização de presença de chat dos contatos;
- Atualização de disponibilidade nos dispositivos de telefonia dos contatos;
- Recebimento de mensagens instantâneas;
- Recebimento de solicitação para participação em salas de chat;
- Recebimento de solicitação de transferência de arquivos;
- Recebimento de chamada no ramal;
- Perda de conectividade com o Componente XMPP.

Todas essas notificações são recebidas por meio de *listeners* implementados no Client-Core, ou seja, o núcleo do cliente de acesso tem rotinas que ficam esperando por alguma notificação a ser recebida do Componente XMPP para então repassá-las ao Client-UI. O Componente XMPP, por sua vez, sabe quais informações interessam a determinados usuários por meio de mecanismos PubSub como discutido na seção 3.2.2.1, ou por identificadores de destino nas Stanzas trocadas.

Vale lembrar que notificações de alteração de disponibilidade nos dispositivos de telefonia do usuário e contatos são sempre armazenadas na tabela hash mostrada na Figura 5.8 e posteriormente enviadas ao usuário. Já as demais notificações são diretamente repassadas ao Client-UI.

A Figura 5.10 ilustra o exemplo de como a informação sobre a alteração na presença de um contato do Roster do usuário chega a ele.

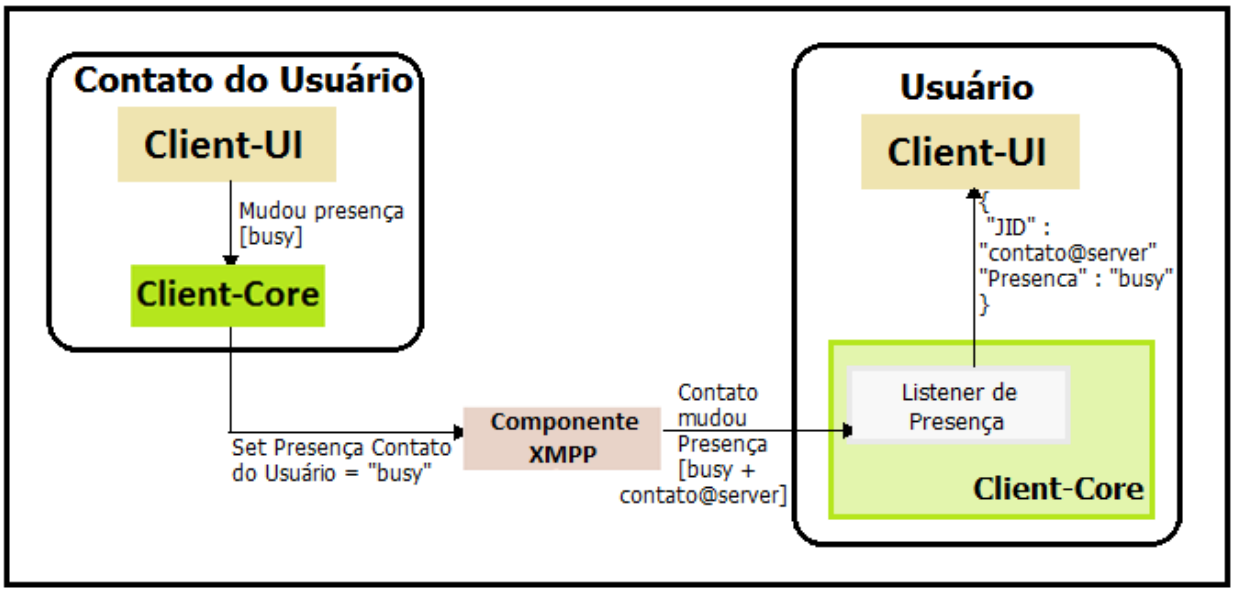

Figura 5.10 - Notificação sobre mudança de presença no Roster do usuário.

O contato do usuário troca a presença através de sua interface. Esse parâmetro, no caso da imagem, “busy”, é enviado ao Client-Core que informa ao Componente XMPP sobre a mudança. Este, por sua vez, identifica quem são os clientes interessados em receber a presença do contato, e envia ao usuário esse parâmetro. O listener de presença de chat do Client-Core do usuário recebe o parâmetro mudado e quem o modificou, repassando ao seu Client-UI a notificação de mudança de presença.

Como citado anteriormente, o Client-Core possui um listener para cada tipo de notificação, No entanto, as rotinas para recebimento de dados por meio desses listeners é sempre semelhante ao da Figura 5.10. O que diferencia cada listener é o fato de terem filtros que conseguem separar mensagens de acordo com seus identificadores, direcionando essas mensagens para os seu tratamento correto.

A Figura 5.11 mostra o que acontece dentro do “Listener de Presença” apresentado na Figura 5.10.

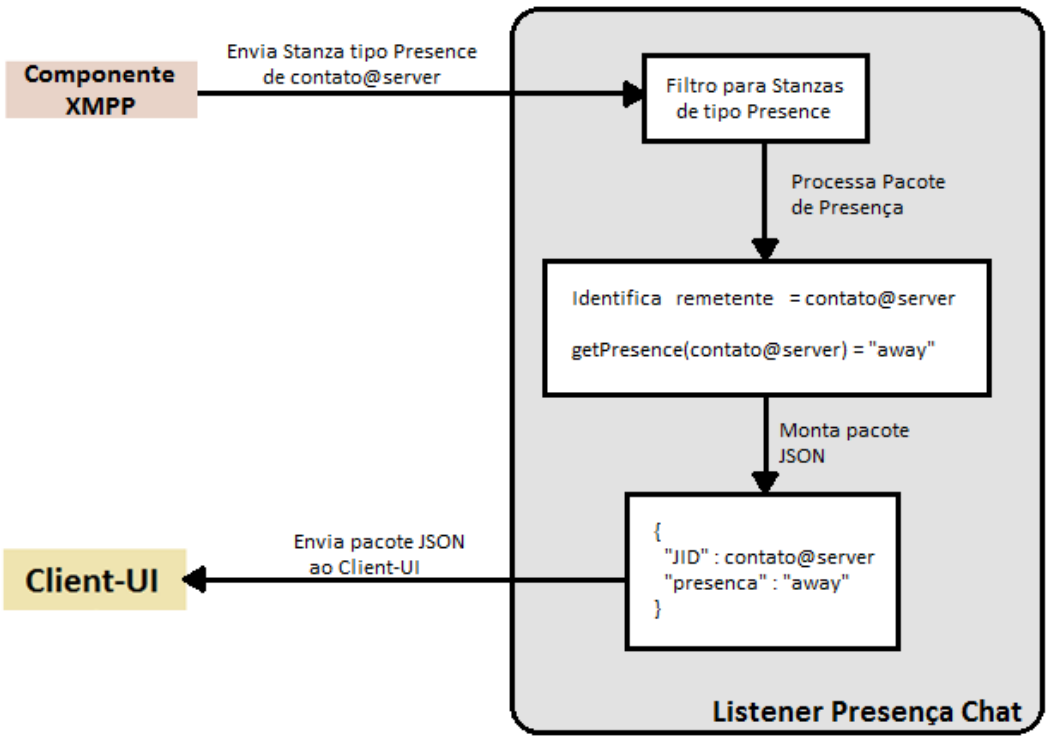

Figura 5.11 - Listener de Presença.

O Client-Core recebe a Stanza do tipo Presence do Componente XMPP indicando que algum contato do usuário trocou o estado da presença de chat. O listener de presença, que é responsável por filtrar pacotes do tipo Presence, recebe a mensagem e identifica o remetente, nesse caso “contato@server”. Quando se conhece o remetente, basta que se identifique para qual valor o parâmetro presença foi mudado. Tendo conhecimento dos parâmetros necessários, um objeto JSON é montado e enviado ao Client-UI para que o usuário seja notificado da mudança de presença.

5.3.4: Configurações de serviços

Outra categoria de atividades do Client-Core é a relacionada às configurações de serviços da solução de UC. Nesse bloco de atividades o usuário pode gerenciar alguns de seus dados que dizem respeito tanto aos serviços de mensagem instantânea (chat) e dados do usuário, quanto aos de telefonia.

As configurações de mensagens instantâneas e dados do usuário diferem das configurações de telefonia pelo fato de que as primeiras ficam armazenadas somente no Componente XMPP, já quando se configura um dado relacionado à telefonia, ele precisa ser também enviado à plataforma de telefonia.

A participação do Client-Core nesse tipo de atividade consiste em receber dados que o usuário determina em seu Client-UI e enviá-los ao Componente XMPP para que distribua ao Roster do usuário.

5.3.4.1: Configurações de chat e dados do usuário

Basicamente, todas as informações de chat e dados do usuário serão encaminhadas ao Componente XMPP e distribuídas aos interessados em receber esses dados, ou seja, os contatos do Roster do usuário. Os contatos recebem essas informações por meio de listeners como mencionado na seção 5.3.3.

As configurações de chat estão ligadas à mudança no estado de presença para troca de mensagens instantâneas. Em outras palavras, o usuário poderá mudar sua presença (available, busy, away), e ainda configurar uma mensagem que ajude a identificar a presença apontada. A Figura 5.12 mostra onde o usuário poderá fazer essas alterações em seu Client-UI.

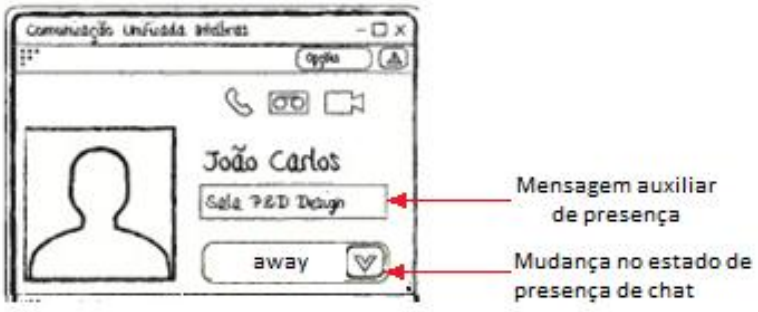

Figura 5.12 - Configurações de presença de chat.

Essas informações que podem ser alteradas, como mostrado na imagem acima, são direcionadas aos contatos segundo fluxo apresentado na Figura 5.10.

Já os dados do usuário, os quais podem ser acessados por qualquer contato de seu Roster, dizem respeito a informações pessoais do usuário como:

- Nome e sobrenome;
- E-mail pessoal;
- Nome da empresa e setor onde trabalha;
- Imagem pessoal.

Todas essas informações são configuradas via interface e enviadas ao Client-Core, que as redireciona ao Componente XMPP, de onde os contatos do usuário conseguem acessá-las.

5.3.4.2: Configurações de telefonia

Essas configurações estão relacionadas aos serviços de telefonia oferecidos na solução de UC, ou seja, estão diretamente ligadas à plataforma de telefonia integrada ao sistema.

Todas as configurações de telefonia são enviadas ao Componente XMPP que as repassa à plataforma de telefonia, de onde vêm os pacotes de Rich Presence, como mostrado na seção 3.2.2.1.

As configurações de telefonia possíveis são as mencionadas na seção 3.1.4:

- Configuração de números alternativos;
- Chamada a Múltiplos Dispositivos (CMD).
- Desvio de chamada;
- Habilitação de mensagem unificada;
- Não perturbe (DND).

Essas configurações, assim como as de presença e dados do usuário, também serão direcionadas aos contatos do Roster do usuário, no entanto, antes que elas cheguem a essa lista de contatos, é necessário que a plataforma de telefonia seja informada de tais mudanças. A Figura 5.13 mostra a diferença no caminho que esses dados percorrem dentro do sistema de UC.

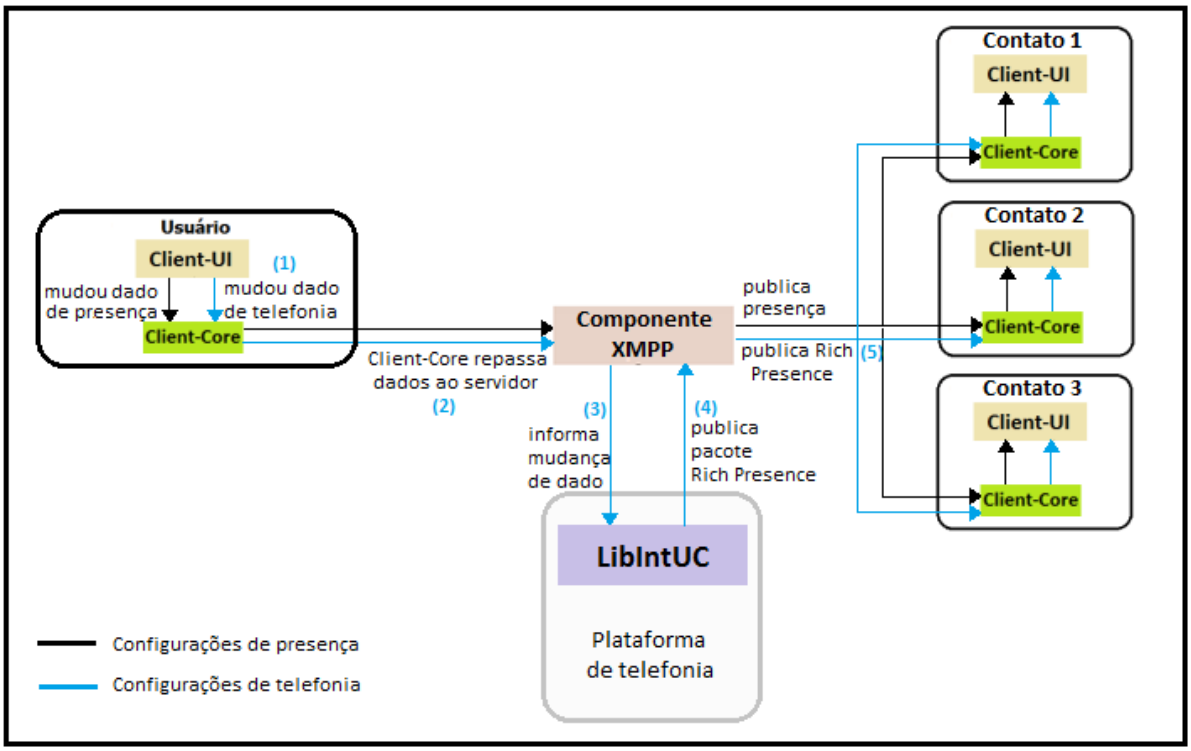

Figura 5.13 - publicação de mudança de configurações.

Através da imagem é possível perceber, pela sequência numérica das mensagens, que as configurações de telefonia precisam ser repassadas primeiramente à plataforma para depois serem publicadas aos contatos do usuário, ao contrário das configurações de presença, as quais já são repassadas diretamente pelo Componente XMPP. Isso acontece porque é a LibIntUC que trata as informações de telefonia e as publica em forma de pacotes de Rich Presence.

5.3.4.3: Mecanismo para publicação de configurações

A Figura 5.13 na seção anterior mostrou como configurações alteradas pelo usuário chegam aos seus contatos. Entretanto, ela ainda não expõe como o Client-Core executa a tarefa que lhe cabe nesse processo, ou seja, receber as informações passadas pelo Client-UI e direcioná-las ao Componente XMPP.

O mecanismo para envio de configurações ao Servidor UC é o inverso do procedimento de notificações. Como mencionado na seção 5.3.3, quando o usuário

recebe uma notificação do Componente XMPP, a mensagem recebida vem sob o protocolo XMPP e precisa ser convertida em JSON, para então ser enviada ao Client-UI. Já o mecanismo de publicação de configurações recebe uma mensagem em JSON do Client-UI e precisa converter em XMPP, ou seja, precisa gerar uma Stanza IQ e enviá-la ao Componente XMPP. A Figura 5.14 mostra o mecanismo para envio de configurações ao Componente XMPP.

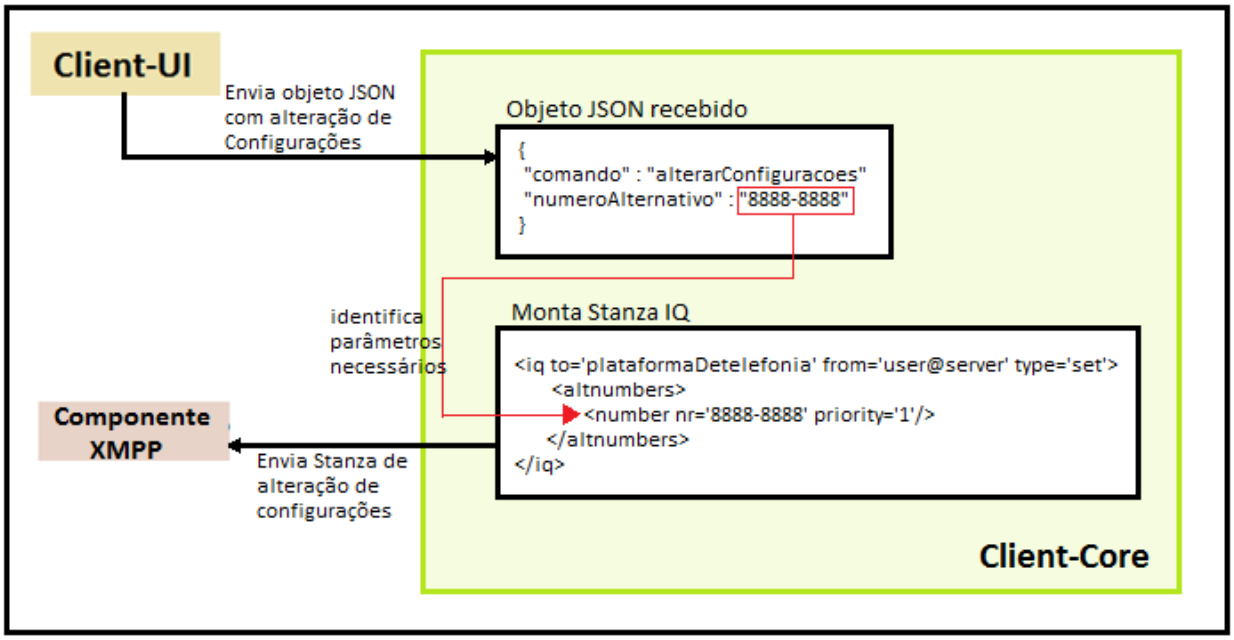

Figura 5.14 - Mecanismo para envio de alteração de configurações.

Pela imagem, vê-se que, após alterar uma configuração de números alternativos, o Client-UI envia um objeto JSON ao Client-Core que interpreta as informações recebidas e as coloca em uma Stanza para ser enviada ao Componente XMPP, de onde será publicada aos usuários, segundo fluxo mostrado na Figura 5.13.

5.3.5: Ações sobre os contatos do usuário

A última das categorias de atividades executadas pelo Client-Core são as ações sobre contatos. Essas tarefas são a forma direta de promover comunicação entre usuários na solução de UC, ou seja, com esse grupo de atividades o usuário

tem a possibilidade de trocar informações não só com o sistema de UC, mas também com os contatos contidos em seu Roster.

As ações sobre os contatos também podem ser divididas em duas categorias: ações de mensagens instantâneas e ações de telefonia.

5.3.5.1: Ações de mensagens instantâneas

Nesse grupo de atividades o usuário utiliza a interface principal do Client-UI para requisitar atividades como:

- Trocar mensagens instantâneas;
- Criar salas de chat;
- Criar grupos de usuários;
- Fazer transferência de arquivos.

Troca de mensagens instantâneas

A primeira ação da lista significa que o usuário pode iniciar um chat selecionando um contato de seu Roster que aparece na interface principal do Client-UI.

Nesse caso, o Client-Core precisa receber a mensagem enviada pelo usuário e o contato para quem ela deverá ser direcionada. Assim, o Client-Core consegue criar uma Stanza do tipo Message, e enviá-la ao Componente XMPP, que se encarregará de encaminhá-la ao seu destino. A Figura 5.15 mostra como funciona o mecanismo de troca de mensagens instantâneas.

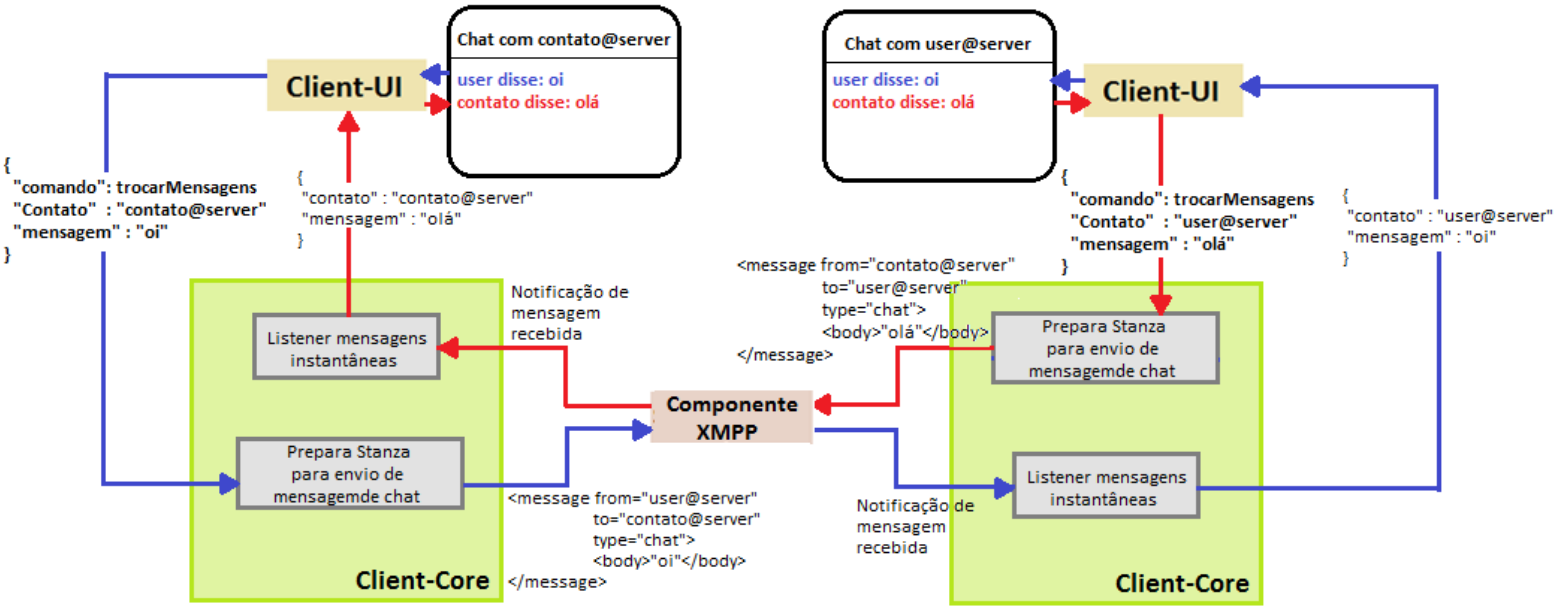

Figura 5.15 - Troca de mensagens instantâneas.

Pela imagem é possível ver que, quando o Client-Core recebe o objeto JSON contendo as informações necessárias para troca de mensagens instantâneas, ele prepara a Stanza enviada ao Componente XMPP, a qual é direcionada ao seu destino. Já o recebimento de mensagens instantâneas foi tratado no item de Notificações (5.3.3) e é feito a partir de um listener para mensagens instantâneas, o qual trata a mensagem recebida e a envia ao Client-UI por um objeto JSON.

Criação de salas de chat

A criação de salas de Chat segue o mesmo princípio da troca de mensagens instantâneas mostrado na Figura 5.16, entretanto, salas de chat são compostas por três ou mais usuários. Dessa forma, quando uma mensagem é enviada pelo usuário, o destino dessa mensagem passa a ser o grupo de contatos que compõem a sala de chat.

Outra diferença entre a troca de mensagens instantâneas com dois usuários e a sala de chat é que, no segundo caso, antes de começar a troca de mensagens propriamente dita, é necessário que uma sala seja criada pelo Client-Core e que os

contatos com os quais o usuário deseja se comunicar sejam inseridos um a um na sala recém criada.

Criação de grupos

A criação de grupos nada mais é do que a união de vários contatos do usuário em uma categoria. Esse serviço da solução poderia também ser considerado como um item de configurações de chat do usuário, descritos na seção 5.3.4.1, pois pode ser entendido como um ajuste que o usuário faz sobre seus contatos. No entanto, a criação de grupos foi enquadrada na categoria de ações sobre os contatos, porque ela é muito útil para formar salas de chat e iniciar chamadas telefônicas com um grupo de usuários, como será descrito na seção 5.3.5.2.

Segundo apresentado na seção anterior, para gerar uma sala de chat, o usuário precisa primeiramente criar essa sala e, em seguida, inserir os usuários com os quais deseja se comunicar um após o outro. No entanto, quando o usuário cria grupos com seus contatos, ele pode inserir vários usuários na sala de chat simultaneamente, agilizando o processo para iniciar um conversa em grupo. A Figura 5.16 mostra com funciona o mecanismo para criação de salas de chat. Como se pode perceber, quando o usuário adiciona um grupo na criação de uma sala, o próprio Client-Core é que se encarrega de identificar quais são os participantes do grupo apontado, selecionando-os com o intuito de iniciar a conversa na sala. Já no caso em que o usuário não possui grupo de contatos, ele precisa selecioná-los um a um para que façam parte do chat.

Vale lembrar que, além de auxiliar na criação de salas de chat e chamadas telefônicas, a criação de grupos pode ser importante para outros serviços como transferência de arquivos, já que esses tipos de recursos também permitem que se adicionem grupos para utilizá-los.

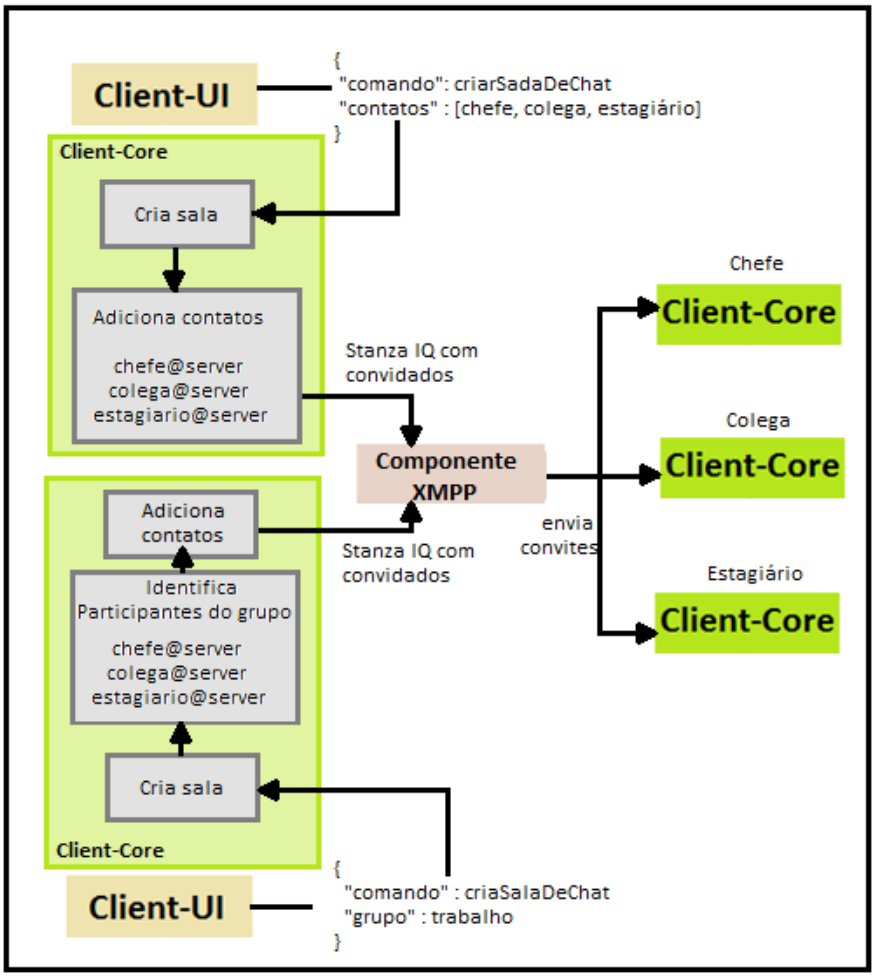

Figura 5.16 - Criação de salas de chat.

5.3.5.2: Ações de telefonia

As ações de telefonia são aquelas nas quais o usuário faz uso da interface principal do Client-UI para interagir com seus contatos contando com os recursos de telefonia (terminais telefônicos e plataforma de telefonia).

As funções implementadas que se relacionam com essa categoria são:

- Click-to-Call;
- Click-to-Conference.

Click-to-Call

Essa funcionalidade pode ser considerada uma das mais importantes dentro do sistema de UC da Intelbras, isso porque ela utiliza praticamente todos os elementos contidos na solução, além de terminais telefônicos, necessitando de uma integração bastante robusta entre todas essas ferramentas.

O Click-to-Call permite que dois usuários iniciem uma chamada telefônica por meio da interface gráfica do sistema de UC. Em outras palavras, nenhum usuário precisa utilizar seu aparelho telefônico e discar para o contato com o qual deseja se comunicar. Basta que ele pressione um botão na interface para que uma chamada telefônica seja disparada tanto para o seu aparelho telefônico, quanto para a pessoa com quem se quer falar. A Figura 5.17 mostra como funciona o Click-to-Call.

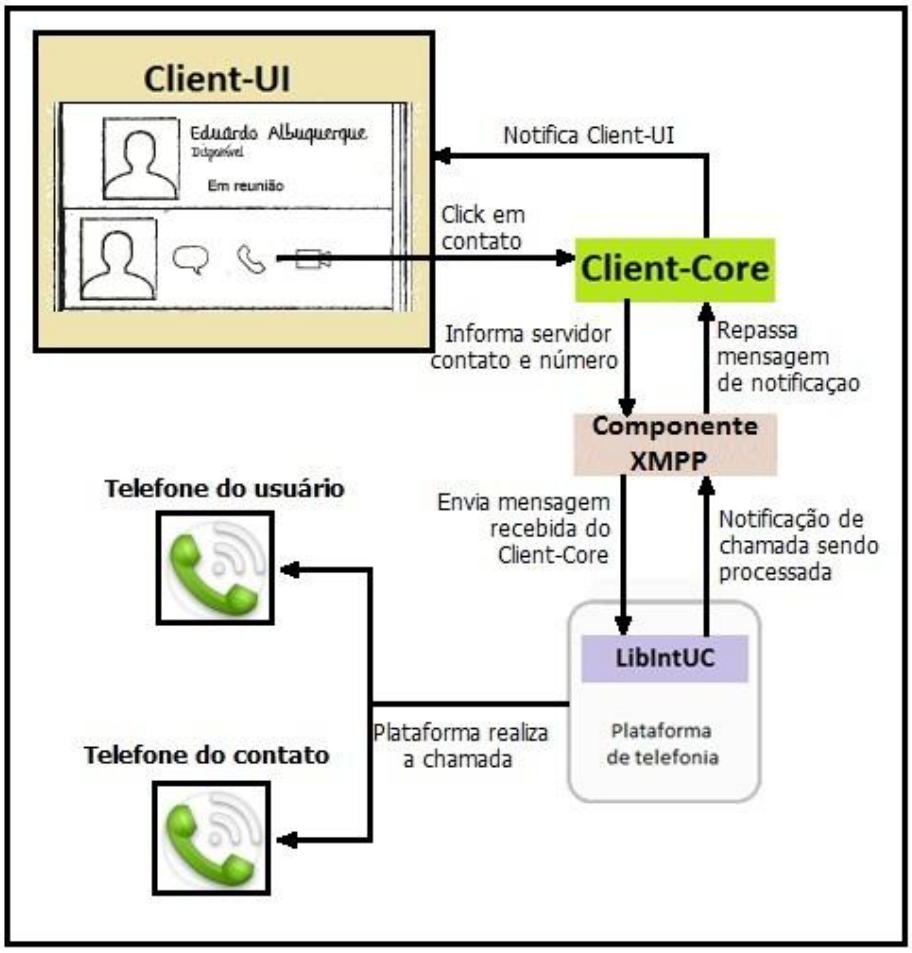

Figura 5.17 - Click-to-Call na solução de UC da Intelbras.

O ciclo do Click-to-Call se inicia quando o usuário identifica a pessoa com quem deseja falar e pressiona o botão referente a chamadas telefônicas na interface. Nesse momento o Client-Core é informado que uma ligação necessita ser efetuada entre o usuário e o contato selecionado. Assim, o núcleo do Cliente envia uma Stanza IQ para o Componente XMPP informando os dados necessários para que se efetue a chamada. O Componente XMPP, por sua vez, repassa os dados para a LibIntUC, a qual se encarregará de informar a plataforma de telefonia, de onde a ligação solicitada será finalmente gerada.

A notificação de chamada sendo processada tem apenas o objetivo de informar ao usuário que a ligação será realizada em breve. Entretanto, essa notificação também poderá alertar sobre uma possível indisponibilidade do serviço de Click-to-Call, impossibilitando a efetuação de chamadas telefônicas.

Apesar de vários elementos na solução possuírem sua devida importância nessa funcionalidade, o Client-Core tem uma tarefa bastante fundamental para que a ligação ocorra, isso porque é ele que mantém em sua estrutura os dados necessários para que a ligação seja efetuada com sucesso. Dessa forma, o Client-Core deve colher todos os dados e repassá-los ao Componente XMPP para que o ciclo do Click-to-Call tenha continuidade.

Na Figura 5.18 é possível ver como o Client-Core executa sua parte dentro do ciclo de Click-to-Call. Quando recebe o JID do contato oferecido pelo Client-UI, o Client-Core deve identificar quais os dados necessários para a execução da chamada. Para isso, ele consulta as informações de disponibilidade para recebimento de chamadas que estão armazenadas em uma tabela hash em forma de pacotes Rich Presence, segundo discutido no item 5.3.2, usando como chave o JID do usuário. Outra consulta é feita nessa tabela também com o objetivo de identificar os dados do pacote de Rich Presence do próprio usuário. Assim, tendo posse de todas as informações necessárias, o Client-Core tem como tarefas finais montar a Stanza IQ com os dados coletados e enviá-la ao Componente XMPP de onde o ciclo tem continuidade.

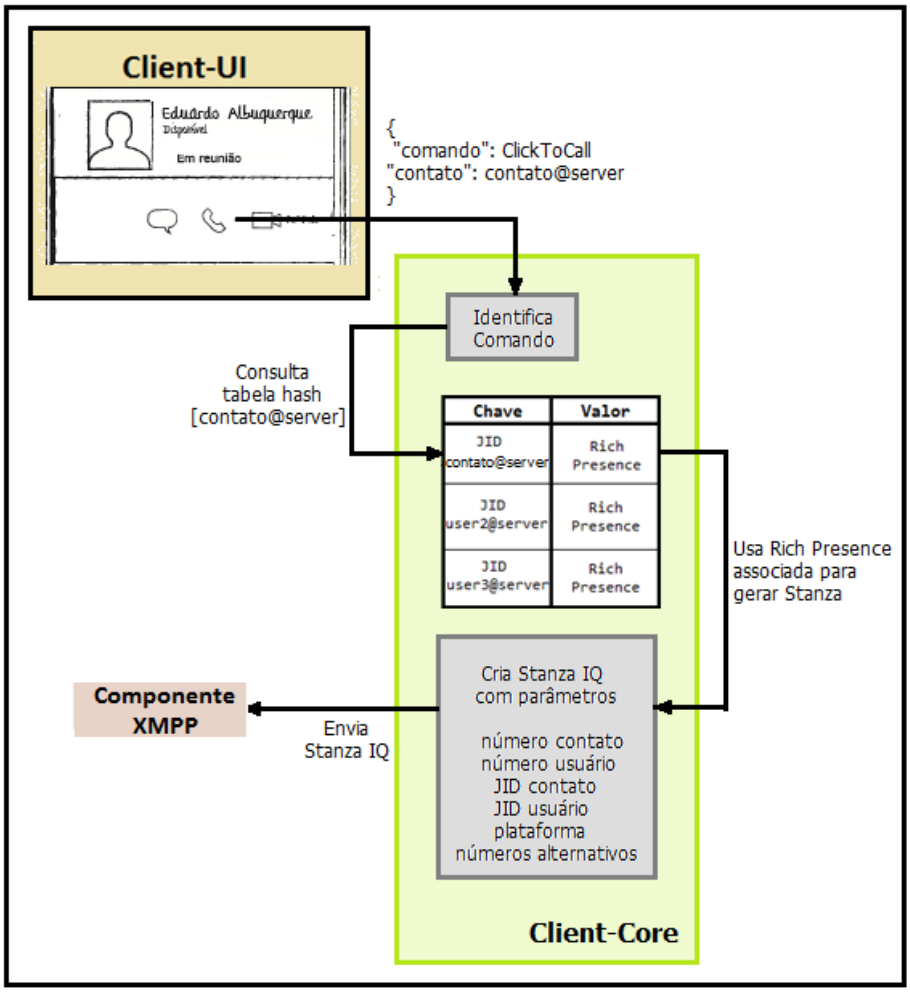

Figura 5.18 - Participação do Client-Core na funcionalidade de Click-to-Call.

Vale ainda salientar que essa funcionalidade de Click-to-Call pode ser combinada com as configurações de telefonia do usuário. Por exemplo, caso um usuário tenha ativado a opção de configuração CMD (vide seção 3.1.4), a ligação do Click-to-Call não será feita exclusivamente para um número do contato com quem se deseja falar, e sim para todos os números configurados por ele, os quais tocarão ao mesmo tempo.

Click-to-Conference

A funcionalidade de Click-to-Conference pode ser entendida como uma extensão do Click-to-Call. Na verdade, o ciclo para que a chamada se realize é

semelhante ao apresentado na Figura 5.18. A diferença nesse caso é que vários usuários podem ser chamados ao mesmo tempo. Ou seja, a Stanza IQ gerada pelo Client-Core conterá os dados de mais contatos selecionados pelo usuário, e a plataforma de telefonia é que se encarregará de realizar a chamada entre três ou mais pessoas.

É importante lembrar também, como citado na seção anterior, que a funcionalidade de Click-to-Conference permite que o usuário adicione grupos para que se efetue a chamada, oferecendo mais agilidade nessa funcionalidade.

Finalizados os aspectos de implementação e funcionamento do Client-Core, pode-se partir agora para aspectos de utilização dos softwares implementados, ou seja, no próximo capítulo serão abordados os testes e resultados obtidos com a utilização de Client-d e Client-Core na solução de UC da Intelbras.

Capítulo 6: Testes e Resultados

Nos dois capítulos anteriores foram apresentados os módulos que compõem o cliente de acesso na solução de UC: Client-d e Client-Core. Nesses capítulos se discutiu a respeito de como esses elementos agem dentro do sistema e o modo como foram implementados para seguirem uma série de requisitos.

No entanto, com os dois módulos do cliente já desenvolvidos, faz-se necessário submetê-los a testes para verificar aspectos de robustez e usabilidade dentro do serviço de UC.

Por conveniência, primeiramente serão apresentados os testes realizados com o Client-d e, posteriormente, com o Client-Core. Os testes de cada software foram aplicados com o objetivo de analisar e, possivelmente, corrigir os comportamentos dos módulos clientes em situações extremas, além de examinar os seus comportamentos em um longo período de atuação dentro da solução sem que fossem interrompidos.

6.1: Testes com o Client-d

Os testes realizados no Client-d foram baseados nos pré-requisitos de sua implementação, ou seja, foi necessário verificar se o software conseguiria gerenciar múltiplas conexões de maneira rápida e sem apresentar falhas em sua execução, o que poderia comprometer todo o processo de conexões de usuários ao sistema de UC.

6.1.1: Testes para controle de exceções

Nessa bateria de testes foram propostos cenários a fim de analisar o comportamento do software gerente de conexões em situações como:

- Falta de portas para alocar novos clientes;

- Atingimento do número máximo de conexões em espera para atendimento;
- Gerenciamento de um número elevado de requisições feitas por sockets clientes.

6.1.1.1: Falta de portas para novas conexões

Uma situação em que faltem portas para serem disponibilizadas para novas conexões de usuários é altamente indesejável dentro do sistema de UC, isso porque o Client-d não terá como instanciar um novo Client-Core para que um cliente o utilize.

Entretanto, caso esse problema ocorra é necessário que o Client-d saiba gerenciá-lo sem interromper ou prejudicar seu funcionamento. Para verificar o comportamento do Client-d frente a essa situação foi montado um cenário com as seguintes configurações:

- Foram disponibilizadas ao Client-d apenas três portas para que clientes pudessem fazer suas conexões;
- Três usuários requisitaram novas conexões ao Client-d ocupando todas as portas disponibilizadas;
- Um quarto cliente pede uma nova requisição enquanto as três portas estão sendo utilizadas em outra sessão.

O resultado do pedido da quarta conexão é mostrado na Figura 6.1, a qual mostra os logs do Client-d.

Figura 6.1 - Logs do Client-d apontando falta de portas.

Como pode ser visto pela Figura 6.1, quando o Client-d recebe o pedido de uma nova conexão e não tem portas para alocar novos clientes, ele instancia um Client-Core passando como parâmetro uma porta com o valor -1, o que inviabiliza a instanciação de um novo Client-Core, gerando a exceção também mostrada na imagem.

Para que o usuário não fique sem obter uma resposta sobre sua conexão ao sistema de UC nessa situação, uma notificação lhe é enviada pelo Client-d informando que a conexão não foi estabelecida, e uma nova tentativa será feita.

A continuidade do teste se deu quando uma das portas que estavam sendo utilizadas foi liberada. Dessa forma o usuário que anteriormente não conseguiu iniciar uma sessão pôde se conectar normalmente ao sistema com a porta liberada.

Esse cenário foi repetido vinte vezes, apresentando sempre o mesmo comportamento. Isso permite afirmar que o Client-d se comporta da maneira esperada nesse tipo de situação.

Vale acrescentar ainda que a falta de portas para iniciar sessões no serviço de UC não deverá ocorrer de maneira rotineira no Client-d, visto que haverá um limite máximo de Client-Cores a ser instanciados como será discutido nos testes da sessão 6.2.3, o que permite também dimensionar o número total de portas que estarão disponíveis.

Entretanto, esse teste foi realizado para verificar e tratar o comportamento do Client-d em um caso de exceção, o qual pode vir a acontecer se, por exemplo, o Client-d não receber a mensagem de encerramento de alguma sessão. Dessa forma, a porta não voltaria à pilha de portas para reuso, e o número de Client-Cores que podem ser agora instanciados passa a ser menor que a quantidade de portas.

6.1.1.2: Atingimento do número máximo de conexões em espera

Como mencionado na seção 4.5.1.1 o Client-d possui uma fila de espera para atender requisições de novas conexões ao sistema. No entanto, se muitas conexões acontecerem simultaneamente, essa fila pode atingir seu valor máximo. Esse caso pode ocorrer, por exemplo, quando a rede de uma empresa para de funcionar, desconectando todos os clientes. No momento em que a rede é recuperada, os usuários vão pedir automaticamente para iniciarem novas sessões, podendo cobrir todas as posições da fila de espera.

Para verificar o comportamento do Client-d nessa situação foi preparado o seguinte cenário:

- O software do Client-d foi modificado para não tirar nenhum pedido de conexão da fila.
- O tamanho máximo da fila foi configurado para três posições.
- Três usuários requisitaram conexão ao Client-d, ocupando todas as posições da fila.
- Um quarto cliente tenta então se conectar ao sistema.

Nas primeiras vezes em que o Client-d foi executado dentro do cenário montado, ele encerrava seu funcionamento em todas as situações em que o quarto usuário tentava se conectar ao sistema. Esse comportamento, no entanto, não é desejável, pois o software para gerência de conexões é interrompido, impossibilitando que clientes iniciem novas sessões.

Para solucionar o problema foi preciso identificar no código do Client-d o ponto em que a fila atingia seu número máximo. Nesse ponto verificou-se que havia

um comando para encerramento do programa. Sendo assim, essa seção do software foi alterada para notificar o cliente sobre a impossibilidade momentânea de se conectar ao sistema, e o comando de encerramento foi retirado.

Com a nova modificação, o cenário foi também replicado vinte vezes e apresentou em todas as ocasiões o comportamento esperado.

É pertinente comentar também que essa fila deve ser dimensionada com o tamanho bastante próximo ao máximo de Client-Cores que podem ser instanciados pelo Client-d como medida para evitar um congestionamento de requisições de novos clientes.

6.1.1.3: Gerenciamento de muitas requisições feitas por sockets clientes.

Após realizar o teste e resolver o problema de atingimento do limite da fila de requisições, foi possível realizar mais uma prova à usabilidade do software: o gerenciamento contínuo de múltiplas conexões.

Para esse teste foi montado o seguinte cenário:

- Foi criado um pequeno software com o objetivo de fazer conexões e desconexões simultâneas no serviço de UC, ou seja, o programa é capaz de fazer o login de vários usuários no sistema, e uma vez conectados, ele efetua os logouts.
- O Client-d foi configurado para poder gerenciar até quinhentas conexões ao mesmo tempo, devendo atender todas as conexões do login ao logout.

Nesse caso o teste foi feito em escalas. Primeiramente, o Client-d teve que gerenciar cinquenta conexões simultâneas, executando todos os fluxos de conexões sem maiores problemas.

O segundo degrau foi feito com cem usuários tentando efetuar o login no sistema. O que se pôde perceber nesse caso foi um tempo maior para que uma conexão conseguisse de fato ser realizada. Isso acontece pelo fato de a fila de espera por conexões aumentar em relação ao primeiro teste. Entretanto, esse

acréscimo de tempo pode ser considerado desprezível levando-se em consideração que a diferença do tempo para atender todas as cem e cinquenta conexões chegou a cerca de 4,5 segundos.

A terceira etapa foi o teste com duzentas conexões, na qual se verificou também um pequeno acréscimo no tempo, porém, o Client-d conseguiu manejar todas as conexões.

Para finalizar, foi realizado o teste com quinhentos usuários. O Client-d mais uma vez conseguiu gerenciar todas as conexões em um tempo em torno de quinze segundos maior que o caso com cinquenta usuários, o que é ainda um resultado bastante satisfatório em seu funcionamento. A Tabela 3 mostra os tempos que o Client-d usou para atender do login ao logout determinados números de conexões simultâneas.

Tabela 3 - Tempo para atendimento de conexões.

Número de conexões atendidas	Tempo para atendimento (s)
50	5,1
100	9,7
200	13,9
500	20,7

Pela tabela é possível perceber que os tempos de gerenciamento podem ser considerados satisfatórios em relação ao número de conexões simultâneas. Na situação de 500 conexões, na pior das hipóteses, um usuário esperaria cerca de 21 segundos para ter sua conexão efetivada, o que ainda pode ser considerado um bom resultado.

Finalizados os testes de situações críticas, foram realizados também testes de longa duração, em que o Client-d passou por um longo período de funcionamento contínuo, sem interrupções ou alterações.

6.1.2: Testes de longa duração

Nessa bateria de testes foi avaliada a capacidade de o Client-d ficar por longos períodos em funcionamento, sem interrupções. Para tanto, o software ficou durante sete dias em operação e foram avaliados os seguintes pontos:

- Problemas com alocação de memória;
- Problemas com envio e recebimento de mensagens.

6.1.2.1: Alocação de memória

Para verificar questões de alocação de memória foi utilizada como ferramenta auxiliar o Valgrind. Essa ferramenta é um framework que tem por objetivo fazer analises dinâmicas de gerenciamento de memória e possíveis *bugs* que o sistema testado possa apresentar.

Dessa forma, o resultado mais importante apresentado pela ferramenta após os sete dias de uso do Client-d são mostradas na Figura 6.2.

```
==29525== HEAP SUMMARY:
==29525==     in use at exit: 6,389 bytes in 27 blocks
==29525==   total heap usage: 4,823 allocs, 4,796 frees, 911,416 bytes allocated
==29525==
==29525== LEAK SUMMARY:
==29525==    definitely lost: 0 bytes in 0 blocks
==29525==    indirectly lost: 0 bytes in 0 blocks
```

Figura 6.2 - Memória alocada no Client-d.

Pela Figura 6.2 se consegue perceber que, durante toda a semana em que ficou em execução, o Client-d não teve problemas com alocação de memória, já que não apresentou perdas de bytes durante a execução (nesse caso, perdas significam que bytes não foram corretamente alocados ou desalocados), indicando que o

gerenciamento de memória do software desenvolvido em C foi implementado com eficácia.

6.1.2.2: Recebimento de mensagens

Durante os sete dias em que ficou ativo, o Client-d foi encarregado de gerenciar uma quantidade bastante grande de conexões. No entanto, foi verificado um problema em sua execução: em alguns casos a mensagem de encerramento de sessões não chegou ao Client-d. Essa situação acarreta a não disponibilização de portas para reuso, o que é indesejável, mesmo que não interrompa o funcionamento do software.

A solução para esse problema então será fazer com que o Client-d verifique periodicamente no sistema quais portas estão realmente sendo usadas. Se alguma porta não estiver sendo utilizada e estiver contida na tabela hash de conexões (vide seção 4.5.2.3) será reintegrada à pilha de portas.

Após a resolução do problema de envio de mensagens, e com todos os outros pontos apontados e resolvidos durante os testes, pode-se então validar a utilização do Client-d dentro da solução de UC.

A partir do controle de exceções de algumas situações, além da verificação de bom funcionamento do software em regime permanente, é possível afirmar que ele cumpre os requisitos exigidos: robustez e velocidade para gerenciar as conexões.

Completados testes e validação do Client-d, serão apresentados então, os testes e resultados obtidos com o Client-Core.

6.2: Testes com o Client-Core

O Client-Core tem como principal tarefa ligar a interface do usuário ao serviço de UC, cumprindo todas as funcionalidades propostas durante o Capítulo 5.

Entretanto, o foco dos testes aplicados a esse software é a validação de sua utilização dentro da solução de UC, abordando os impactos que ele traz ao sistema. Para tanto, foram aplicados três testes ao Client-Core:

- Tempo de inicialização de Client-Core remoto versus Client-Core local;
- Carregamento de informações de contatos e envio de mensagens;
- Memória alocada para instâncias do Client-Core.

6.2.1: Tempo de inicialização de Client-Core remoto versus Client-Core local

Como mencionado na seção 3.2.3.1, o Client-Core poderá ficar junto ao Servidor UC, sendo acessado remotamente por interfaces Web e Mobile, ou poderá estar localizado no próprio dispositivo do usuário caracterizando um Client-Core local. A diferença básica entre esses dois processos é que o Client-Core remoto precisa ser instanciado pelo Client-d, enquanto o local se conecta diretamente à interface. No entanto, a maior dissemelhança a ser considerada é que as mensagens trocadas entre interface (Client-UI) e o Client-Core local não precisam circular na rede de dados ao contrário do acesso remoto.

Sendo assim, o objetivo do teste foi identificar a diferença entre o tempo de inicialização para o acesso a solução por meio de um Client-Core remoto e um local. Para tanto, o seguinte cenário foi reproduzido:

- Foi preparado um ambiente para acesso ao Client-Core remoto e outro para acesso local;
- Foram adicionados timestamps na interface que mostram o tempo global ao requisitar uma conexão e ao finalizar o carregamento das informações iniciais, além de mostrar a diferença entre as duas marcações;
- Um mesmo usuário efetuou o login com o Client-Core remoto e local para não haver diferença no número de informações a serem carregadas.

O resultado do teste é mostrado na Figura 6.3.

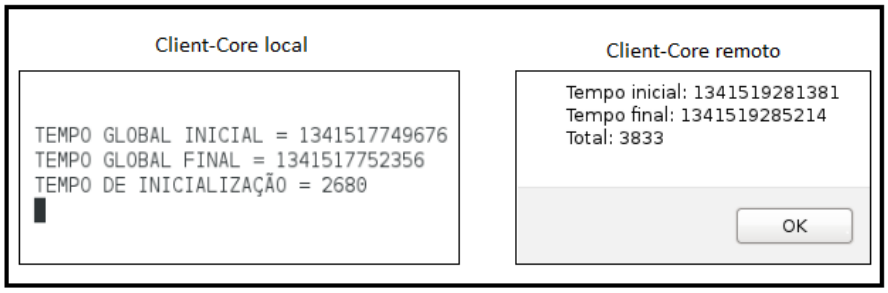

Figura 6.3 - Tempo de inicialização de Client-Core local e remoto em milissegundos.

Observando a Figura 6.3 é possível notar que a inicialização remota demorou pouco mais de 1 segundo se comparada com a conexão local, o que ainda garante boa usabilidade dos dois tipos de acesso.

Certamente, o tempo de inicialização baixo para os dois casos se deu pelo fato de tanto o Servidor UC quanto as interfaces de acesso estarem na mesma rede. O acesso ao sistema por uma rede externa pode implicar um tempo de inicialização maior, entretanto, no atual estágio de desenvolvimento, não foi possível realizar esse teste e mensurar esse tipo de dado.

6.2.2: Carregamento de informações de contatos e envio de mensagens

O teste de carregamento de informações e envio de mensagens também está relacionado à inicialização de um acesso ao sistema de UC, ou seja, o teste visa medir o tempo necessário para carregar as informações de um usuário e seus contatos e o envio correto de mensagens levando em consideração o número de contatos que ele possui.

Esse teste é necessário para verificar se será preciso limitar o número de contatos que um usuário pode possuir, ou se essa informação poderá ser indiferente no sistema. Para tanto, o seguinte cenário foi proposto:

- Foi implementado um pequeno software responsável por adicionar um número especificado de contatos a um usuário.

- Foram realizadas escalas de testes com 0, 10, 25, 100, 250, 500 e 800 contatos adicionados à lista de um usuário.
- Foi medido o tempo de inicialização de cada caso para carregar as informações dos contatos do usuário por um acesso remoto ao Client-Core.

O resultado do teste é apresentado na Figura 6.4, a qual mostra um gráfico do tempo de inicialização de acordo com a quantidade de contatos que o usuário possui.

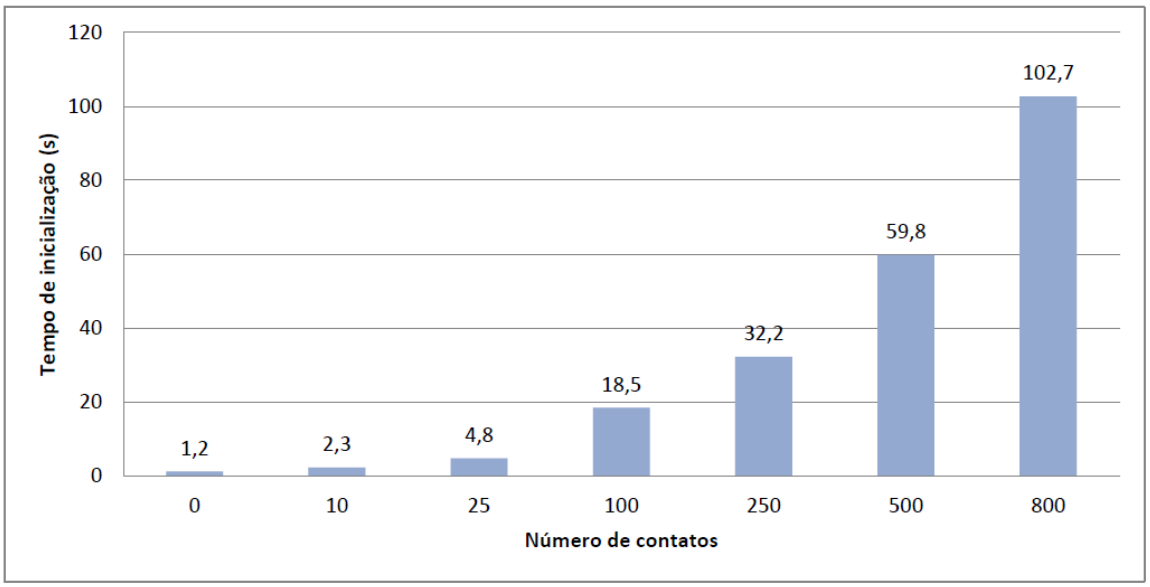

Figura 6.4 - Tempo de inicialização em relação número de contatos.

Pelo gráfico apresentado é possível perceber que o tempo para carregar as informações ao inicializar uma sessão cresce consideravelmente com o aumento do número de contatos, o que pode indicar a necessidade de uma limitação no número de contatos, visto que o tempo para carregar informações de um usuário com cerca de 800 contatos ultrapassa a marca de um minuto e meio.

Como proposta a corrigir esse problema, pode-se considerar uma modificação no modo como o Client-Core envia as mensagens à interface, ou seja, as mensagens enviadas podem ser agrupadas em pacotes maiores para que haja um menor fluxo de dados na rede, o que poderia diminuir consideravelmente esse tempo de inicialização. Entretanto, até a escrita desse trabalho, não foi possível fazer essas modificações e testá-las para quantificar o ganho de tempo.

Ainda nesse teste se verificou problemas com o envio de algumas mensagens, pois os seus tamanhos eram demasiadamente grandes a ponto de serem fragmentadas durante o envio. Para corrigir esse problema, foi implementado, na interface, um mecanismo para unir possíveis mensagens fragmentadas, corrigindo essa adversidade.

6.2.3: Memória alocada para instâncias do Client-Core

O último, e talvez mais importante teste a ser feito com o Client-Core é o de alocação de memória. Com esse teste se pode mensurar quanto em média uma instância desse software irá ocupar de memória dentro do Servidor UC, permitindo que se chegue a uma relação entre número de clientes conectados à solução simultaneamente e o hardware necessário no Servidor UC.

Para realizar o teste o seguinte procedimento foi tomado:

- Foram instanciados 30 processos do Client-Core dentro de uma máquina;
- Por meio de uma ferramenta de análise de gerenciamento de memória do sistema operacional Linux foi obtida uma média de quanto cada instância do Client-Core ocupa de memória a partir dos trinta processos instanciados;
- Em seguida foi feito um levantamento de ocupação de memória relacionado ao número de contatos que um usuário possui, com uma escala de 0, 10, 25, 100, 250, 500, 800 contatos adicionados à sua lista.

A Figura 6.5 mostra o quanto cada uma das 30 instâncias do Client-Core ocupou da memória física (coluna destacada da imagem) do computador utilizado no teste.

Através da Figura 6.5 se pode perceber que em média cada instância do Client-Core ocupa 25,2 Mb de memória, o que pode ser um limitador bastante forte para a utilização do Client-Core na solução.

Com o propósito de melhorar esse aspecto foi realizada uma varredura no código do software desenvolvido com o objetivo de encontrar pontos de possível overhead na alocação de memória. Nessa análise foram encontradas algumas

seções nas quais o código poderia ser refinado, aplicando-lhe técnicas para melhor gerenciamento de memória em Java. Entretanto, mesmo com diversos refinamentos no código, a ocupação de memória teve uma melhora pouco expressiva.

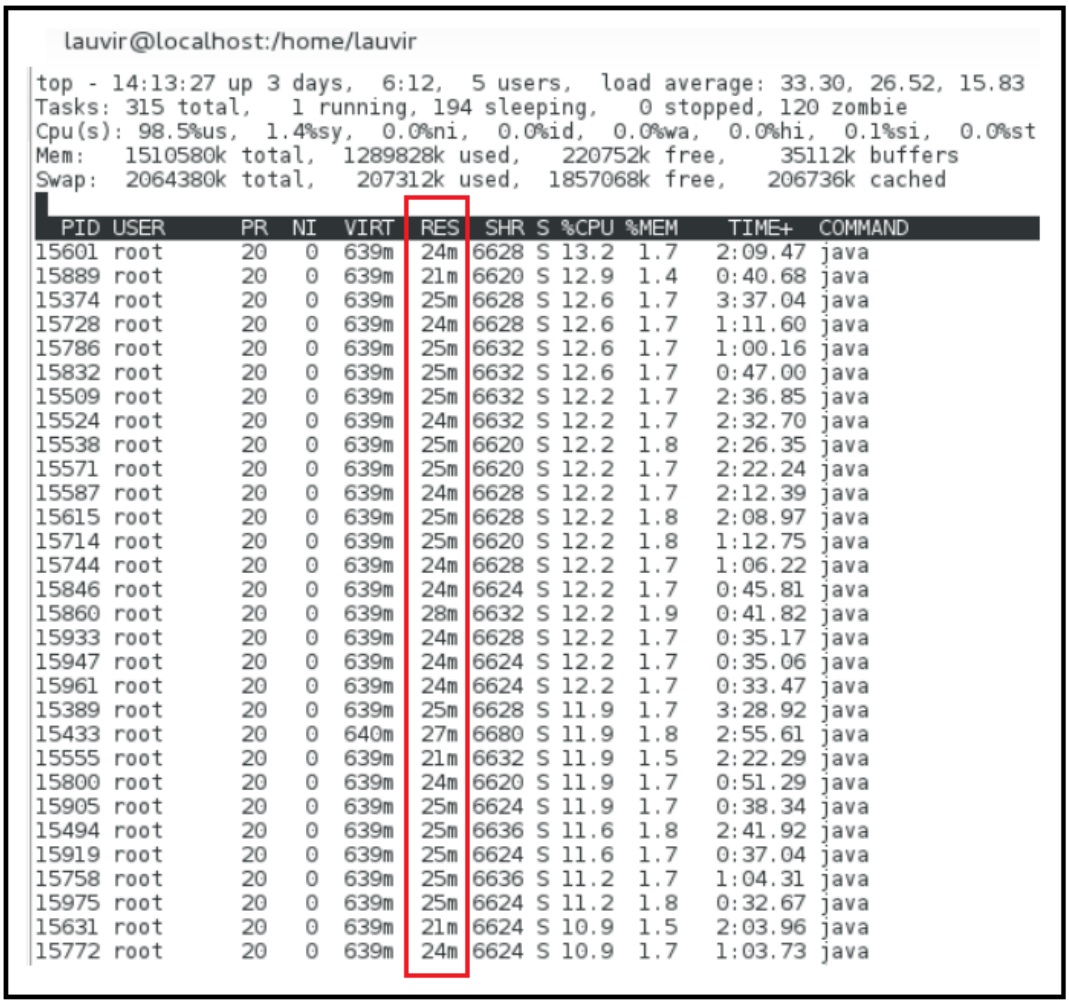

Figura 6.5 - Memória física ocupada por instâncias do Client-Core em megabytes.

Dessa forma, após pesquisas e testes de ocupação de memória de outros softwares desenvolvidos em Java, constatou-se que boa parte do overhead gerado se deve à própria máquina virtual de Java (Java Virtual Machine - JVM) que ocupa uma quantidade significativa de memória. Porém, outra boa parte da ocupação da memória se deve ao fato de o Client-Core precisar armazenar uma grande quantidade de informações sobre os contatos do usuário em variáveis contidas dentro do próprio código, o que leva à segunda parte do teste: a alocação de

memória relacionada ao número de contatos que o usuário possui, com resultado mostrado na Figura 6.6.

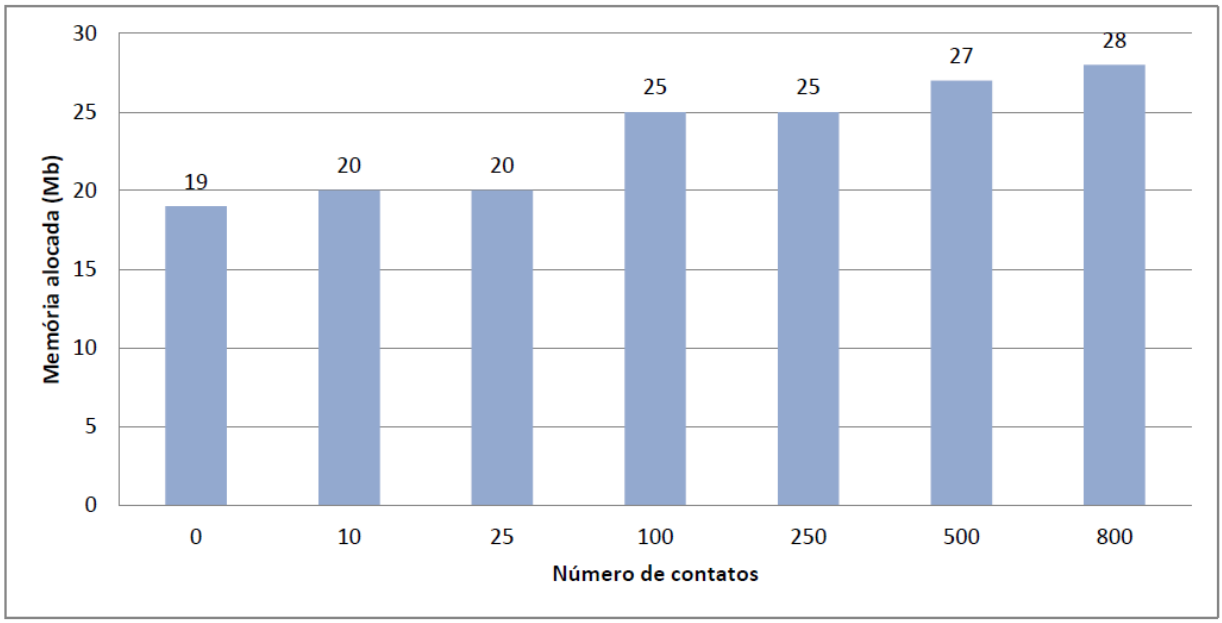

Figura 6.6 - Alocação de memória relacionada ao número de contatos do usuário.

Pela Figura 6.6 se pode ver que a memória cresce ao se adicionar muitos contatos, pois o Client-Core precisa reter muitas informações durante sua execução. Entretanto, esse aumento não é linear, e não cresce muito com o aumento de muitos contatos, evidenciando mais uma vez que o processo de alocação de memória da JVM tem um overhead bastante grande.

Esse comportamento de alto consumo de memória pode inviabilizar o uso do Client-Core dentro do Servidor UC, pois com a conexão de muitos usuários, o sistema poderia apresentar problemas como lentidão na execução de tarefas, falta de memória para alocar e em casos mais extremos a interrupção do funcionamento do servidor.

No entanto, como tentativa de solucionar esse problema podem ser propostas três alternativas:

- Remodelar o Client-Core para uma nova versão em que as conexões dos clientes seriam todas tratadas por um único Client-Core a partir de threads que representariam cada sessão;
- Trocar o armazenamento de informações no Client-Core por um sistema que busque os dados necessários no Servidor UC sob requisições enviadas pela interface.
- Aproveitar a arquitetura distribuída do sistema e adicionar máquinas especialmente para executar instâncias do Client-Core.

Fazendo uma análise das alternativas apresentadas, pode-se afirmar que a primeira é a que possivelmente apresentará os melhores resultados. Ela já vem sendo adotada para uma nova versão do Client-Core e apresentando boa performance. Entretanto, essa solução do problema necessitará de um prazo maior para ser realizada.

A segunda alternativa resolveria parte do problema de alocação de memória, entretanto, poderia trazer outros problemas como um maior tráfego de dados na rede e menor agilidade para a obtenção de informações, pois o usuário precisaria disparar eventos ao sistema toda vez que necessitasse obter algum dado.

A última das alternativas pode ser a mais plausível pensando-se em um curto prazo, visto que o sistema foi desenvolvido com o objetivo de ser distribuído e totalmente escalável. Obviamente, essa alternativa não resolve o problema de alocação de memória, mas a partilha em várias estações, o que é uma característica desejável dentro de um sistema distribuído.

Entretanto, ao se adotar a solução de distribuir o Client-Core em várias máquinas, faz-se necessário dimensionar o número de instâncias possíveis a partir do hardware de cada máquina.

Os números da Figura 6.5 mostram que para instanciar 1000 processos do Client-Core seriam necessários 25Gb de memória, o que é inviável para uma máquina comum. Normalmente servidores possuem melhores recursos de hardware, entretanto, a instanciação de 1000 processos é uma tarefa bastante crítica para somente uma máquina.

A Tabela 4 mostra alternativas de distribuição de 1000 instâncias do Client-Core em um número específico de máquinas.

Tabela 4 - Processo do Client-Core distribuídos em várias máquinas.

Client-Core instanciados por máquina	Quantidade de máquinas necessárias	Quantidade de memória por máquina (Gb)
100	10	2,5
200	5	5,0
250	4	6,2
333	3	8,3
500	2	12,5

A partir da Tabela 4 é possível perceber que, para recursos medianos de hardware (em torno de 6 a 8 Gb de memória), seriam necessárias cerca de três ou quatro máquinas para instanciar 1000 processos.

Com os resultados desse teste somados aos demais propostos ao Client-Core, pode-se afirmar que ele atinge o principal objetivo de ligar os usuários à solução de UC, porém possui algumas limitações que não impedem sua utilização, mas podem deixar o sistema com um custo mais elevado se for considerada a alternativa de distribuição dos processos do Client-Core em diversas máquinas.

Realizados os testes com os dois softwares implementados e confirmando suas validações ao sistema de UC, pode-se então partir para as considerações finais levantadas sobre o trabalho.

Capítulo 7: Conclusões e perspectivas

O sistema de UC apresentado pode ser considerado uma ferramenta muito importante para empresas em um futuro próximo, visto que os resultados alcançados com a implantação de sistemas como esse são consideravelmente positivos, ou seja, a empresa passa a otimizar o tempo de seus funcionários em suas jornadas de trabalho, além de conseguir contatar clientes e fechar negócios com maior rapidez.

Os módulos do cliente de acesso apresentados cumpriram seus papéis dentro do sistema como um todo, pois cada um deles alcançou seus requisitos pré-estabelecidos, mesmo com algumas limitações que foram ou serão corrigidas.

O Client-d, por necessitar gerenciar múltiplas conexões, apresentou-se um software bastante robusto e com desempenho elevado em relação à velocidade com que trata as requisições vindas de outros módulos da solução. Apesar de ele poder realizar somente cinco tipos de rotinas, cada uma delas é de fundamental importância para que o sistema funcione. Certamente, as proteções para possíveis falhas nessas rotinas permitem que o software não seja interrompido ou seu funcionamento seja comprometido, garantindo a confiança necessária para que o Client-d pudesse ser inserido no sistema de UC.

Já o Client-Core, mesmo tendo um consumo de memória razoavelmente elevado, atingiu sua meta de se integrar à solução de forma flexível, podendo ser executado local ou remotamente. Além disso, esse módulo da solução será decisivo para o bom funcionamento do sistema, isso porque é ele que possui a versatilidade de ser executado em qualquer dispositivo e até mesmo dentro do Servidor UC, ou seja, será necessário mensurar quantidades de instâncias do Client-Core que poderão ser executadas de acordo com as especificações do servidor para que não haja uma sobrecarga de memória e processamento. Certamente, com a segunda versão do Client-Core baseada em threads, problemas dessa espécie deixarão de existir, melhorando a usabilidade do software.

Há de se considerar também que os protocolos de comunicação utilizados na solução foram bastante decisivos para que um sistema que exige alto fluxo de troca

de mensagens conseguisse gerenciar todos os pacotes que transitam entre seus módulos.

Tendo em vista essas observações relacionadas aos softwares desenvolvidos, pode-se considerar que a meta geral do trabalho de integrar o cliente ao sistema de comunicações unificadas foi concluída de forma bem sucedida.

Entretanto, ainda se faz pertinente comentar os aspectos pessoais envolvidos no desenvolvimento desse trabalho, isso porque houve um ganho profissional em relação ao conhecimento e convivência em um ambiente corporativo, além de se intensificar e aprimorar conhecimentos novos e já adquiridos durante toda a graduação.

Como perspectiva para trabalhos futuros, é esperada a integração de outros recursos e mídias no sistema de UC apresentado, como videoconferência, para que a solução consiga atingir um número maior de dispositivos de comunicação.

Além disso, espera-se também que o sistema passe a contar com mais módulos de gerenciamento de conexões e servidores atuando de forma integrada como um único sistema, o que provaria a escalabilidade da solução.

Bibliografia

[1] P. H. Gregory, “Comunicação Unificada para Dummies”. Wiley, 2008.

[2] “5 Things That Waste Your Time at Work”. Disponível em: <http://mashable.com/2012/04/13/wasting-time-work/>. Acesso em maio de 2012.

[3] “Comunicações Unificadas”. Pacto Telecom. Disponível em: <http://pactotelecom.com/home/comunicacoes_unificadas.php>. Acesso em maio de 2012.

[4] F. Wagner e M.C. da Silva, “Comunicação Unificada – Um estudo de base para análise dos seus benefícios para as empresas”. Monografia de Graduação - Curso de Engenharia Elétrica Telemática. Universidade do Sul de Santa Catarina, 2011.

[5] M. Parker, “A Short History of UC”. Disponível em: <http://www.ucstrategies.com/unified-communications-strategies-views/a-short-history-of-uc.aspx>. Acesso em maio de 2012.

[6] “Brief History of Unified Communications”. Disponível em: <http://www.australianscience.com.au/technology/brief-history-of-unified-communications/>. Acesso em maio de 2012.

[7] Y.N.A Singh, “Como as Comunicações Unificadas Estão Sendo Aplicadas nas Empresas”. Disponível em: <http://www.slideshare.net/Ishtecnologia/como-as-comunicaes-unificadas-esto-sendo-usadas-nas-empresas>. Acesso em maio de 2012.

[8] C. Herbert, “The Good, the Bad and the Ugly of Unified Communications”. Disponível em: <http://blog.allstream.com/the-good-the-bad-and-the-ugly-of-unified-communications/>. Acesso em maio de 2012.

[9] J. Skorupa, D. Curtis e L. Orans, “Prepare sua rede para comunicação unificada”. Disponível em: <http://info.abril.com.br/noticias/corporate/gartner/prepare-sua-rede-. shl>. Acesso em maio de 2012.

[10] L.G. Harbaugh, “Cinco razões para adotar comunicação unificada”. Disponível em: <http://computerworld.uol.com.br/tecnologia/2011/10/06/cinco-razoes-para-adotar-a-comunicacao-unificada/>. Acesso em maio de 2012.

[11] “Top Five Benefits of Unified Communications”. Disponível em: <http://www.telephonyworld.com/basics/article/top-five-benefits-of-unified-communications/>. Acesso em maiode 2012.

[12] C. Johnson, “Exclusive Unified Communications Survey Results”. Disponível em: <http://www.cio.com/article/503900/Exclusive_Unified_Communications_Survey_Results>. Acesso em maio de 2012.

[13] B. Hafner, “The Unified Communications Scenario”. GARTNER, 2008.

[14] E.M. de Araújo, “Arquitetura - UC”. Intelbras, 2011.

[15] “Diferença entre VoIP e Telefonia IP”. Disponível em: <http://tecnologia-voip.blogspot.com.br/2010/11/diferenca-entre-voip-e-telefonia-ip.html>. Acesso em aio de 2012.

[16] “Diferenças entre VoIP e Telefonia IP”. Disponível em: <http://www.internext.com.br/index.php?id=128&option=com_content&task=view>. Acesso em maio de 2012.

[17] E.M. de Araújo, “Árvore de Funções do Cliente da Solução UC Intelbras”. Intelbras 2012.

[18] Futurecom Rio de Janeiro – 2012. Disponível em: <http://www.futurecom.com.br/2012/local/>. Acesso em junho de 2012.

[19] P. Saint-Andre, K. Smith e R. Troçon, “XMPP: The Definitive Guide ”. O’Reilly, 2009.

[20] B.A.C Souza, “XMPP - Extensible Messaging and Presence Protocol”. Monografia Graduação – urso de Sistemas de Informação, Faculdade Anhanguera, Belo Horizonte, 2011.

[21] D.S. Mendes, “Criação de um cliente para protocolo PubSub/XMPP”. Monografia Graduação – Curso de Ciências da Computação, Universidade Luterana do Brasil, Gravataí, 2010.

[22] E.M. de Araújo, “Integração das plataformas ao sistema de UC”. Intelbras, 2011.

[23] E.M. de Araújo, “LibIntUC”. Intelbras 2011.

[24] “XEP-0060: Publish-Subscribe”. XMPP Standards Foundation. Disponível em: < http://xmpp.org/extensions/xep-0060.html>. Acessado em maio de 2012.

[25] “Smack API 3.2.2”. Jive Software. Disponível em:<http://www.igniterealtime.org/projects/smack/index.jsp>. Acesso em junho de 2012.

[26] “Introducting JSON”. Dispomível em: <http://json.org>. Acesso em junho de 2012.