Índice

Sumário

[Esquema Cliente-Servidor 3](#_Toc232813359)

[Cliente 3](#_Toc232813360)

[Requisição de conexão 3](#_Toc232813361)

[Requisição da lista de clientes 4](#_Toc232813362)

[Requisição de desconexão 5](#_Toc232813363)

[Servidor 6](#_Toc232813364)

[Estabelecendo a conexão 6](#_Toc232813365)

[Fornecendo lista de clientes 6](#_Toc232813366)

[Estabelecendo desconexão 6](#_Toc232813367)

[Serviço de Criptografia 6](#_Toc232813368)

[Usando o Chat 6](#_Toc232813369)

[Comunicação TCP 6](#_Toc232813370)

[Threads 6](#_Toc232813371)

[Transferindo Arquivo 7](#_Toc232813372)

[Características da Aplicação 7](#_Toc232813373)

[Estrutura do pacote 7](#_Toc232813374)

[Estrutura do ACK 8](#_Toc232813375)

[Lado remetente 8](#_Toc232813376)

[Segmentação do Arquivo 8](#_Toc232813377)

[Buffer de Transmissão 9](#_Toc232813378)

[Lado destinatário 10](#_Toc232813379)

[Buffer de Recepção 11](#_Toc232813380)

[Remontador do Arquivo 11](#_Toc232813381)

[Confiabilidade 12](#_Toc232813382)

[Controle de Congestionamento 17](#_Toc232813383)

[Arquitetura P2P 18](#_Toc232813384)

[Diagrama de Classes UML 20](#_Toc232813385)

Índice de figuras

[Figura 1 Tela inicial do chat 3](#_Toc232813386)

[Figura 2 Tela com Informação dos usuários on-line. 4](#_Toc232813387)

[Figura 3 Tela com informação de desconexão de um usuário. 4](#_Toc232813388)

[Figura 4 Janela de seleção do arquivo a ser enviado 6](#_Toc232813389)

[Figura 5 Estrutura do Pacote 7](#_Toc232813390)

[Figura 6 Ilustração de uma Segmentação 8](#_Toc232813391)

[Figura 7 Formato do Buffer de Transmissão 9](#_Toc232813392)

[Figura 13 Buffer de Recepção 10](#_Toc232813393)

[Figura 14 Remontagem de um Arquivo 11](#_Toc232813394)

[Figura 8 Primeira situação: ocorreu perda de pacote e o decorrente esgotamento do temporizador. 13](#_Toc232813395)

[Figura 9 Segunda situação: ocorreu perda de pacote e a decorrente chegada de ACKs duplicados. 14](#_Toc232813396)

[Figura 10 Terceira situação: o destinatário recebeu um pacote com número de sequência maior que o esperado. 14](#_Toc232813397)

[Figura 11 Máquina de estados finita do Remetente 15](#_Toc232813398)

[Figura 12 Máquina de estados finita do destinatário 16](#_Toc232813399)

Esquema Cliente-Servidor

# Cliente

## Requisição de conexão

O cliente, através da interface gráfica, faz uma requisição de conexão para o servidor enviando uma mensagem TCP com o tipo de requisição para conexão e o login e senha em seguida. A segurança do usuário será garantida através do serviço de criptografia oferecido.



Figura Tela inicial do chat

## Requisição da lista de clientes

Para utilizar o chat e enviar arquivos, o usuário precisa saber quais usuários estão on-line. Para obter uma lista de usuários on-line ele manda uma mensagem de controle TCP para o servidor requisitando tal lista. Ele precisa manter essa lista sempre atualizada para tanto o uso do chat quanto o envio de arquivos para os outros usuários.

## printChatAbaLateral.jpg

Figura Tela com Informação dos usuários on-line.

## Requisição de desconexão

Da mesma maneira que a conexão, o cliente manda uma mensagem TCP com tipo de requisição para desconexão em seguida o login.

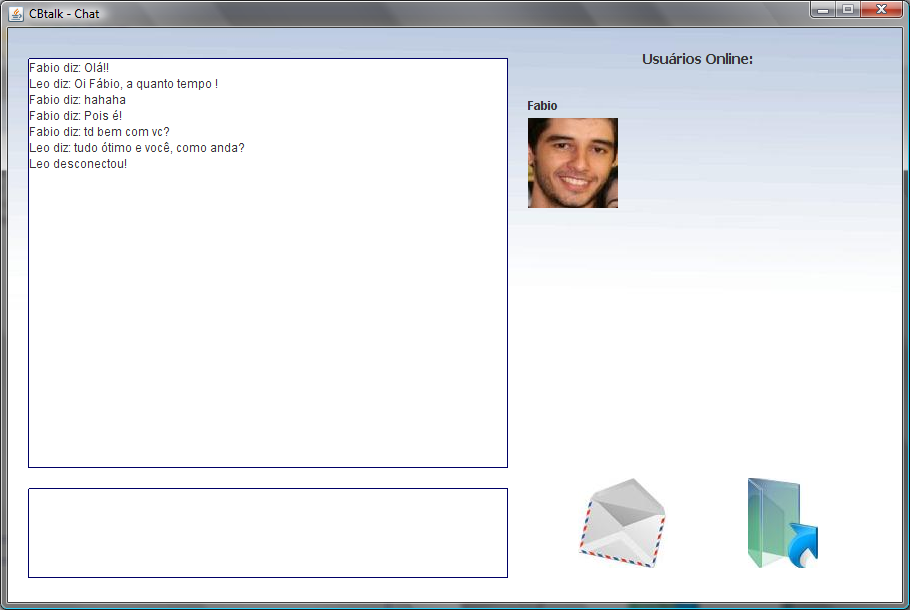


Figura Tela com informação de desconexão de um usuário.

# Servidor

## Estabelecendo a conexão

Ao receber uma mensagem TCP do tipo conexão de algum cliente, o servidor irá checar em seu banco de dados (arquivo), se o login e senha estão corretos para em seguida confirmar o status de conectado ao cliente. Caso o login ou senha estejam incorretos, o servidor enviará uma mensagem TCP sinalizando o erro.

## Fornecendo lista de clientes

Depois que o usuário conecta-se ao servidor, o servidor receberá mensagens TCP do tipo status, que será uma requisição da lista de clientes on-line que ele possui no sistema.

## Estabelecendo desconexão

Ao receber uma mensagem TCP do tipo desconexão, o servidor buscará o login do usuário que requisitou a desconexão e mudará seu status na lista de clientes para off-line e fechará a porta de conexão.

## Serviço de Criptografia

Para fornecer uma maior segurança para o usuário, foi utilizado um serviço de criptografia no envio da senha do cliente para a autenticação do servidor. Essa funcionalidade foi implementada utilizando criptografia de xor com uma chave fixa.

Usando o Chat

## Comunicação TCP

Toda comunicação via chat é feita através de TCP. Como vários usuários podem estar on-line ao mesmo tempo, é necessária a escrita em várias portas para envio a esses usuários. Então um laço é realizado para que a mesma mensagem seja enviada para todos os usuários da lista de clientes obtidas previamente do servidor.

## Threads

Como podem existir vários clientes conectados, e todos eles podem estar enviado arquivos, é necessário que um cliente esteja apto a receber vários arquivos ao mesmo tempo. Isso é realizado através de threads, ou seja, para cada cliente da lista de cliente que tentar enviar um arquivo, será criada uma thread e, em cada uma delas, o recebimento do arquivo é habilitado.

Transferindo Arquivo

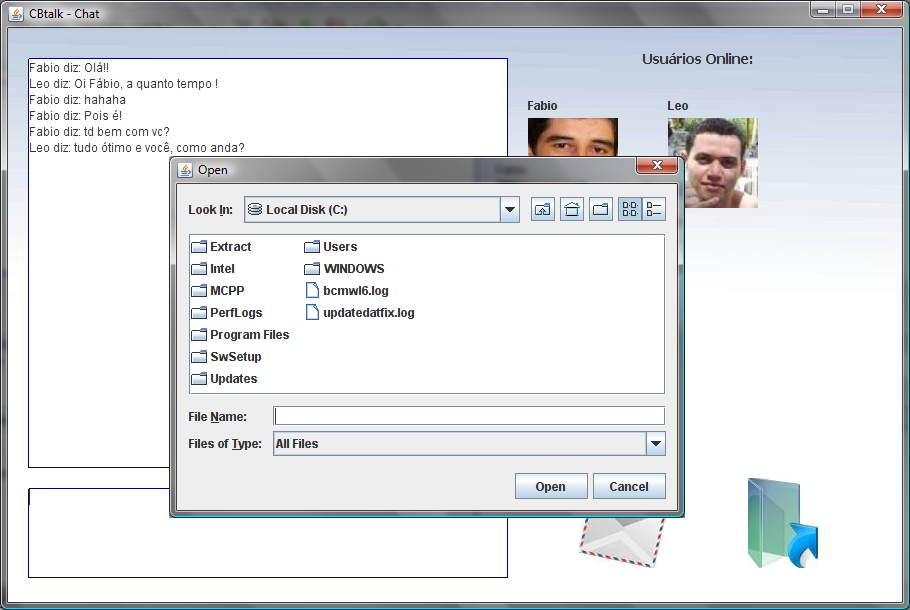


Figura Janela de seleção do arquivo a ser enviado

# Características da Aplicação

O cliente só poderá enviar 1 arquivo por vez, repassando-o para todos os usuários e poderá receber *n* arquivos dos *n* usuários da aplicação. A aplicação possui um buffer de recepção para cada arquivo enviado, possuindo tamanho máximo de 64 Kbytes.

# Estrutura do pacote

O pacote será implementado como segue na figura abaixo. A estrutura do pacote contém os seguintes campos:

* Campo Flag (1 byte): Este campo indica se ainda existem pacotes a serem enviados, se o valor da Flag for 1, significa que existem mais pacotes para serem enviados, caso contrário, não existem mais pacotes para enviar, o pacote que possui Flag 0 é o último pacote do arquivo;
* Campo Tamanho (2 bytes): Este campo representa o tamanho total do pacote;
* Campo numSeq (4 bytes): Este campo contém o número de seqüência do pacote.
* Campo de dados: Este campo possui os dados úteis à aplicação.



Figura Estrutura do Pacote

O cabeçalho terá tamanho fixo de 7 bytes e o tamanho máximo do campo de dados será de 1000 bytes. Logo, o tamanho máximo do pacote será de 1007 bytes.

# Estrutura do ACK

O ACK será enviado via TCP. Portanto, será enviada uma mensagem contendo o número de seqüência do ACK e o tamanho da janela de recepção, utilizada pelo Controle de Fluxo Remetente.

# Lado remetente

Primeiramente, é necessário enviar para todos os usuários on-line o nome do arquivo que será enviado e seu respectivo tamanho via TCP. A partir disso, os usuários on-line aguardam a chegada do arquivo. O primeiro passo para o envio será:

## Segmentação do Arquivo



Figura Ilustração de uma Segmentação

Para envio do arquivo, é necessária a sua divisão em blocos. Isso se deve ao fato do arquivo poder ter um tamanho muito grande e estourar o heap (parte da memória reservada para alocação dinâmica do programa). Depois de pegar apenas um desses blocos, que garantidamente não haverá estouro de heap, essa parcela será ainda quebrada em vários arrays de bytes, e cada um desses arrays será um pacote a ser enviado.

Após esse passo, toda manipulação (verificação e envio) será feita para cada pacote. Para envio de pacotes com confiabilidade, é necessário:

## Buffer de Transmissão



Figura Formato do Buffer de Transmissão

O buffer de saída tem a função de armazenar os pacotes que foram enviados para o destinatário, mas que ainda foram reconhecidos. Dessa forma, quando um envio é realizado, ele fica no buffer até sua confirmação (ACK) chegar. Esse ACK é acumulativo, dessa maneira, esse ACK representa o último pacote recebido com sucesso em ordem esperada. Além disso, o protocolo não descarta pacotes fora de ordem. Eles ficam guardados no buffer até que cheguem outros pacotes com numero de seqüência menor que o dele, e em ordem, possam ser passados para remontagem do arquivo. Esse buffer tem a restrição de ser de 64 kbytes.

# Lado destinatário

Depois de toda parte do remetente, os destinatários irão receber o(s) arquivo(s). Para tal, existe um buffer de chegada para cada arquivo enviado, e esse buffer não pode exceder os 64 kbytes.

## Buffer de Recepção



Figura Buffer de Recepção

O buffer de recepção armazena os pacotes recebidos, e que ainda não foram lidos pela aplicação. Como mostrado acima, tem uma capacidade total de 64 kbytes. À medida que os pacotes são lidos do buffer, novos pacotes podem ser recebidos. Essa leitura é realizada apenas quando o buffer está suficiente cheio a fim de obter um melhor desempenho.

## Remontador do Arquivo



Figura Remontagem de um Arquivo

Para o recebimento do arquivo, primeiramente é necessário transformar um pacote recebido em array de bytes. Depois disso será montado um bloco de arrays de bytes, esse bloco corresponderá a uma parte do arquivo original. Juntando cada um dos blocos em um arquivo final, teremos, no lado destinatário, o arquivo original enviado.

# Confiabilidade

O protocolo de confiabilidade implementado pela aplicação garante a entrega dos dados, sem corrupção e na ordem correta, considerando um canal de comunicação em que existem cenários de atrasos e perdas de pacotes. O protocolo obedece aos conceitos básicos de confiabilidade, como o uso de ACKs acumulativos para confirmar o recebimento de pacotes, o uso de números de seqüências para ordenação dos pacotes, o uso de temporizadores para identificar os possíveis casos de atrasos e perdas e para que seja possível retransmitir os dados que caiam nesses casos.

O protocolo implementado apresenta as seguintes características:

* Envio de ACKs cumulativos: Como já foi explicado anteriormente, o ACK recebido servirá para confirmar o último pacote recebido com sucesso em ordem esperada. Considere que sejam enviados os pacotes com números de seqüência 1,2 e 3 respectivamente. Cada um dos três receberá seus respectivos ACKs de confirmação, pois todos foram recebidos com sucesso e estão na ordem esperada. Caso o ACK do pacote 3 chegue primeiro que os outros ACKs, ele servirá para confirmar todos os pacotes que foram enviados. Considere agora que sejam enviados 6 pacotes com números de seqüência 1,2..,6 respectivamente. Caso ocorra uma perda do pacote 5 e o restante seja recebido com sucesso, o ACK que será enviado será para confirmar o recebimento do pacote 4, pois ele é o último pacote que foi recebido com sucesso e que está em ordem.
* Tratamento dos pacotes desordenados: Os pacotes que chegarem fora de ordem serão armazenados no buffer de recepção, mas não serão enviados ACKs de confirmação destes pacotes. Os ACKs de confirmação deverão ser enviados quando os pacotes na ordem desejada chegarem, fazendo com que os pacotes fora de ordem fiquem na ordem correta. Considere que sejam enviados 6 pacotes com números de seqüência 1,2,..,6, como no exemplo anterior. O pacote 5 se perde e todos os outros chegam com sucesso. Como vimos, o ACK enviado será para o pacote 4. Então, o ACK de confirmação do pacote 6 só será enviado quando o pacote 5 for recebido.
* Envio de ACKs duplicados para sinalizar perdas: Ao identificar a ocorrência de três ACKs com números de seqüência iguais, o remetente irá interpretar a perda do pacote com número de seqüência igual ao dos ACKs duplicados e providenciará a retransmissão desse pacote.
* Retransmissões em casos de perda ou esgotamento do temporizador: Esses dois casos representam os dois tipos de retransmissões, retransmissão comum e a retransmissão rápida. A retransmissão comum se encaixa no caso do esgotamento do temporizador, nesse caso, o remetente irá retransmitir o pacote que iniciou o temporizador. A retransmissão rápida se encaixa no caso da ocorrência de ACKs duplicados, quando isso ocorre o remetente retransmite o pacote com número de seqüência igual ao dos ACKs duplicados.

Abaixo, vamos analisar alguns cenários e as soluções realizadas pelo protocolo implementado. Com isso, mostraremos o seu comportamento diante de cada situação.



Figura Primeira situação: ocorreu perda de pacote e o decorrente esgotamento do temporizador.

O primeiro cenário é apresentado acima na figura 4. Ele descreve os acontecimentos em casos de esgotamento do temporizador. Pode-se resumir o seguinte: Quando ocorre uma perda de um pacote e nenhum ACK foi recebido antes do esgotamento do temporizador, o remetente reconhece que houve perda ou atraso e reenvia o pacote que iniciou o temporizador.



Figura Segunda situação: ocorreu perda de pacote e a decorrente chegada de ACKs duplicados.

O segundo cenário é representado pela figura 5. Ele descreve os acontecimentos decorrentes do recebimento de ACKs duplicados. Pode-se resumir: Quando ocorrer a perda de um pacote e o destinatário continuar a receber os próximos pacotes, ele enviará ACKs duplicados com a confirmação do último pacote recebido na ordem esperada. Ao receber os ACKs duplicados, antes do esgotamento do temporizador, o remetente reconhece a perda do pacote (previsto pelo ACK) e reenvia o pacote perdido. O destinatário ao receber o pacote perdido, enviará um ACK acumulativo para todos os pacotes recebidos em ordem.

Figura Terceira situação: o destinatário recebeu um pacote com número de sequência maior que o esperado.

Abaixo



Figura Máquina de estados finita do Remetente



Figura Máquina de estados finita do destinatário

# Controle de Congestionamento

Com o intuito de não provocar uma sobrecarga na rede pelo envio excessivo de pacotes, foi implementado na aplicação o serviço de Controle de Congestionamento. Ele tem como base uma variável chamada Janela de Congestionamento, esta restringe a taxa à qual um remetente pode enviar tráfego para dentro da rede. Tal restrição limita a quantidade de dados não reconhecidos no remetente e, portanto, limita indiretamente a sua taxa de envio.

O controle de congestionamento é composto por três etapas:

* Partida Lenta – Resulta em uma duplicação da janela do congestionamento a cada RTT.
* Aumento aditivo – Resulta em um aumento da janela de congestionamento de 1 MSS (tamanho máximo do segmento) a cada RTT.
* Diminuição multiplicativa – Resulta em uma diminuição de 50% da janela de congestionamento.

De acordo com a ocorrência de reconhecimentos, perdas de pacote ou esgotamento de temporizador, o controle de congestionamento oscila entre suas três etapas. Dessa forma, a janela de congestionamento está sempre se adequando a capacidade da rede.

O controle de congestionamento monitora os intervalos de envio de cada pacote e de chegada dos seus respectivos ACKs. Esses monitoramentos irão servir para calcular as amostragens dos RTTs. Por conseguinte, estes servirão para calcular os valores Estimados de RTT, que servirão para calcular os valores de DevRTT, e por fim estabelecer os melhores valores de Timeout dos pacotes.

Arquitetura P2P





Diagrama de Classes UML

