Projeto da Disciplina Infraestrutura de Comunicações - 2022.1

Neste projeto, a equipe desenvolverá um cliente de chat e um servidor de chat de sala única. 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.

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

<hora><nome_usuario>: <mensagem>
onde:

- <nome_usuario>: nome do usuário
- <mensagem>: mensagem recebida
- <hora>: hora da mensagem recebida, de acordo com o horário do servidor

Um exemplo de mensagem recebida é dado a seguir.
14:31:47 ana: Alguma novidade do projeto de infracom?

O socket UDP deverá contar com transmissão confiável para todos os terminais (cliente e sala de chat), implementada em camada de aplicação segundo o rdt3.0 que consta no livro "Redes de Computadores e a Internet" do Kurose.

Cada usuário conectado deverá ser cadastrado na tabela de chat, armazenada no servidor de chat, assim como quem sair deve ser deletado dela. Exemplo de tabela de chat:

Nome	**IP:PORTA**
Felipe Maltez	128.65.27.104:5000
Vitor Azevedo	195.143.1.171:5500

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_de_usuario>**
Sair da sala	**bye**
Exibir lista de usuários	**list**
Mensagem particular (inbox)	**@<nome_de_usuario> <mensagem>**
Expulsão	**ban @<nome_de_usuario>**

- Quando um usuário se conectar à sala, os outros usuários deverão receber uma mensagem de alerta da nova presença (ex: Dario entrou na sala).
- Após estar conectado, qualquer mensagem enviada ao servidor será exibida na íntegra para os outros usuários, exceto o comando de inbox, dado que é uma mensagem particular, e o comando de conectar-se à sala.
- O comando list exibirá a lista dos nomes dos usuários conectados, sem expor seus IPs.
- O comando mensagem particular fará com que apenas o usuário referido receba a mensagem em sua tela.
- É importante que ao enviar uma mensagem ela deve chegar apenas para os outros usuários, uma vez que o que foi escrito já fica presente no seu próprio terminal.
- Cada usuário deve possuir um contador de bans. O comando ban deve receber o nome do usuário que se deseja banir e somar um ao contador daquele usuário. Se o contador de bans do usuário chegar a ⅔ da quantidade de usuários conectados, o usuário é desconectado da sala, ou seja, seu nome entra numa tabela de exclusão. O usuário que está com o nome na tabela de exclusão não pode ter conexões aceitas à sala. O comando ban só pode ser chamado de X em X segundos.

A entrega será dividida em 3 etapas, implementadas gradualmente (uma sobre a outra):

- **(3,0 pontos)** Implementação de cliente e servidor UDP comum utilizando a biblioteca **Socket** na linguagem **Python**, com envio de arquivo (daremos uma pasta de arquivos que devem funcionar) e devolução. Prazo máximo de entrega: 30/08/2022 às 23:59
- **(3,0 pontos)** Implementação de confiabilidade com uma checklist de confiabilidade, segundo o canal de transmissão confiável rdt3.0. Prazo máximo de entrega: 29/09/2022 às 23:59.
- **(4,0 pontos)** Implementação do chat, exibido por linha de comando. Prazo máximo de entrega: 28/10/2022 às 23:59

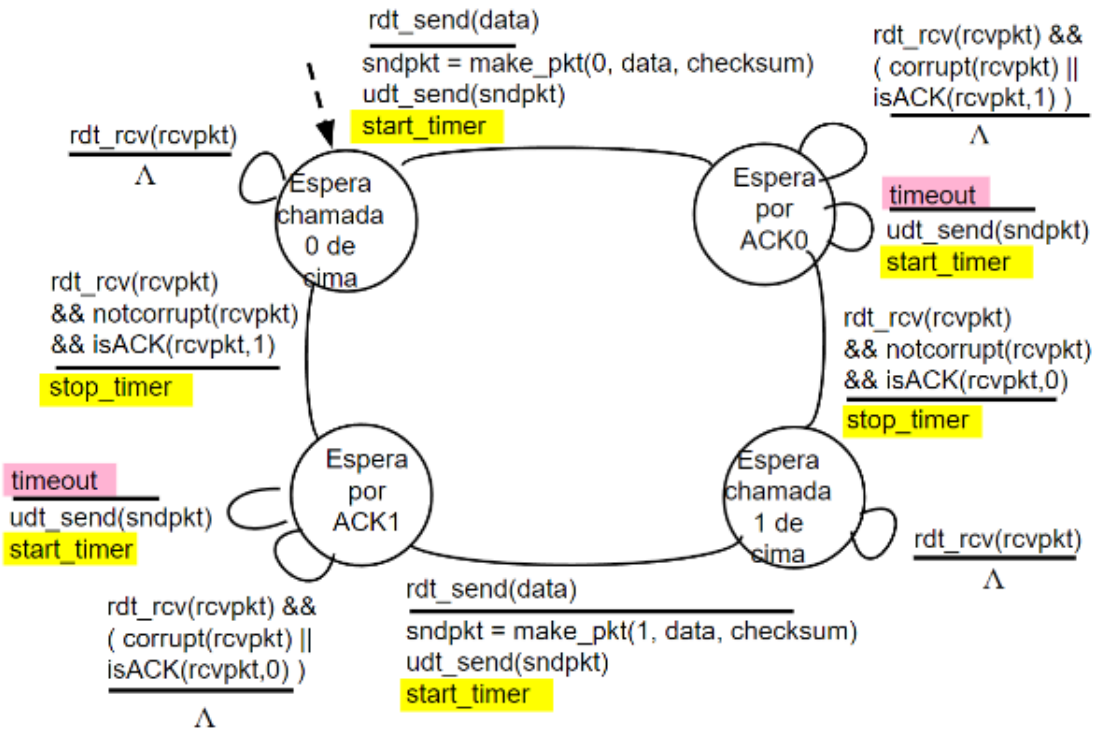
Serão postadas atividades no Google Classroom referentes a cada etapa do projeto. A equipe deve realizar ***todas*** as entregas para que a nota final (soma das 3 etapas) seja validada. Em cada etapa, deverá ser entregue, pelo Google Classroom, uma pasta compactada com os arquivos necessários ou o link do github com uma pasta para cada entrega.
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 seu funcionamento. Todos os integrantes do grupo devem participar
Cada equipe deve ser composta por, no máximo, 4 alunos. Será disponibilizada uma tabela para a definição dos grupos com data de entrega para 20/08/2021. A nota final do projeto vai compor 30% da média final da disciplina.

Atenção!

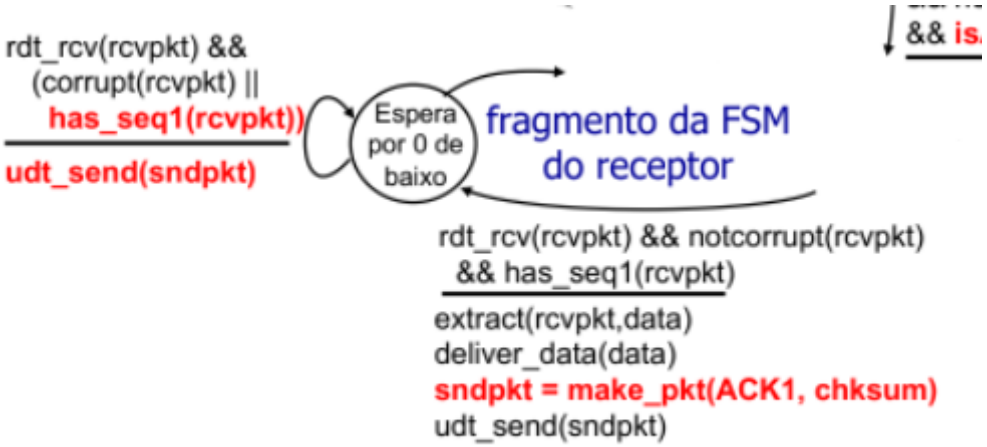
- Um readme deve estar na pasta em cada entrega, com instruções de execução e eventuais observações necessárias. Códigos muito complicados acarretarão em convocação da equipe para desvendar o mistério. COMENTEM!!!
- Não usem ip fixo como argumento para o socket! Usem “localhost” (a string mesmo) ou achem através de alguma função. A porta é com vocês.

rdt3.0

Considerando uma situação totalmente real, onde o canal pode perder dados no meio do caminho, é implementado um temporizador no transmissor, que retransmite o pacote caso nenhum ACK seja recebido no intervalo de tempo definido. O receptor permanece o mesmo do rdt2.2.



- Transmissor: Quando recebe uma chamada da camada de cima, cria o pacote com os dados e o número de sequência, calculando o seu checksum (make_pkt) e envia o pacote (udt_send), iniciando o temporizador. Depois que faz o envio, fica esperando a resposta do receptor (rdt_rcv), caso receba um ACK com o número de sequência errado ou uma resposta corrompida, faz o envio novamente do pacote. Caso receba um ACK com o número de sequência correto, altera o número de sequência para o próximo pacote. Caso o temporizador estoure, é feita a retransmissão do pacote e o temporizador e reiniciado.



- Receptor: Quando recebe os dados do transmissor (camada de baixo), primeiramente checa se os dados e o número de sequência estão corretos, caso esteja com erro, manda um ACK e o checksum do seu último pacote recebido para o transmissor, para que ele faça o reenvio quando receber um número de sequência errado. Caso esteja tudo certo, extrai os dados do pacote, entrega para o socket apropriado (deliver_data) e envia um ACK e o checksum do seu último pacote recebido (Que é o próprio pacote em questão) para o transmissor. Se tudo ocorrer corretamente, altera o número de sequência para o próximo pacote.