Projeto da Disciplina Infraestrutura de Comunicações - 2024.2

O projeto será dividido em três partes, a primeira entrega será dada por uma contenção de paradigma Cliente - Servidor, com envio e recebimento de arquivos, pelo menos de 2 tipos diferentes (txt, imagem, mp4, mp3, pdf, etc), utilizando comunicação UDP. Para a segunda entrega, complementando o código da entrega 1, o grupo deve ser otimizado, de modo a implementar um protocolo básico de transferência confiável utilizando UDP e o método RDT 3.0 apresentado em sala de aula. Por fim, na terceira e última etapa as equipes devem entregar um sistema de chat em grupo “ChatCin” com as especificações descritas neste documento. **Fiquem atentos aos prazos**.

A seguir, serão descritas as especificações e requisitos de cada uma das etapas:

1. Primeira Etapa: Transmissão de arquivos com UDP

- **(2,0 pontos)** Implementação de comunicação **UDP** utilizando a biblioteca **Socket** na linguagem **Python**, com envio e devolução de arquivo (o arquivo deve ser enviado pelo cliente, armazenado no servidor e devolvido ao cliente) em pacotes de até 1024 bytes (buffer_size). Não é necessária a implementação de transferência confiável nessa etapa, somente na etapa 2. Prazo máximo de entrega: 10/02

- **A implementação deverá ser realizada conforme os requisitos a seguir:**

    - Nessa entrega se deve implementar o envio e devolução de arquivos, reforçando que o envio de strings não é suficiente. Sugerimos que testem o programa para ao menos dois tipos de arquivos, como por exemplo um .txt e uma imagem.
    - É necessário a alteração do nome do arquivo antes da devolução ao cliente para demonstrar o funcionamento correto do código.

    - É recomendável a divisão do cliente e servidor em arquivos (.py) diferentes, para a execução de funcionalidades específicas. Lembrar que essa modularização é importante dado a escalabilidade das próximas entregas.

    - **Dica**: É uma dúvida comum se mensagem é diferente de arquivo, ou se ambos podem ser enviados em um só pacote com buffer_size infinito. Uma mensagem ou um arquivo são ou devem ser considerados a mesma coisa, bits, e devem seguir o mesmo fluxo. O que muda é que arquivos ou mensagens maiores que o buffer_size (lembrando: 1024 bytes) devem ser fragmentados em pacotes e reconstruídos no recebedor, recomendamos utilizar file.read e file.write (Python).

2. Segunda Etapa: Implementando uma transferência confiável com RDT 3.0

- **(3,5 pontos)** Simulação de transferência confiável, segundo o canal de transmissão confiável RDT3.0, apresentado na disciplina e presente no Kurose, utilizando-se do código resultado da etapa anterior (**envio de arquivos de tipos diferentes, entrega e devolução dos mesmos**). A cada passo executado do algoritmo, em tempo de execução, deve ser printado na linha de comando, de modo a se ter compreensão do que está acontecendo. Para o teste do algoritmo, deve ser simulado um gerador de perdas de pacotes aleatórios, ocasionando timeout no transmissor para tais pacotes, com o intuito de demonstrar a eficiência do RDT3.0 implementado. Prazo máximo de entrega: 15/03

    - **A implementação deverá ser realizada conforme os requisitos a seguir:**

        - **O socket UDP de cada cliente e do servidor deverá contar com transmissão confiável, implementada em camada de aplicação segundo o RDT3.0 que consta no livro "Redes de Computadores e a Internet" do Kurose.**

            - Obs.: O RDT3.0 apresentado pelo Kurose utiliza um checksum. Entretanto, para esse projeto não é necessário a implementação do checksum, pois o UDP já realiza essa função (e antes do UDP também há um checksum na camada de enlace).

3. Terceira Etapa: Sistema de chat em grupo com paradigma cliente-servidor

- **(4,5 pontos)** O Cin criou um sistema de comunicação para facilitar a interação entre os estudantes, professores e colaboradores. O projeto será dado a partir da implementação de um sistema de chat, exibido por linha de comando. Apesar do reaproveitamento das etapas anteriores, o histórico da execução dos algoritmos, como o envio e devolução de arquivos, não deve ser exibido nessa etapa, apenas a aplicação como descrita nesse documento e mantendo o uso do RDT3.0. Prazo máximo de entrega: 01/04

- **A implementação deverá ser realizada conforme os requisitos a seguir:**

  - Na terceira entrega do projeto é necessário que o "ChatCin" funcione para mais de um cliente simultaneamente, ou seja, deverão ser abertos o terminal do servidor e ao menos dois terminais de clientes (com portas diferentes) sem que ocorra interrupção do funcionamento.

  - **O socket UDP de cada cliente e do servidor deverá contar com transmissão confiável, implementada em camada de aplicação segundo o RDT3.0 que consta no livro "Redes de Computadores e a Internet" do Kurose.**

  - Adicionalmente, para a última entrega, a equipe deverá apresentar um vídeo com no máximo 15 minutos de duração. Nele, a equipe irá explicar o código e mostrar o chat funcionando. Todos os integrantes do grupo devem participar.

!!O que está em vermelho não é negociável!!

Requisitos das funcionalidades:

Exemplo de lista de contatos:

Nome	**IP:PORTA**
Felipe Maltez	192.168.100.100:5000
Vitor Azevedo	192.168.100.100:5500

- Obs.: **Considere clientes como processos. Verifique que cada cliente tem portas diferentes. Pode considerar o IP local como padrão para todos os clientes.**
- Obs.**: Não pode existir nomes iguais**

As funcionalidades serão executadas/solicitadas através de linhas de comando pelo cliente e serão interpretadas pelo sistema. A tabela abaixo apresenta as funcionalidades requeridas.

Funcionalidade	**Comando**
Conectar ao sistema	**login <nome_do_usuario>**
Sair do sistema	**logout**

Exibir pessoas conectadas	**list:cinners**
Exibir seus amigos	**list:friends**
Exibir seu grupos	**list:mygroups**
Exibir grupos que faz parte	**list:groups**
Seguir amigo	**follow <nome_do_usuario>**
Deixar seguir amigo	**unfollow <nome_do_usuario>**
Criar grupo	**create_group <nome_do_grupo>**
Excluir grupo	**delete_group <nome_do_grupo>**
Entrar em grupo	**join <nome_do_grupo> <chave_grupo>**
Sair do grupo	**leave <nome_do_grupo>**
Banir amigo de grupo	**ban <nome_do_usuario>**
Conversar no grupo	**chat_group <nome_grupo> <chave_grupo> <mensagem>**
Conversa particular	**chat_friend <nome_friend> <mensagem>**

Descrição das funcionalidades:

login <nome_do_usuario>

Cliente deve estar online para acesso de todas as funcionalidades do sistema

- O cliente irá logar com nome. Dois usuários não podem se conectar ao sistema utilizando o mesmo nome.
- Recomendamos que haja alguma lista com informações automáticas com nomes que os associam com um ip:porta já definidos. De toda forma, cada cliente deve ter uma porta única.
- Ao logar, o servidor deve enviar a seguinte mensagem para o cliente: "*você está online!*"

logout

- Ao usar o comando logout, o cliente deve ser excluído da lista de usuários on-line para o servidor.
- Ao sair, o cliente não pode ter acesso às funcionalidades

create_group <nome_do_grupo>

- O usuário “Organizador / criador do grupo” não pode ter mais de um grupo com o mesmo nome
    - Caso haja uma tentativa de criar uma acomodação já existente, o cliente deve ser alertado pelo servidor com uma mensagem de erro.
- É recomendável a existência de uma **chave id** por acomodação de modo a facilitar a validação da entrada de novos usuários no grupo, assim como a verificação do grupo referenciado pelo nome. Esse id pode ser gerado automaticamente no momento de criação da reserva, ou ser criado diretamente como parâmetro da função **create_group.**
- O cliente que criou a acomodação deve receber do servidor a seguinte mensagem: “**o grupo de nome <nome_da_acomodação> foi criada com sucesso!”**
- Ao criar um novo grupo, este será adicionado à lista de grupo criada, podendo consultar posteriormente.

delete_group <nome_grupo>

- Apenas o administrador do grupo pode excluir o grupo
- Ao excluir o grupo, o administrador do grupo deve receber uma mensagem de confirmação.
- Ao excluir o grupo, os amigos do grupo também devem receber uma mensagem com a seguinte estrutura:
  [<nome_do_administrador>/<IP>:<SOCKET>]<mensagem>.
    - <mensagem>: ‘O grupo {nome_do_grupo} foi deletado pelo administrador’

join <nome_do_grupo> <chave_grupo>

- Usar a chave gerada na criação dos grupos para entrar
- Se o usuário já está no grupo, enviar mensagem “Você já está no grupo”
- Se o usuário não está no grupo, o servidor enviará para todos os usuários já existentes no grupo:
  [<nome_usuário>/<IP>:<SOCKET>]<mensagem>.
    - <mensagem> : “{nome_usuário} acabou de entrar no grupo”

leave <nome_do_grupo>

- Ao sair do grupo, não poderá mais comunicar-se ou acompanhar interações dos outros amigos no grupo.
- Ao sair do grupo, o servidor enviará para todos os usuários existentes no grupo:
  [<nome_usuário>/<IP>:<SOCKET>]<mensagem>.
  - <mensagem> : “{nome_usuário} saiu do grupo”

ban <nome_do_usuario>

- Apenas o administrador do grupo pode banir alguém do grupo
- Todos os integrantes do grupo, menos quem foi banido deve receber a mensagem: “{nome_do_banido} foi banido do grupo”
- O cliente que foi banido deve receber a seguinte mensagem:
  [<nome_do_administrador>/<IP>:<SOCKET>]<mensagem>.
  - <mensagem>: ‘O administrador do grupo {nome_do_grupo} o baniu”

list:cinners

- Lista todos os usuários conectados no servidor do “ChatCin”

list:friends

- Lista de todos os amigos do particulares do usuário

list:groups

- Lista de todos os grupos que o usuário faz parte, com as seguintes informações: **nome do grupo, dia e horário de criação, nome do administrador**
- Não precisa ser necessariamente amigo do administrador do grupo para entrar, todos os grupos estarão abertos, apenas com restrição da chave do grupo para entrar.

list:mygroups

- Lista de todos os grupos o usuário criou
- Deverá conter os nomes dos grupos e as chaves de cada um.

follow <nome_do_usuario>

- Verificar se o usuário especificado existe na lista de amigos (list:friends).
- Se o usuário já estiver sendo seguido, retornar uma mensagem: "Você já está seguindo [nome_do_usuario]."
- Enviar para quem foi seguido a seguinte mensagem: “Você foi seguido por [<[nome_do_usuário]> / <IP>:<SOCKET>]
- Caso contrário, adicione o nome à lista de amigos que o usuário está seguindo e retorne: "[nome_do_usuario] foi adicionado à sua lista de amigos seguidos."

unfollow <nome_do_usuario>

- **Enviar para o usuário que deixou de ser seguido a seguinte mensagem:** [<[nome_do_usuário]> / <IP>:<SOCKET>] “{nome_do_usuário} deixou de

seguir você”

chat_group <nome_grupo> <chave_grupo> <mensagem>

- Verificar se o grupo existe na lista list:groups.
- Validar a chave do grupo para permitir a entrada.
- Retornar mensagens de erro caso o grupo não exista ou a chave esteja errada.
- A mensagem deve ser enviada para todos os outros amigos do grupo, menos pra quem enviou, com a seguinte estrutura:
  [<nome_do_usuário>/<IP>:<SOCKET>]<mensagem>.

chat_friend <nome_amigo> <mensagem>

- Verifique se o nome do amigo está na lista (list:`friends`).
- Se estiver, enviar a mensagem.
- Caso contrário, retornará um erro informando que o amigo não foi encontrado.
- A mensagem deve ser enviada para o amigo com a seguinte estrutura:
  [<nome_do_usuário>/<IP>:<SOCKET>]<mensagem>.

Obs: Se alguma regra, que achar importante, não foi retratada, sentir-se à vontade para discutir com os monitores.
obs: A falta de qualquer ponto dos requisitos das funcionalidades causarão penalidade na nota final.

Instruções adicionais:

Serão postadas atividades no Google Classroom referentes a cada etapa do projeto. A equipe deve realizar **todas** as entregas para que a nota final (soma das 3 etapas) seja validada. Em cada etapa, deverá ser entregue, pelo Google Classroom, uma pasta compactada com os códigos ou um link do github com uma pasta para cada entrega. A atividade deverá ser entregue por cada um dos membros da equipe, nesses termos.

- **Obs:** Cada entrega deve conter, em sua pasta, um readme exclusivo daquela entrega, com instruções de execução e eventuais observações necessárias e o nome dos integrantes. **Comentem o código!**

Cada equipe deve ser composta por, no máximo, 6 alunos. Será disponibilizada uma tabela para a definição dos grupos com data de entrega para 01/12. A nota final do projeto vai compor 25% da média final da disciplina.