Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Bacharelado em Computação

Disciplina: Redes de Computadores

Prof: Miguel Franklin

Trabalho: CAPIM - *Centralized APplication of Instant Messaging*

Relatório: Projeto e Implementação

Integrantes da Equipe

Matrícula	**Nome do (a) aluno (a)**
286740	Murilo Lima de Holanda
288809	Paulo Sérgio Vasconcelos Alves Júnior
286750	Rafael de Lima

Relatório elaborado em abril de 2009.

SUMÁRIO

I.INTRODUÇÃO.....3
II.DESCRIÇÃO E FUNCIONALIDADES DO SISTEMA.....7
III.PROTOCOLO DE TRANSMISSÃO CONFIÁVEL DE DADOS.....7
IV.MODELAGEM DO PROTOCOLO DE INSTANT MESSENGER.....7
V.MODO DE USO.....7
VI.CONCLUSÃO.....9

I.INTRODUÇÃO

O trabalho consiste em modelar e implementar uma aplicação de mensagem instantânea baseada em sockets UDP. Para tanto, abaixo temos a lista com os requisitos básicos para o mesmo:

1. Conversação de Texto de 1 para 1 ou de N para N: Um usuário pode conversar ao mesmo tempo com uma só pessoa ou com várias (conferência).

2. A aplicação será toda baseada em *sockets* UDP: Toda a comunicação entre clientes e servidor deverá ser realizada através de UDP. Não haverá comunicação direta entre dois usuários, isto é, o servidor deverá centralizar todas as operações.

3. O serviço será confiável: Nenhuma mensagem, seja de conversação ou de sinalização, deverá se perder. Deve-se implementar o serviço de transmissão de mensagens confiável, que pode ser baseado em retransmissões. Esse protocolo deverá ser devidamente documentado no relatório de implementação, incluindo descrição detalhada e diagramas de sequência das mensagens.

4. Sistema de *Keepalive:* O servidor deverá se manter atualizado sobre o estado de cada usuário, através de mensagens *keepalive* que os usuários devem mandar periodicamente para o servidor.

5. Controle de amigos: Um usuário cadastrado pode adicionar e remover outros usuários. Quando da adição de um usuário, o mesmo deverá aceitar ou não a sua adição na lista de alguém.

6. Controle de status dos usuários: O cliente deverá ser dotado de um timer de inatividade, que sinalizará status de "ausente" ou "ausente prolongado" para o servidor, que por sua vez deverá sinalizar o status aos amigos do usuário.

7. Autenticação de usuário: O usuário, para se conectar ao sistema, deverá se autenticar através de nome de usuário e senha. O uso de criptografia para senha é opcional. Da mesma forma que o usuário entra no servidor, ele poderá sinalizar sua saída. A saída pode ser amigável (sinalizada pelo usuário) ou brusca (por falta de *keepalive*).

8. Controle de presença: Se um usuário entrar (autenticar) no servidor, todos os amigos registrados desse usuário deverão ser comunicados de sua entrada. Da mesma forma, se o usuário sair ou se um tempo de *timeout* expirar (falta de *keepalive*), o servidor deverá enviar atualização de status a todos os amigos.

9. Interface gráfica no lado cliente: Os usuários deverão ter uma interface gráfica para interagir com o sistema.

O sistema consiste de duas partes principais:

1.Um servidor sempre disponível que faz o intercâmbio de mensagens entre clientes e fornece o serviço.

2.Uma aplicação cliente, dotada de interface gráfica, capaz de se comunicar com o servidor.

I.DESCRIÇÃO DAS FUNCIONALIDADES DO SISTEMA

Por ser um aplicação que segue o modelo Cliente-Servidor, os clientes dependem completamente do servidor para usufruírem do serviço de mensagem instantânea oferecido pelo CAPIM.

1. Banco de dados

No servidor, há um banco de dados com as seguintes tabelas:

Tabela 'usuarios':

Tabela que guarda informações atuais dos usuários do sistema

user_login	user_pass	user_ip	user_port	user_status

Tabela 'contatos':

Tabela que guarda as ligações entre os usuários do sistema

name1	name2

Tabela 'requisicoes':

Tabela que guarda requisições de novo contato pendentes entre os usuários

name2	name1

2. Mensagens

As mensagens enviadas pela aplicação cliente ao servidor têm a seguinte semântica:

<código>	%%	*<campo1>*	%%	...	%%	*<campoN>*	%%

código - Código da mensagem a ser enviada.
%% - Caracteres especiais para diferenciação de campo.
user_login - Login do usuário a ser cadastrado no sistema.
user_pass - Senha do usuário a ser cadastrado no sistema.

Grande parte das mensagens enviadas pelo servidor são apenas respostas de OK ou ERRO às mensagens enviadas pela aplicação cliente. As mensagens de OK e de ERRO são as seguintes, respectivamente:

<codigo>	%%	OK	%%

Mensagem OK

<codigo>	%%	ERR	%%

Mensagem de erro

As outras mensagens enviadas pelo servidor ao cliente tendem a ser diferentes, dependendo do tipo de iteração. As iterações entre o cliente e o servidor são:

a)Cadastro:

O usuário deve se cadastrar no sistema para ter acesso ao mesmo. O cadastro é feito através de um nome de usuário *user_login* e uma senha *user_pass*. O *user_login* é a chave

primária de comunicação entre o servidor e o cliente, sendo o atributo que identifica o usuário dentro do sistema.

A mensagem de requisição de cadastro do cliente para o servidor é a seguinte, sendo 1 o código para cadastro:

1	%%	*<user_login>*	%%	*<user_pass>*	%%

Para que o cadastro seja feito com sucesso o servidor verifica a existência de um *user_login* igual no sistema, emitindo mensagens de erros em caso de resposta positiva, mensagens estas que são tratadas pela aplicação cliente, que avisa ao usuário da indisponibilidade do *user_login* escolhido por ele. Em caso de mensagem negativa, ou seja, o *user_login* poderá ser usado, o cadastro é feito e o usuário é avisado do sucesso da operação através de uma mensagem resposta OK.

b)Login:

Quando o usuário de se loga no sistema, há a verificação de usuário e senha dele. Caso *user_login* exista e a *user_pass* esteja correta, o usuário pode logar no sistema. Neste momento vários eventos acontecem:

1- É iniciada uma thread de *keepalive* para este usuário. Isso vai ser explicado mais à frente, mas pode-se adiantar que as listas de amigos online, com seus respectivos status, e de offline são carregadas.

2- Todos os usuários online que são amigos do usuário recentemente logado no sistema recebem uma notificação de que isto aconteceu. Este processo também é explicado a seguir.

3- A tela principal do usuário é carregada.

A mensagem de requisição de login do cliente para o servidor é a seguinte, sendo 0 o código para login:

0	%%	*<user_login>*	%%	*<user_pass>*	%%

Caso haja algum erro como usuário inexistente ou senha errada, uma mensagem de erro(ERR) com código 0 é enviada para a aplicação cliente, que trata esse erro e o informa ao usuário. Quando o usuário loga-se com sucesso uma mensagem de OK com código 0 é enviada à aplicação cliente, que permite esta ação ao usuário.

c)Keepalive:

d)Adição de contato:

Usuários podem adicionar outros usuários para sua lista de contatos. Isto é feito mandando uma mensagem de requisição, de código 5, para o servidor:

5	%%	*<nome do requisitante>*	%%	*<nome do contato>*	%%

Recebendo esta mensagem o servidor adiciona uma requisição do usuário para o contato a ser adicionado. No próximo *keepalive* do usuário a ser adicionado ele receberá a requisição e a responderá com uma mensagem de código 3, indicando a aceitação, ou não, do novo contato:

3	%%	*<user_login>*	%%	*<nome do contato>*	%%	<resposta>	%%

Neste momento a requisição é deletada da tabela, indicando que o usuário já a recebeu, emitindo a resposta. Se a resposta for SIM, um contato é feito entre os usuários. Uma ligação indo de cada usuário para o outro é feita na tabela de contatos. Caso a resposta seja NAO, o contato não é feito.

e)Remoção de contato:

Usuários também podem remover usuários da sua lista de contatos. Isto é feito mandando uma mensagem de remoção, de código 8, para o servidor:

8	%%	*<nome do requisitante>*	%%	*<contato a excluir>*	%%

Recebendo esta mensagem o servidor remove a relação ente os dois usuários.

f)Alteração de Status:

Eventualmente o usuário necessita modificar o seu status, seja por necessidade, em caso de saída, ou por preferência, escolhendo ausente ou ocupado. Isso é feito através de uma mensagem de código 4, onde a aplicação cliente envia o novo status do usuário:

4	%%	*<user_login>*	%%	*<status>*	%%

O servidor altera o status do usuário e envia uma mensagem de resposta OK para o cliente.

g)Aviso de status

Quando o usuário se loga, ele deseja que todos os seus amigos sejam informados de que ele está online. Isto é feito através de avisos que são enviados para todos os amigos.

A mensagem utilizada pela aplicação cliente para pedir o aviso é:

7	%%	*<user_login>*	%%

A mensagem enviada pra cada amigo do usuário *user_login* é igual a de cima, sendo que o cliente a interpreta, informando ao usuário que usuário de login *user_login* está online.

h)Troca de mensagens entre usuários:

I.PROTOCOLO DE TRASMISSÃO CONFIÁVEL DE DADOS

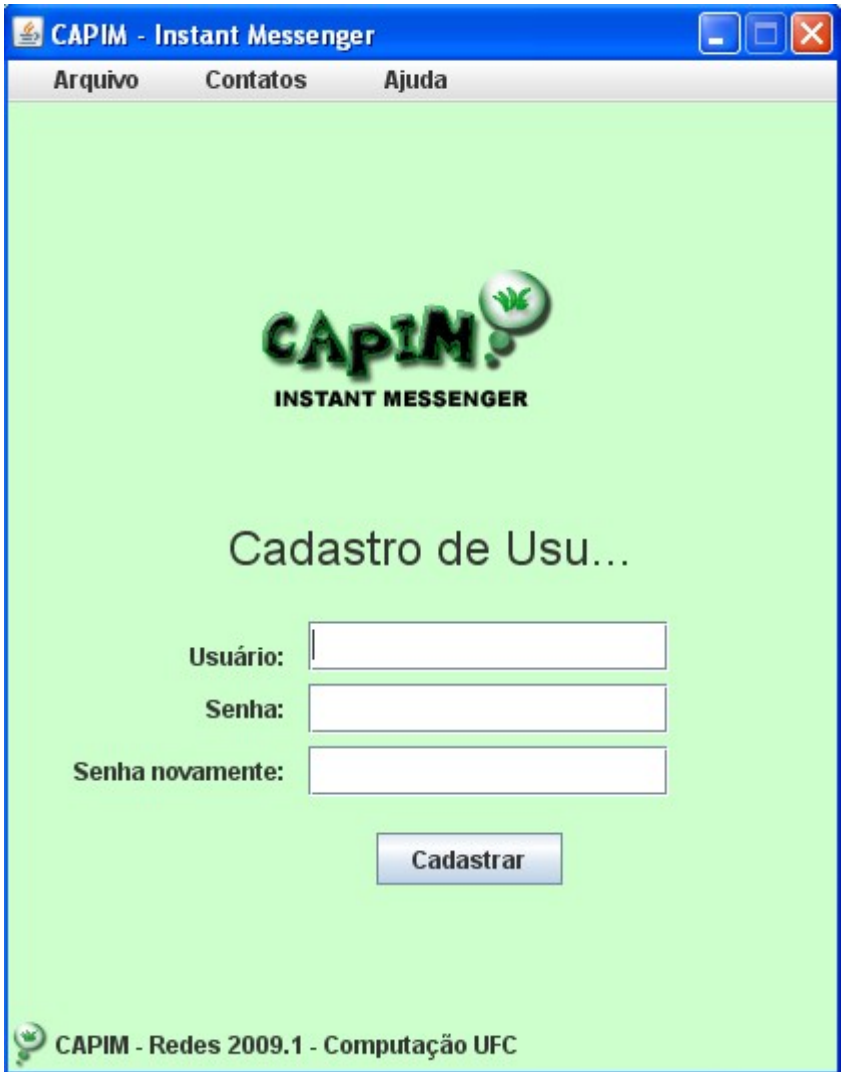
Não pudemos implementar o protocolo de transmissão confiável de dados não foi implementado no prazo.

II.MODELAGEM DO PROTOCOLO DE INSTANT MESSENGER

A modelagem também não foi realizada.

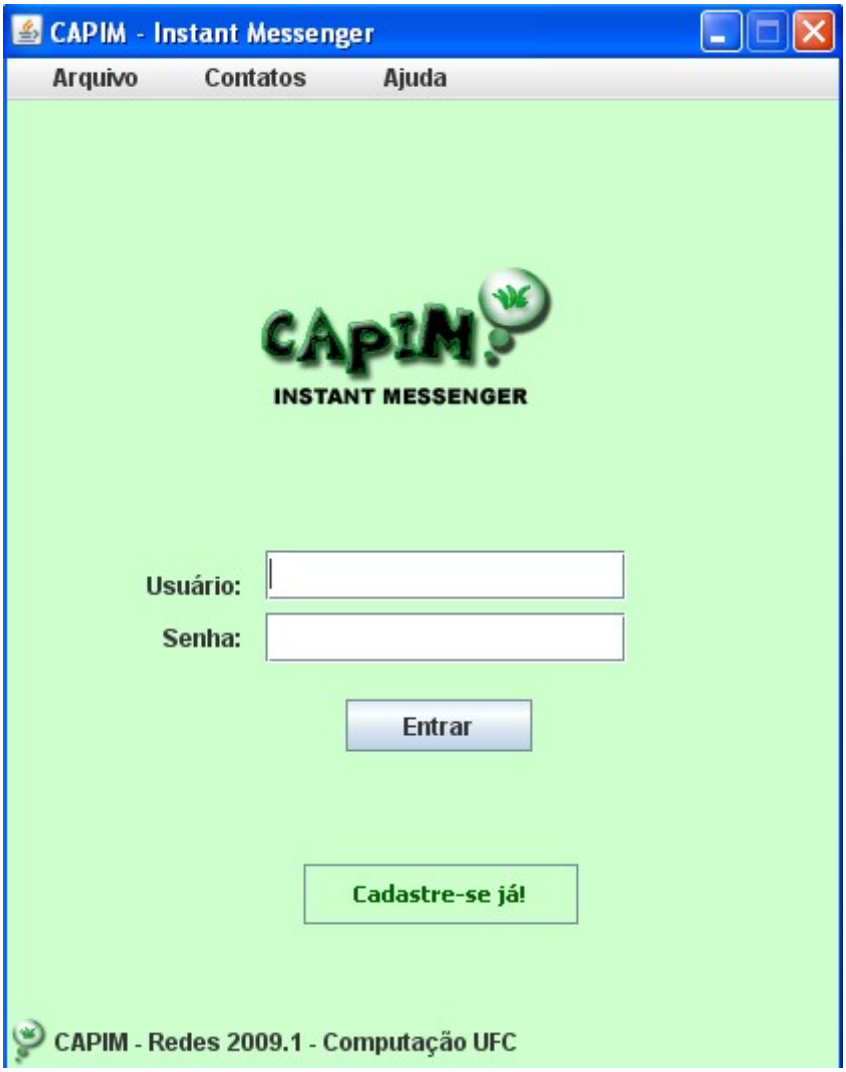
III.MODO DE USO

Primeiramente, o usuário deve efetuar um cadastro, onde ele apenas informa um nome de usuário e uma senha :



Tela de cadastro.

Após se cadastrar o usuário pode acessar o sistema através de seu nome de usuário e de sua senha, pela tela de login:



Tela de Login.

Ao se logar, é mostrada a tela principal do usuário na aplicação, onde ele pode ver o status de seus amigos e efetuar várias ações.

Neste momento todos os seus amigos que estão online são informados de seu status. O cliente também recebe as requisições de novos contatos e pode tanto abrir janelas de conversa com seus amigos como receber mensagens dos mesmos através de janelas que pop-up.

O cliente também pode adicionar ou remover amigos de sua lista de contatos. Quando quer adicionar um contato, o usuário envia uma requisição para o mesmo. A relação entre os contatos só é feita quando o último a aceita.

Quando quer iniciar uma conversa com um de seus contatos online o usuário clica no nome correspondente e uma janela de conversação é aberta.

IV.CONCLUSÃO

Ao final do trabalho, percebemos que o mesmo não saiu como esperávamos, mas mesmo com todos os problemas surgidos, o mais importante é ter exercitado as noções referentes a Linguagem Java, Multithread, Protocolos de transmissão confiáveis entre outros conceitos da disciplina de Redes de Computadores. Fica a experiência da tentativa de realização de um grande trabalho, sem obter um grande êxito.

O foco maior do trabalho foi a implementação, o que talvez tenha sido um dos maiores problemas, por não ter sido feita uma modelagem e análise das etapas do mesmo. Acredita-se que a aplicação dos conceitos de Engenharia de Software poderia melhorar o rendimento da equipe.