Redes de Computadores

Licenciatura em Engenharia Informática

Trabalho Final

Sistema Comunicação em Tempo-Real

UNIVERSIDADE DE ÉVORA

Grupo: E18

André Gonçalves, 58392 | André Zhan, 58762

Miguel Pombeiro, 57829 | Miguel Rocha, 58501

Departamento de Informática
Universidade de Évora
Junho 2025

Índice

1 Introdução **2**

2 Protocolo **3**
2.1 Pacotes . . . 3
2.1.1 Autenticação e Registo . . . 3
2.1.2 Envio de mensagens . . . 4
2.1.3 Gestão de Grupos . . . 7
2.1.4 Transferência de Ficheiros . . . 8
2.1.5 Outros . . . 9
2.2 Fluxo de Comunicação . . . 9
2.2.1 Autenticação . . . 10
2.2.2 Envio de Mensagens . . . 10
2.2.3 Gestão de Grupos . . . 11
2.2.4 Transferência de Ficheiros . . . 12
2.2.5 Saída/Perda de Conexão . . . 12
2.3 Tratamento de Erros . . . 13
2.3.1 Envio e Recepção de Mensagens . . . 13
2.3.2 NACKs Mensagens . . . 13
2.3.3 Erros do Utilizador e Clientes Mal Implementados . . . 14

3 Escolhas de Implementação **15**

4 Interface de Cliente **16**
4.1 Menu de autenticação . . . 16
4.2 Menu principal . . . 16
4.3 Menu operação dos grupos . . . 17

5 Avaliação crítica e objetiva da aplicação **17**
5.1 BACK_END . . . 17
5.2 FRONT_END . . . 18

1 Introdução

Este documento descreve a implementação e especificação de um sistema de comunicação letra-a-letra em tempo-real desenvolvido na linguagem de programação C. O objetivo principal deste sistema é permitir a funcionários de empresas trocar mensagens curtas de forma imediata, transmitindo cada tecla à medida que é pressionada - sem a necessidade de um botão "enviar".

Uma vez que o sistema tem de funcionar em tempo-real – com uma latência muito baixa – o sistema foi implementado com recurso ao protocolo UDP, que proporciona uma comunicação rápida. Assim, foi desenvolvido um protocolo de aplicação que permite estruturar a comunicação entre os clientes e o servidor de forma a garantir um bom funcionamento do sistema.
A aplicação consiste, então, em dois componentes principais:

- Um servidor que é responsável por receber, processar e distribuir mensagens entre os clientes conectados e pela manutenção do sistema de grupos.
- Um cliente, que serve de interface para os utilizadores permitindo, entre outros, fazer a captura de *input*, o envio de letras para o servidor (e por consequência outros clientes), bem como a exibição da mensagem recebidas.

Este documento é dividido em duas partes principais:

1. Protocolo (Seção 2), que contém a especificação do procedimento de comunicação implementado. Inclui formatos de pacotes, tipos de mensagens, tratamento de erros, entre outros.
2. Relatório, onde são descritas decisões acerca da implementação do sistema, e uma avaliação critica do projeto.

2 Protocolo

2.1 Pacotes

De forma a facilitar a comunicação entre o servidor e os clientes foram utilizados pacotes de vários tipos, cada um com uma funcionalidade diferente. Os pacotes foram desenhados e implementados como *strings*, sendo que as várias informações enviadas no *payload* foram separadas por *pipes*, |.

Os pacotes que foram utilizados são descritos nesta secção, e estão na forma:

`<Packet_Type>|<payload 1>|<payload 2>|...`

em que

- `<Packet_Type>`, representa o tipo de pacote e é sempre a primeira informação contida no pacote. Na implementação em C foram representados como inteiros entre 1 e 34.
- `<payload ...>`, as restantes informações que são passadas no pacote, e que são necessárias para o bom funcionamento.

2.1.1 Autenticação e Registo

Estes são os pacotes utilizados no processo de registo e autenticação dos clientes no servidor.

Login e registo de um utilizador

`LOGIN|<username>|<password>`

Este é o pacote que os clientes mandam ao servidor quando pedem para iniciar sessão. É composto por dois *payloads*, sendo estes o nome de utilizador e a palavra-passe escolhidos.

`REGISTER|<username>|<password>`

Este é o pacote que os clientes mandam ao servidor ao registar um novo utilizador. É composto por dois *payloads*, sendo estes o nome de utilizador e a palavra-passe escolhidos.

Confirmações

`LOGIN_OK`

Pacote de resposta do servidor quando recebe um pedido de *login* ou registo. Este pacote é enviado de volta ao cliente quando esse pedido foi bem sucedido, ou seja, quando o servidor conseguiu autenticar o utilizador.

`LOGIN_NOK`

Pacote de resposta do servidor quando recebe um pedido de *login* ou registo. Este pacote é enviado de volta ao cliente quando o servidor não conseguiu autenticar o utilizador, como por exemplo, um pedido com as credenciais inválidas.

`LOGOUT_OK`

Pacote de confirmação que o *logout* foi bem-sucedido. Este pacote não foi implementado no lado do cliente porque o servidor já consegue detetar clientes que abandonaram sem dizer a palavra de despedida. Por este motivo, foi apenas implementado no lado do servidor, para futuras melhorias ao trabalho.

`LOGOUT_NOK`

Este é o pacote de confirmação que o *logout* não foi bem sucedido. Pelas mesmos motivos do pacote `LOGOUT_OK`, este não foi implementado do lado do cliente.

Exit

`EXIT`

Este é o pacote enviado ao servidor quando o utilizador deseja sair por conta própria, ao introduzir no terminal, a palavra de despedida, descrita na Seção 4.

2.1.2 Envio de mensagens

Estes são os pacotes utilizados no envio de mensagens entre os clientes. O envio de mensagens pode ser feito de três formas diferentes: mensagens privadas (`MSG_ONE`), mensagens para grupos (`MSG_GROUP`) e mensagens de *broadcast* (`MSG_BROADCAST`).

O processo de envio de mensagens é feito em três passos diferentes:

1. O remetente pede autorização ao servidor para enviar mensagem - Pacotes *Request*.
2. Caso seja possível enviar mensagem, o servidor avisa o(s) destinatário(s) - Pacotes *Warning* - e envia uma confirmação ao remetente. Caso não seja possível, essa informação também é enviada ao remetente.
3. As letras são enviadas uma a uma e quando a mensagem termina, o remetente envia essa informação ao servidor que a reencaminha ao(s) destinatário(s).

De salientar, ainda, que os pacotes de mensagem que contêm letras utilizam um sistema de controlo de erros por sequência, com recurso a NACKs.

Pedidos de Envio de Mensagens

Estes são os pacotes responsáveis por iniciar uma conversa com outros clientes. Como foram implementados três tipos de conversas distintas, cada uma destas tem o seu pedido de iniciação de conversa no lado do servidor.

`MSG_BROADCAST_REQUEST`

Pacote que o cliente envia para o servidor quando pretende iniciar uma conversa para todos os utilizadores disponíveis na altura do pedido.

`MSG_ONE_REQUEST|<destinatário>`

Pacote que o cliente envia para o servidor quando pretende iniciar uma conversa com outro utilizador (destinatário).

`MSG_GROUP_REQUEST|<grupo>`

Pacote que o cliente envia para o servidor quando pretende iniciar uma conversa para todos os utilizadores de um determinado grupo.

Avisos de Mensagens

Estes pacotes são enviados do servidor para todos os clientes destinatários da mensagem para os informar de que vão começar a receber pacotes de letras com uma mensagem de outro cliente. São estes pacotes que vão avisar os clientes dos destinatários que devem bloquear o utilizador de modo a que a mensagem que estão a receber possa ser escrita em tempo-real.

`MSG_BROADCAST_WARNING|<remetente>`

Pacote que o servidor envia aos clientes destinatários a avisar que vão receber uma mensagem de *broadcast* de um cliente, <remetente>.

`MSG_ONE_WARNING|<remetente>`

Pacote que o servidor envia ao cliente destinatário a avisar que vai receber uma mensagem de um cliente, <remetente>.

`MSG_GROUP_WARNING|<grupo>|<remetente>`

Pacote que o servidor envia aos clientes destinatários pertencentes a um grupo, a avisar que vão receber uma mensagem de um cliente, <remetente>.

Envio de Letras

Estes pacotes são enviados do remetente para o servidor e do servidor para o(s) destinatário(s). Estes pacotes contêm uma das letras da mensagem que está a ser escrita pelo remetente, de forma a simular a escrita em tempo-real no(s) cliente(s) destinatário(s). Os pacotes contém, ainda, um número sequencial que indica a posição da letra na mensagem. Este número permite acautelar erros na transmissão de pacotes, tais como a perda de pacotes e a receção fora-de-ordem.

`MSG_BROADCAST|<nº sequência>|<letra>`

Pacote de letra enviado no caso de uma mensagem em *broadcast*.

`MSG_ONE|<nº sequência>|<letra>`

Pacote de letra enviado no caso de uma mensagem entre dois clientes.

`MSG_GROUP|<nº sequência>|<letra>`

Pacote de letra enviado no caso de uma mensagem num grupo.

Controlo de Envio de Mensagens

Estes são os tipos de pacotes responsáveis pelas confirmações de envios de mensagens, de modo a garantir o bom funcionamento dos mesmos.

`MSG_END`

Este pacote é enviado pelo cliente, que acabou de escrever a mensagem, para o servidor, para indicar que a mensagem terminou.

`MSG_OK`

Este pacote é enviado pelo servidor para informar que o pedido de iniciação de mensagem ao(s) outro(s) cliente(s) foi bem sucedido.

`MSG_NOK|<mensagem de aviso>`

Este pacote é enviado pelo servidor para informar que o pedido de iniciação de mensagem ao(s) outro(s) cliente(s) não foi bem sucedido. Para tal, o pacote também contém uma mensagem de aviso, que será útil para o cliente que fez o pedido saber o motivo de não ter sido bem sucedido.

`MSG_NACK|<nº sequência>`

Este pacote pode ser enviado do destinatário de uma mensagem, para o servidor para informar que não recebeu a letra com número de sequência certo, ou seja, recebeu ou um pacote duplicado ou fora de ordem. Também pode ser enviado do servidor para o remetente da mensagem, para informar que não recebeu o pacote de número de sequência certo.

2.1.3 Gestão de Grupos

Estes são os pacotes utilizados para o cliente enviar informação sobre gestão de grupos ao servidor. Cada uma das operações que é possível realizar ao nível de grupos, recorre a um dos seguintes pacotes.

Operações sobre Grupos

`GROUP_CREATE|<grupo>`

Pacote enviado pelo cliente para o servidor, com o objetivo de criar um grupo com o nome que está no *payload* do pacote, <grupo>.

`GROUP_DELETE|<grupo>`

Pacote enviado pelo cliente para o servidor, com o objetivo de apagar um grupo com o nome que está no *payload* do pacote, <grupo>. É de notar que, apenas o criador do grupo o pode apagar.

`GROUP_INVITE|<grupo>|<username>`

Pacote enviado pelo cliente para o servidor, com o objetivo de convidar um utilizador, <username> para o grupo, <grupo>. Qualquer membro do grupo pode convidar membros, e estes não precisam de aceitar o convite.

`GROUP_KICK|<grupo>|<username>`

Pacote enviado pelo cliente para o servidor, com o objetivo de expulsar um utilizador, <username> do grupo, <grupo>. Qualquer membro do grupo pode expulsar membros.

`GROUP_LEAVE|<grupo`

Pacote enviado pelo cliente para o servidor, com o objetivo de sair de um grupo, <grupo>. O dono do grupo não consegue sair do grupo.

Controlo de Operações sobre Grupos

Foram implementados dois pacotes relacionados ao sucesso de qualquer operação sobre grupos mencionada acima.

`GROUP_OK`

Este pacote é enviado do servidor para o cliente com objetivo de informar o utilizador que a operação sobre grupos escolhida, foi realizada com sucesso.

`GROUP_NOK|<mensagem de erro>`

Este pacote é enviado do servidor para o cliente com objetivo de informar o utilizador que a operação sobre grupos escolhida, não foi realizada com sucesso. Este pacote também guarda uma mensagem de erro, relativamente ao não sucesso da operação.

`GROUP_INFO|<mensagem>`

Este é um pacote de informação que é enviado do servidor para um determinado cliente. A mensagem do *payload* deste pacote é relativamente a atualizações que aconteceram ao nível dos grupos, como a informação de expulsão de um utilizador de determinado grupo ou do convite para um grupo.

2.1.4 Transferência de Ficheiros

A transferência de ficheiros entre clientes é feita com base no sistema *peer-to-peer*, sem que o ficheiro passe pelo servidor. Deste modo, a comunicação com o servidor é apenas feita para que o remetente do ficheiro possa informar os destinatários da sua intenção de partilhar um ficheiro. De salientar que a transferência de ficheiros é feita apenas em modo *broadcast*, sendo que todos os clientes conectados ao servidor vão ser informados quando um cliente inicia a partilha.

Os pacotes utilizados neste processo são descritos nesta secção.

Pedido de Envio de Ficheiro

`FILE_SEND_REQUEST|<filename>|<tamanho>|<ip_remetente>|<porta_tcp>`

Este pacote é enviado do cliente que pretende enviar o ficheiro para o servidor, para o informar que pretende iniciar a partilha de um ficheiro. Este pacote contém informação relativa ao ficheiro e à partilha, que deve ser retransmitida aos outros utilizadores. Assim, está incluído o nome do ficheiro, bem como o seu tamanho e ainda, o endereço IP do remetente e a porta TCP que está disponível para a partilha do ficheiro.

Aviso de Envio de Ficheiro

`FILE_SEND_WARNING |<filename>|<tamanho>|<porta_tcp>|<username>`

Este pacote é enviado do servidor para todos os clientes ativos para informar que um cliente pretende partilhar um ficheiro. É utilizado para informar os clientes que estão bloqueados de receber mensagens até que enviem uma resposta a este pedido. Neste pacote são incluídas informações essenciais ao processo de recepção do ficheiro, como o nome do ficheiro, o seu tamanho e o nome do remetente, bem como o endereço IP do remetente e sua a porta TCP que está disponível para a partilha do ficheiro.

Aceitação/Rejeição de Ficheiro Estes pacotes são enviados dos clientes destinatários para o servidor para indicar que o cliente em questão já tomou a sua decisão relativamente ao ficheiro. Servem para avisar o servidor que o cliente já pode ser desbloqueado e que está disponível para receber mensagens.

`FILE_SEND_ACCEPT |<filename>`

Este pacote é enviado ao servidor para indicar que o cliente aceitou receber o ficheiro.

`FILE_SEND_DECLINE |<filename>`

Este pacote é enviado ao servidor para indicar que o cliente não aceitou receber o ficheiro.

2.1.5 Outros

Os restantes pacotes utilizados ao nível do controlo e partilha de informação por parte do servidor são descritos nesta secção.

Heartbeat - Controlo de Clientes Ativos

`HEARTBEAT`

Este pacote é utilizado para que o servidor consiga controlar quais os clientes que ainda estão ativos e à espera de receber mensagens. O servidor espera que todos os clientes enviem este pacote em intervalos de tempo regulares. Quando o servidor recebe este pacote, responde ao remetente com um pacote igual de forma a sinalizar que recebeu e que ainda continua ativo.

Anúncios do Servidor

`SERVER_ANNOUNCEMENT |<mensagem>`

Este pacote é utilizado pelo servidor para difundir anúncios importantes para todos os clientes ativos. É, normalmente, utilizado para avisar os utilizadores quando um novo cliente se autentica ou se desconecta no servidor.

2.2 Fluxo de Comunicação

A comunicação entre os clientes e o servidor é feita com recurso aos pacotes já descritos na Subseção 2.1. Nesta secção são descritos os principais processos de comunicação do sistema de comunicação em tempo-real.

2.2.1 Autenticação

A autenticação é o procedimento inicial que o cliente vai fazer para se identificar ao servidor. O servidor apenas vai aceitar pacotes de autenticação de clientes não autenticados. Os restantes tipos de pacotes serão ignorados, uma vez que não irão afetar o estado do servidor.

1. O processo de autenticação é iniciado pelo cliente que, dependendo da ação desejada (*Login* ou Registo), envia o respetivo pacote ao servidor.

    1.1 No caso de *login*, o servidor irá verificar se as credenciais fornecidas correspondem a um utilizador já registado e se são válidas.

    1.2 No caso do registo de um novo cliente, o servidor irá verificar se já existe algum utilizador registado com as mesmas credenciais.

2. Por fim, o servidor vai enviar uma mensagem de resposta ao cliente que o informa se o seu pedido foi ou não bem sucedido. Assim, se foi possível fazer correspondência das credenciais fornecidas ou registar o utilizador, o cliente fica autenticado e pode começar a interagir com o servidor.

2.2.2 Envio de Mensagens

Os três tipos de mensagens que podem ser trocadas entre os clientes seguem um procedimento idêntico.

1. O cliente remetente inicia o processo enviando um pacote ao servidor com um pedido para enviar mensagem. Este pacote especifica o tipo de mensagem que é pretendida.

2. Após receber o pacote de pedido, o servidor vai identificar o remetente e verificar se é possível atendê-lo. Estes pedidos apenas podem ser atendidos se os destinatários estiverem disponíveis para receber mensagens (não estão bloqueados).

    - Caso seja possível atender ao pedido, tanto o remetente como os destinatários, são bloqueados (não podem receber outras mensagens) e são notificados, através de pacotes de aviso, de que o processo de envio de letras vai começar.
    - Caso não seja possível atender o pedido, o remetente é informado e o processo termina.

3. Depois de notificado, o cliente remetente pode começar a enviar pacotes de letras para o servidor que, por sua vez, os reencaminha para o(s) destinatário(s).

4. Quando a mensagem termina, o remetente notifica o servidor. Este, por sua vez, vai proceder ao desbloqueio e notificação de todos os clientes envolvidos.

5. Quando recebem esta notificação, os clientes destinatários ficam disponíveis para receber novas mensagens.

Durante o processo de envio e recepção de letras, é utilizado um processo de controlo de erros com base em NACKs. Este processo utiliza os números sequenciais incluídos nos pacotes de mensagem para detetar se os pacotes chegam ao destino por ordem.

Quando um pacote de letra é recebido tanto pelo cliente como pelo servidor, é verificado se o número recebido corresponde ao esperado:

- No caso do servidor, se o pacote tiver um número superior ao esperado, é enviado um NACK ao remetente, identificando o número não recebido. O remetente deverá voltar a enviar o pacote correspondente.
- No caso do cliente destinatário, se o pacote tiver um número superior ao esperado, é enviado um NACK ao servidor. O servidor irá voltar a enviar o pacote correspondente.

Desta forma, é garantido que as letras são sempre exibidas ao utilizador pela ordem correta, garantindo o bom funcionamento da aplicação.

2.2.3 Gestão de Grupos

A gestão dos grupos é composta por várias operações, como a criação, convite, expulsão, abandono ou eliminação de grupos. O processo de comunicação deste é semelhante.

1. O cliente envia ao servidor um dos pacotes de operações sobre grupos, conforme a ação desejada.
2. O servidor recebe o pacote da respetiva ação, e confirma se é possível realizar tal ação.
    - Caso seja possível realizar a ação, o servidor inicia o processo de tratar do pedido no seu lado e envia um pacote de volta para informar que o pedido foi processado com sucesso.
    - Caso não seja possível realizar a ação, o servidor não realiza tal pedido e envia um pacote de volta para informar que ação foi sem sucesso.
3. O servidor pode ainda, em casos como convites ou expulsões, enviar um pacote ao utilizador afetado (caso esteja online), notificando-o sobre a sua alteração no determinado grupo.

2.2.4 Transferência de Ficheiros

A transferência de ficheiros entre clientes recorre a uma combinação de comunicação UDP e a própria transferência de ficheiros por TCP. Isto deve-se ao protocolo TCP garantir a ordem dos pacotes e não ter perda dos mesmos, sendo confiável para este propósito.

1. O cliente que pretende enviar um ficheiro abre um *socket* TCP, que ficará à escuta de ligações. Após a criação deste *socket*, é enviado um pacote ao servidor com as informações do ficheiro que os clientes necessitam. O servidor TCP é depois encerrado após o tempo limite de 60 segundos.

2. O servidor manda um pacote de aviso a todos os clientes disponíveis, informando que alguém está a enviar um ficheiro.

3. Cada cliente notificado recebe assim uma mensagem solicitando uma resposta, de aceitação ou negação do ficheiro, com um tempo limite de 60 segundos.

    - Caso o cliente aceite, é estabelecida uma ligação TCP entre o destinatário e o remetente. Depois, o ficheiro necessita de ser lido e enviado *byte* a *byte* até conseguir ler o conteúdo todo do mesmo. Após isto, os dados são gravados num ficheiro com o mesmo nome.
    - Caso o cliente rejeite, não se conecta à porta TCP de quem está a enviar o ficheiro.

4. Após o destinatário receber o ficheiro, é encerrada a sua ligação TCP.

2.2.5 Saída/Perda de Conexão

A desconexão do cliente pode ocorrer por dois motivos: o utilizador escreve a palavra de despedida ou, o servidor deteta inatividade e desconecta o cliente.

No caso do utilizador pretender sair do *chat*, ele escreve a palavra de despedida e será enviado um pacote `EXIT` para o servidor. Já para tratar da inatividade por parte do cliente, é implementado um sistema por *heartbeats*, em que o servidor verifica quanto tempo passou desde o último *heartbeat* enviado pelo cliente. Se for excedido um período máximo de 120 segundos, o mesmo será desconectado pelo servidor. Acrescentando, do lado do cliente deve ser enviado um pacote de *heartbeat* a cada 60 segundos.

Para desconectar o cliente, o servidor adota o seguinte procedimento:

1. O servidor remove o cliente do *chat*.

2. Logo após, o servidor verifica ainda se o cliente estava a enviar mensagens - caso estivesse, o estado dos clientes, que estavam a receber as mensagens é restaurado, evitando que o mesmo fique bloqueado no estado anterior.

3. Por fim, o servidor envia uma mensagem a confirmar a saída do cliente e notifica todos os clientes ativos acerca da mesma.

2.3 Tratamento de Erros

Este sistema de comunicação em tempo-real assenta no protocolo de transporte UDP e, como tal, está bastante suscetível a falhas na transmissão de pacotes, que tiveram de ser acauteladas no desenvolvimento do protocolo de aplicação. Assim sendo, foram adotadas várias estratégias para garantir que as mensagens importantes sejam entregues ao destinatários sem problema. Estas estratégias assentam em recuperação de erros na transmissão de pacotes, bem como em mecanismos de *timeout* para garantir que os utilizadores não sejam afetados permanentemente quando ocorrem problemas.

2.3.1 Envio e Recepção de Mensagens

Este sistema de comunicação letra-a-letra funciona com base num terminal de computador, que os utilizadores usam para interagir com o cliente. Assim, há que tomar cuidados especiais de forma a otimizar a experiência para os utilizadores e garantir que não há várias mensagens a ser mostradas ao mesmo tempo. Para que tal aconteça cada cliente apenas pode receber ou enviar uma única mensagem de cada vez. Este processo de bloqueio dos clientes é feito ao nível do servidor que apenas permite que mensagens sejam enviadas para clientes que não estejam já em processo de troca de mensagem. Ao nível dos clientes, é também garantido que enquanto uma mensagem está a ser recebida o cliente tem o seu terminal bloqueado. No final, quando a troca de mensagens é terminada pelo remetente, todos os clientes envolvidos são desbloqueados, voltando ao estado normal.

Uma vez que a interface do cliente destinatário também é bloqueada quando recebe mensagens de letra, é também garantido que este não fica bloqueado permanentemente quando o cliente remetente se desconecta de forma inesperada, sem terminar a mensagem. Este processo de desbloqueio recorre ao método de deteção por *heartbeat*.

2.3.2 NACKs Mensagens

Para tratar a perda de pacotes durante o envio de mensagens, foi implementado um controlo de erros com base em NACKs. Cada pacote de letra enviado é acompanhado de um número sequencial de forma a ser possível fazer o acompanhamento dos pacotes enviados e recebidos tanto para o servidor como para o cliente. As estratégias implementadas no servidor e no cliente foram muito semelhantes. Para o servidor, caso seja detetado um número superior ao esperado, é enviado um NACK ao remetente, identificando o número do pacote não recebido, tendo o remetente que reenviar o pacote correspondente. Para o Cliente, caso o pacote não tenha a numeração correta, é enviado um NACK ao servidor, que terá que reenviar o pacote correspondente.

2.3.3 Erros do Utilizador e Clientes Mal Implementados

Para tratar clientes mal implementados o servidor segue as seguintes instruções, de acordo com a situação:

- Em situações de autenticação, quando o *username* ou *password* são inválidos, assim como a eventualidade do cliente já estar autenticado e tentar autenticar-se na mesma, serão tratadas com o envio de um aviso (`LOGIN_NOK`) por parte do servidor.
- Quando o cliente envia um pacote com parâmetros inválidos, que requer, por exemplo, o *username* do destinatário ou o nome do grupo, e estes não passam na verificação de validade dos mesmos por parte do servidor, este envia uma mensagem de aviso: `MSG_NOK` e `GROUP_NOK` respetivamente.
- Pacotes não reconhecidos pelo servidor (e.g. pacotes mal-formatados), são ignorados.

3 Escolhas de Implementação

Ao longo da realização do trabalho, foi necessário fazer escolhas de alguns pontos não mencionados no enunciado, e de outras situações mais específicas para a esta implementação. Uma vez que os utilizadores registados e os grupos necessitam de persistir após o servidor reiniciar, foi decidido implementar as bases de dados a partir de ficheiros ".csv", isto é, ficheiros *comma-separated values*. Neste tipo de ficheiros, os dados estão separados por vírgulas, facilitando assim a leitura e processamento dos mesmos.

Para a implementação dos grupos, e devido à omissão desta informação no enunciado, foi decidido que a expulsão de um determinado utilizador dentro de um grupo pode ser realizada por qualquer membro desse mesmo grupo. Também relativamente aos grupos, o dono de um grupo não consegue sair desse tal grupo, ou seja, se ele quiser sair, precisa de apagar o grupo.

Relativamente ao tratamento de mensagens simultâneas de vários utilizadores, o grupo optou por bloquear os clientes que estiverem a receber mensagens (caracteres), de forma a evitar "letras" misturadas no *output*, e facilitar o uso da aplicação por parte dos utilizadores. Isto deve-se ao facto da interface do utilizador escolhida ser baseada no emulador do terminal, não tendo outra opção. Sendo assim, esta implementação precisou de usar semáforos para bloquear os utilizadores que estão a conversar, bem como para não interromper os clientes que estão no meio de uma conversa, com avisos do servidor. Foi também necessário o uso de *threads* no cliente para a execução simultânea de várias tarefas em paralelo.

Como o serviço é baseado no protocolo de transporte UDP, podem existir perdas de pacotes e pacotes fora de ordem. Para tal, foi implementado um sistema por *timeouts* que, para pacotes que necessitam de confirmação, espera pela resposta da confirmação do servidor durante um intervalo de tempo limitado. Desta forma, quando um pedido do cliente não tem um resposta atempada do servidor, este acaba por desbloquear, informando o utilizador da situação.

Do mesmo modo, foi implementada uma estratégia de controlo de erros para garantir que as mensagens chegam na ordem correta, baseado em NACKs, já descrito no protocolo na Seção 2.

Quanto à transferência de ficheiros, e devido à omissão no enunciado, foi decidido que o aviso de envio dum ficheiro, é direcionado para todos os clientes disponíveis e não para um determinado cliente.

Por fim, como foi referido no protocolo, Seção 2, foi implementado um sistema por *heartbeats*, de modo a garantir a deteção de clientes que abandonaram sem enviar a mensagem de despedida, ou seja, clientes que não avisaram o servidor que saíram.

4 Interface de Cliente

Como este serviço foi implementado com a linguagem C, optou-se por não usar qualquer tipo de interface externa ao programa, sendo esta o próprio emulador do terminal. A interface do cliente é composta por três menus, sendo estes, o menu de autenticação, o menu principal de operações, e o menu de seleção de operações de grupos.

4.1 Menu de autenticação

Este menu é composto por três escolhas principais: login(1), registo(2), e sair(3). Cada escolha destas está representada por um inteiro, que o utilizador precisa de escrever no terminal. Caso a escolha seja *login* ou registo, serão adicionados dois campos extra, para o *username* e a palavra-passe dos mesmos. Cada campo destes tem o limite de 31 caracteres, ou seja, não há utilizadores com nomes superiores. Caso a escolha seja sair, o programa terminará sem o utilizador se autenticar. Depois de cada escolha, este menu deixa de existir, e o utilizador passa para o menu principal.

4.2 Menu principal

Este é o menu onde todas as operações principais serão realizadas. Estas operações foram divididas em comandos, que o utilizador precisa de digitar para poder realizar o pedido desejado.
Estes comandos são:

- /help - Este comando mostra todos os comandos disponíveis deste menu.
- /exit - Este comando serve como a palavra de despedida do utilizador, permite ao utilizador sair do programa.
- /broadcast - Este comando serve para mandar uma mensagem para todos os utilizadores disponíveis, ou seja, aqueles que não estão bloqueados a receber ou a mandar mensagens. Após isso, o processo de enviar mensagens globais irá começar, e devido ao *chat* ser em tempo real, para acabar a mensagem é necessário teclar "ENTER".
- /pm - Este comando serve para enviar uma mensagem privada para um determinado utilizador que esteja disponível para receber mensagens. Depois de selecionar este comando, será pedido ao utilizador para indicar o nome do cliente ao qual pretende mandar mensagem. Após isso, o processo de enviar mensagens privadas irá começar, e devido ao *chat* ser em tempo real, para acabar a mensagem é necessário teclar "ENTER".

- /group - Este é o comando intermediário da seleção do menu dos grupos. Após digitar este comando, o utilizador sairá deste menu, e irá para o menu de operações de grupos, descrito abaixo.
- /sendfile - É o comando para iniciar o envio de um ficheiro para todos os utilizadores disponíveis. Depois de selecionar esta opção será pedido o nome do ficheiro. Caso o ficheiro se encontre noutra diretoria, é necessário escrever o caminho completo.

4.3 Menu operação dos grupos

Este é o menu que contém todas as operações relacionadas a grupos. Cada operação de grupos está associada a um número inteiro, que o utilizador precisa de digitar para escolher a opção desejada. As operações possíveis são:

- 1. mensagem para grupos
- 2. convidar para grupo
- 3. sair de um grupo
- 4. expulsar alguém de um grupo
- 5. criar um grupo
- 6. apagar um grupo
- 0. voltar para o menu principal

Dependendo da operação escolhida, será depois, perguntado por um nome de grupo, e/ou por *username* de alguém (caso a operação o permita).

5 Avaliação crítica e objetiva da aplicação

Para concretizar uma avaliação crítica e objetiva da aplicação justa e coerente, foram expostos os pontos fortes da mesma, assim como os pontos fracos, através de uma análise persistente do `BACK_END` e `FRONT_END` da implementação.

5.1 BACK_END

Neste trabalho, o foco do grupo esteve direcionado na implementação bem concebida das várias funcionalidades sugeridas no enunciado, além de procurar solucionar da melhor forma os vários dilemas encontrados no decorrer da realização do mesmo. Tendo em conta vários testes realizados à aplicação das diferentes funcionalidades, para uma avaliação crítica consistente, o grupo observou que as funcionalidades cruciais ao funcionamento

do *chat* - como o envio de mensagens entre clientes (letra-a-letra), a notificação de entrada/saída de clientes por parte do servidor e a deteção por parte do servidor, que um cliente "abandonou" o serviço sem enviar a mensagem de despedida - estavam completamente funcionais. Além disso, o grupo optou também por implementar as funcionalidades extras apresentadas no enunciado.

A autenticação foi implementada com o uso de um *username* e *password* únicos por utilizador, que o mesmo usa para autenticar no *chat*, identificadores estes que são concebidos no ato de registo do mesmo. Além disso, estes atributos são guardados num ficheiro, para que não sejam perdidos quando o servidor for reiniciado. Para solucionar a possível perda de pacotes, devido ao uso de UDP, foi implementado um sistema por *timeouts*, e no caso das mensagens, um sistema por NACKs, também já descrito (Seção 2). Também a transferência de ficheiros através de um servidor TCP, para evitar perda de pacotes, foi implementada, estando completamente funcional. Acrescentando, a possibilidade da criação de grupos e a comunicação entre os membros dos mesmos foi bem concebida, possibilitando a criação e remoção do grupo apenas pelo criador, assim como o convite e remoção de utilizadores para o mesmo.

Após a revisão e testes às funcionalidades anteriormente mencionadas, o grupo deliberou que estas estavam muito bem concebidas, e considerou o `BACK_END` como o ponto forte do trabalho. No entanto, a solução encontrada para tratar mensagens simultâneas de vários utilizadores, que recorre ao bloqueamento dos clientes envolvidos, não foi a melhor solução possível para o problema.

5.2 FRONT_END

Ao nível da interface e apresentação da aplicação - que o grupo reconhece ser fundamental para questões de intuitividade e uso devido do *chat* por parte dos utilizadores - não atenderam às potenciais expectativas estabelecidas. Como mencionado anteriormente, durante a realização deste trabalho, o foco esteve direcionado para a implementação das diversas funcionalidades referidas e, consequentemente, o grupo teve em consideração outras prioridades em detrimento do desenvolvimento visual. Este facto constitui, portanto, um ponto negativo na avaliação crítica do projeto, dado que compromete, de certa forma, a experiência global do utilizador.

Concluindo, a avaliação crítica do projeto permitiu identificar os principais méritos e limitações da implementação realizada. Destaca-se, como ponto forte, a robustez e completude do `BACK_END`, com a correta implementação das funcionalidades essenciais e adicionais propostas. Por outro lado, a menor atenção dedicada ao `FRONT_END` traduziu-se numa interface menos cuidada, o que prejudica parcialmente a usabilidade da aplicação. Assim, o grupo concluiu que a análise efetuada revela um trabalho globalmente sólido, ainda que com margem de melhoria na vertente visual e de interação com o utilizador.