# Projeto da Disciplina Infraestrutura de Comunicações - 2025.1

Neste projeto, a equipe desenvolverá um cliente e um servidor de chat de sala única, onde o cliente enviará um pedido de conexão à sala e então passará a receber todas as mensagens dos outros usuários, além de poder enviar mensagens também. O projeto será composto por 3 etapas, em que na primeira etapa o grupo deve desenvolver uma ferramenta de transferência de arquivos utilizando uma comunicação com UDP. Na segunda etapa deverá ser implementado 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 o chat completo com as especificações descritas neste documento.

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: 05/07/2025 às 23:59

**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: 22/07/2025 às 23:59

    - **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: Chat de sala única com paradigma cliente-servidor

- **(4,5 pontos)** Implementação do chat, exibido por linha de comando. Apesar do reaproveitamento das etapas anteriores, o histórico da execução dos algoritmos não deve ser exibido nessa etapa, apenas a aplicação como descrita nesse documento e mantendo o uso do rdt3.0. Prazo máximo de entrega: 12/08/2025 às 23:59

- Obs.: Na terceira entrega do projeto é necessário que o chat funcione para mais de um cliente simultaneamente, ou seja, deverão ser abertos o terminal do servidor e ao menos dois terminais de clientes sem que ocorra interrupção do funcionamento.

- **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 rdt 3.0 que consta no livro "Redes de Computadores e a Internet" do Kurose.**

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

  Cada mensagem aparecerá, no chat público, para cada usuário, no seguinte formato:

<IP>:<PORTA>/~<nome_usuario>: <mensagem> <hora-data>
onde:

- <IP>: Endereço IP do emissor da mensagem
- <PORTA>: Número da porta do emissor da mensagem, do IP descrito acima.
- <nome_usuario>: nome do usuário
- <mensagem>: mensagem recebida
- <hora-data>: hora e data da mensagem recebida, de acordo com o horário do servidor

Um exemplo de mensagem recebida é dado a seguir:
*192.168.0.123:67890/~ana: Alguma novidade do projeto de infracom? 14:31:26 08/08/2025*

Um participante poderá conhecer previamente outro usuário através de uma lista de contatos local de cada um.

Exemplo de lista de contatos:

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

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

| **Funcionalidade** | **Comando** |
|---|---|
| Conectar à sala | **hi, meu nome eh <nome_do_usuario>** |
| Sair da sala | **bye** |
| Exibir lista de usuários do chat | **list** |
| Exibir lista de amigos | **mylist** |
| Adicionar usuário à lista de amigos | **addtomylist <nome_do_usuario>** |

| | |
|---|---|
| Remover usuário da lista de amigos | **rmvfrommylist <nome_do_usuario>** |
| Banir usuário da sala | **ban <nome_do_usuario>** |

Quando um usuário se conectar à sala, os outros usuários deverão receber uma mensagem de alerta da nova presença (ex: João entrou na sala).

Após estar conectado, qualquer mensagem enviada ao servidor será exibida na íntegra para os outros usuários.

Dois usuários não podem se conectar à sala utilizando o mesmo nome.

Adicionar usuário à lista de amigos: Ao usar o comando, você vai adicionar o usuário X à sua lista de amigos e a partir desse momento ele ganha uma tag de [ amigo ]. Então, da próxima vez que ele mandar mensagem no chat tem que aparecer como: **[ amigo ] userX**

Remover usuário da lista de amigos: Ao usar o comando, você vai remover o usuário X da sua lista de amigos e a partir disso ele não possui mais uma tag de [ amigo ]. Então, da próxima vez que ele mandar mensagem no chat tem que aparecer no formato tradicional.

Banir usuário da sala: Ao usar esse comando, o servidor abre uma contagem para banir o usuário, em que caso essa contagem atinja mais da metade dos clientes conectados, o usuário mencionado será banido. Adicionalmente, após cada comando de ban ser enviado, todos os usuários devem receber no chat uma mensagem enviada pelo servidor da seguinte forma: **[ nome_do_usuario ] ban x/y**. Onde x é o número de votos e y é a quantidade necessária para que o usuário seja removido do servidor (um a mais que a metade de clientes conectados durante a abertura do comando).

**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 05/06. A nota final do projeto vai compor 25% da média final da disciplina.