Índice

1. Esquema Cliente-Servidor
   1. Cliente
      1. Requisição de conexão
      2. Requisição da lista de clientes
      3. Requisição de desconexão
   2. Servidor
      1. Estabelecendo a conexão
      2. Fornecendo lista de clientes
      3. Estabelecendo desconexão
2. Usando o Chat
   1. Comunicação TCP
   2. Threads
   3. Parte Remetente do Chat
   4. Parte Destinatário do Chat
3. Transferindo Arquivo
   1. Características da Aplicação
   2. Estrutura do pacote
   3. Estrutura do ACK
   4. Parte Remetente
      1. Segmentação do Arquivo
      2. Janela de Remetente
      3. Controle de Fluxo do Remetente
      4. Confiabilidade no Remetente
   5. Parte Destinatário
      1. Remontagem do Arquivo
      2. Janela de Destinatario
      3. Controle de Fluxo do Destinatário
      4. Confiabilidade no Destinatário
   6. Controle de Congestionamento
4. Máquina de Estados
5. Arquitetura P2P
6. Arquitetura Cliente-Servidor
7. Package de Aplicação
8. Package de Confiabilidade
9. Package de Controle
10. Package de Transporte
11. Package de TransporteTCP
12. Package de TransporteUDP

Índice de ilustrações

[Figura 1 Tela inicial do chat 3](#_Toc232794309)

[Figura 2 Tela com Informação dos usuários on-line. 4](#_Toc232794310)

[Figura 3 Tela com informação de desconexão de um usuário. 4](#_Toc232794311)

[Figura 4 Estrutura do Pacote 7](#_Toc232794312)

[Figura 5 Ilustração de uma Segmentação 8](#_Toc232794313)

[Figura 6 Formato do Buffer de Transmissão 9](#_Toc232794314)

[Figura 7 Primeira situação: ocorreu perda de pacote e o decorrente esgotamento do temporizador. 11](#_Toc232794315)

[Figura 8 Segunda situação: ocorreu perda de pacote e a decorrente chegada de ACKs duplicados. 12](#_Toc232794316)

[Figura 9 Terceira situação: o destinatário recebeu um pacote com número de sequência maior que o esperado. 13](#_Toc232794317)

[Figura 10 Máquina de estados finita do Remetente 14](#_Toc232794318)

[Figura 11 Máquina de estados finita do destinatário 14](#_Toc232794319)

[Figura 12 Buffer de Recepção 16](#_Toc232794320)

[Figura 13 Remontagem de um Arquivo 17](#_Toc232794321)

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.

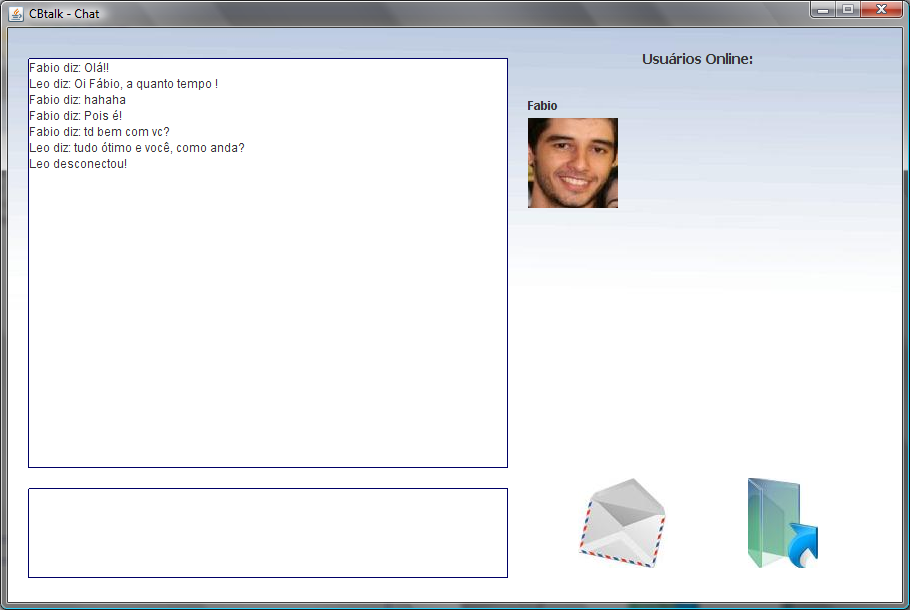


Figura Tela com informação de desconexão de um usuário.

# 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 TRANS ARQUIVO - GUI)

# 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.

# Parte do 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.

## 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.



Figura Máquina de estados finita do Remetente



Figura Máquina de estados finita do destinatário

## Eficiência (Controles) – cong e fluxo

## Janela de Remetente

A Janela de Remetente é implementada a partir de um buffer para o serviço de confiabilidade na transferência do arquivo utilizando protocolo de comunicação UDP. Nela está registrado o status de cada pacote a ser enviado pertencente a essa janela. No caso da Janela de Remetente temos os possíveis status:

1. Pacote já reconhecido;
2. Pacote enviado e ainda não reconhecido;
3. Pacote autorizado e ainda não enviado;
4. Pacote não autorizado;

A confiabilidade da parte do Remetente atualiza a Janela de Remetente sempre que é enviado algum pacote, ou então um ACK de algum dos pacotes enviados é recebido, alterando o status da posição da janela correspondente ao pacote de acordo com cada evento descrito. Para a operação de atualização, são feitas operações para verificar se o pacote pertence à Janela de Remetente e para saber se o ACK recebido é para o pacote que corresponde à posição base da Janela.

## Controle de Fluxo do Remetente(Figura)

Para não provocar um transbordamento no buffer da aplicação do destinatário pelo excessivo envio de pacotes, foi implementado na aplicação o serviço de Controle de Fluxo, nesse caso será explicado à funcionalidade do lado remetente do Controle de Fluxo da aplicação.

Esse lado do Controle registra informações sobre o último byte enviado, o último byte reconhecido (último ACK recebido) e sobre o tamanho da janela de recepção do destinatário, informação enviada pela aplicação destinatária durante a confirmação do recebimento de cada pacote enviado pelo remetente, toda vez que um novo pacote for enviado ou quando um ACK for recebido. Através desses registros será criada uma restrição que, juntamente com a restrição do Controle de Congestionamento, irá manipular a quantidade de pacotes enviados pela aplicação remetente. A restrição do Controle de Fluxo do Remetente é a mesma utilizada pelo Controle de Fluxo do Remetente do protocolo TCP.

# Parte do 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.

## Confiabilidade (Figura)

## 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)



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.

## Janela de Destinatário

A Janela de Destinatário é implementada para o serviço de confiabilidade na transferência do arquivo utilizando protocolo de comunicação UDP e no Controle de Fluxo do Destinatário. Nela está registrado o status de cada pacote a ser recebido pertencente à essa janela. No caso da Janela de Destinatário temos os possíveis status:

1. Pacote fora da ordem já reconhecido;
2. Pacote aguardado ainda não recebido;
3. Pacote aceitável;
4. Pacote não autorizado;

A confiabilidade da parte do Destinatário utiliza a Janela de Destinatário sempre que é recebido algum pacote, nesse caso a janela é atualizada alterando o status da posição da janela correspondente ao pacote recebido, ou então para enviar o ACK pertencente ao pacote com o número de seqüência esperado. Para a operação de atualização, são feitas operações para verificar se o pacote pertence à Janela de Destinatário e para a operação de enviar ACK, será calculado através da Janela de Destinatário, o número de seqüência esperado pela aplicação do próximo pacote.

O Controle de Fluxo do Destinatário utiliza a Janela de Destinatário para saber quais pacotes ainda não foram recebidos, bastando apenas consultar os status das posições correspondentes aos pacotes aguardados e ainda não recebidos. Esse interesse pela Janela de Destinatário será explicado no tópico seguinte.

## Eficiência (Controles)

## Controle de Fluxo do Destinatário

Agora será explicado a funcionalidade do lado destinatário do Controle de Fluxo da aplicação, para concluir a explicação da implementação do serviço de Controle de Fluxo.

Esse lado do Controle irá registrar informações sobre o último byte recebido pelo buffer (podendo pertencer a um pacote fora de ordem), o último byte lido do buffer e sobre o tamanho da janela de recepção, informação calculada e enviada para a aplicação remetente durante o envio dos ACKs, toda vez que um novo pacote for adicionado no buffer ou quando a aplicação fizer uma requisição de leitura do buffer, o qual removerá os dados logo após serem lidos. Através desses registros e da Janela de Destinatário será calculado o tamanho da janela de recepção. Considerando que os pacotes chegam fora de ordem , não podemos contar apenas com o último byte recebido, pois caso ele pertença a um pacote fora de ordem, iremos calcular a janela considerando que os pacotes antes dele já foram recebidos, o que não necessariamente é verdade. Para resolver este impasse, consulta-se a Janela de Destinatário, com ela saberemos quantos pacotes ainda não chegaram desde que o último byte recebido fora de ordem foi adicionado no buffer. Depois disso, é calculado o tamanho desses pacotes e com esse tamanho, juntamente com o valor do último byte recebido e o último byte lido, iremos efetivamente calcular o tamanho da janela de recepção.

# 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, que 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.

Ao serem identificadas perdas de pacotes (caso onde são recebidos três ACKs repetidos), a restrição sobre a quantidade de pacotes a serem enviados pela aplicação (janela de envio) é automaticamente mudada pelo serviço de Controle de Congestionamento, utilizando o algoritmo de controle de congestionamento TCP Reno.

Arquitetura P2P





Diagrama de Classes UML (fazer também em camadas!)

Package de Aplicação



Package de Confiabilidade



Package de Controle



Package de Transporte



Package de TransporteTCP



Package de TransporteUDP

