Relatório

do

Trabalho Prático

Redes de computadores 2

Aluno: Alan da Silva Gomes

Professora: Debora Christina Muchaluat Saade

Monitor: Rômulo Augusto Vieira Costa

Objetivo

Implementar um servidor socket usando o protocolo TCP/IP para o envio de mensagens de controle entre os usuários/clientes da rede.

Cada usuário se conecta ao servidor através do par (IP, PORTA) e através desse consegue enviar mensagens para os demais usuários conectados a esse mesmo servidor. Assim também como mensagens broadcast e solicitar a lista de informações de cada usuário conectado.

O que foi implementado

O servidor socket foi implementado e possibilita a troca de mensagens entre os usuários/sockets conetados.

- Listar os usuários conectados por uma tabela dinâmica.

- Mandar mensagens broadcast.

- Verificar a unicidade dos usuários. Cada usuário possui um identificador único que é o seu nome. E esse atributo fica armazenado na tabela dinâmica.

Se faz a verificação da singularidade através do nome do usuário, logo cada usuário deve possuir um nome diferente. Dessa forma não é possível haver mais um de usuário usando o mesmo nome, mesmo que a sua porta seja diferente.

- Manda uma mensagem privada para um usuário específico.

- O usuário consegue se desconectar.

- O usuário consegue fechar o servidor.

- O usuário consegue enviar uma mensagem de “eco”.

- Fazer uma ligação diretamente para um usuário em específico, não havendo a necessidade dos dados passarem pelo servidor.

- Interface de log do server. Todas as mensagens do server sao armazenadas em uma pasta contendo todos os arquivos log. txt .

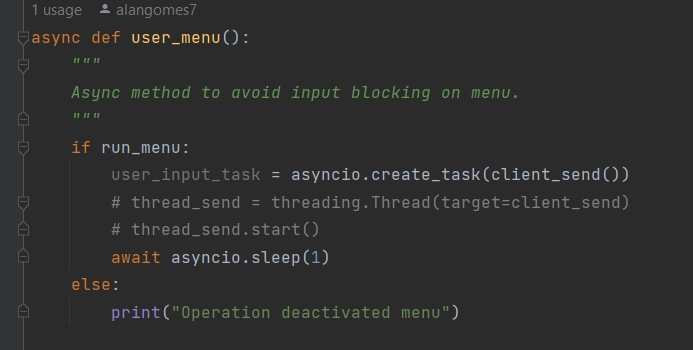
Implementação do código

Temos três arquivos de códigos implementados na linguagem python3. Um para a implementação do servidor, outro para a implementação do cliente e outro somente para o tratamento de mensagens (encaminhamento, codificação e decodificação).

Tive bastante dificuldade na hora de implementar o código, pois se tratava de mensagens entre redes e a solução do uso de threads nem sempre funcionou. Logo tive que implementar as seguintes abordagens:

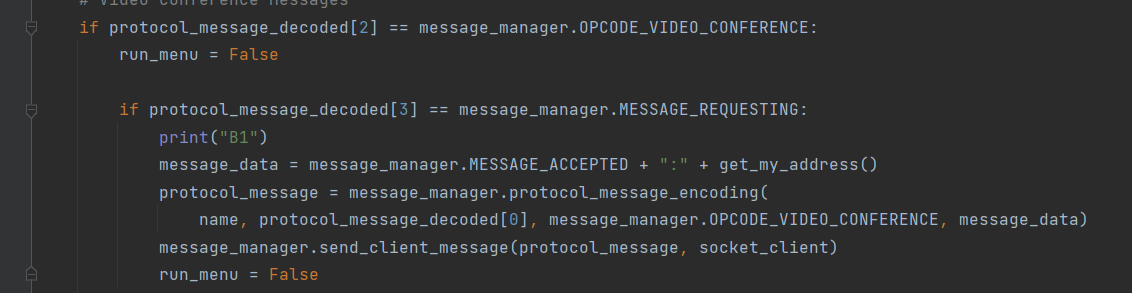
Async:

A operação mais bloqueante era a de esperar o usuário digitar alguma entrada. Para resolver esse problema, foi colocado um async e uma variável de controle para bloquear/encerrar essa thread.

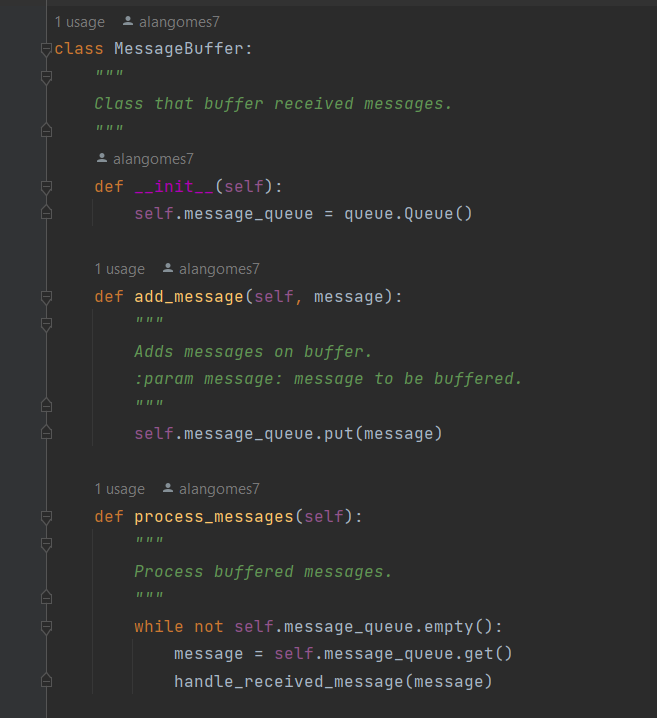


No script do cliente temos um assync para rotar ou não a task client\_send(). Essa task exibe o menu e espera uma entrada do usuário.

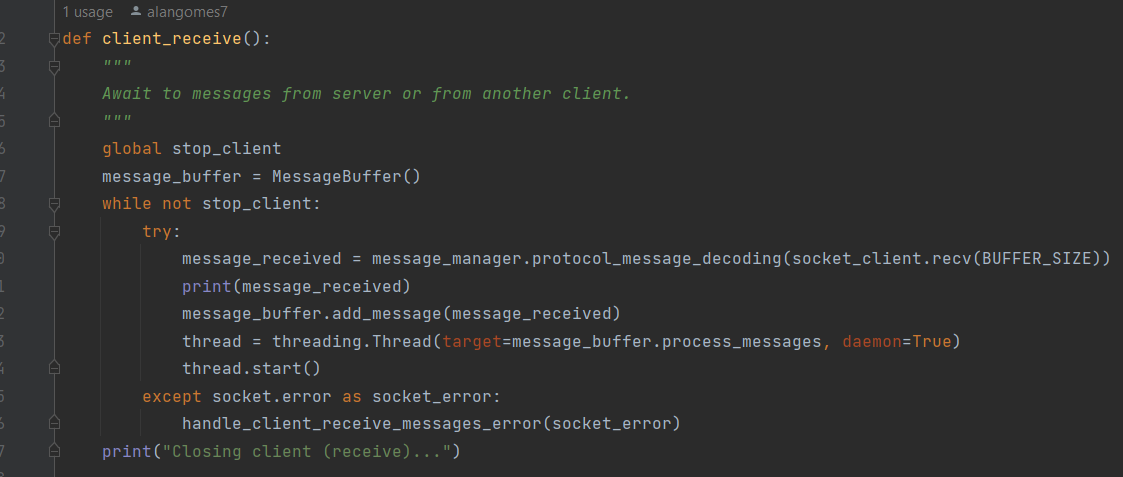
Também temos uma variável de sinal, ela irá dizer se o menu deve ou não ser executado de acordo com a operação recebida:



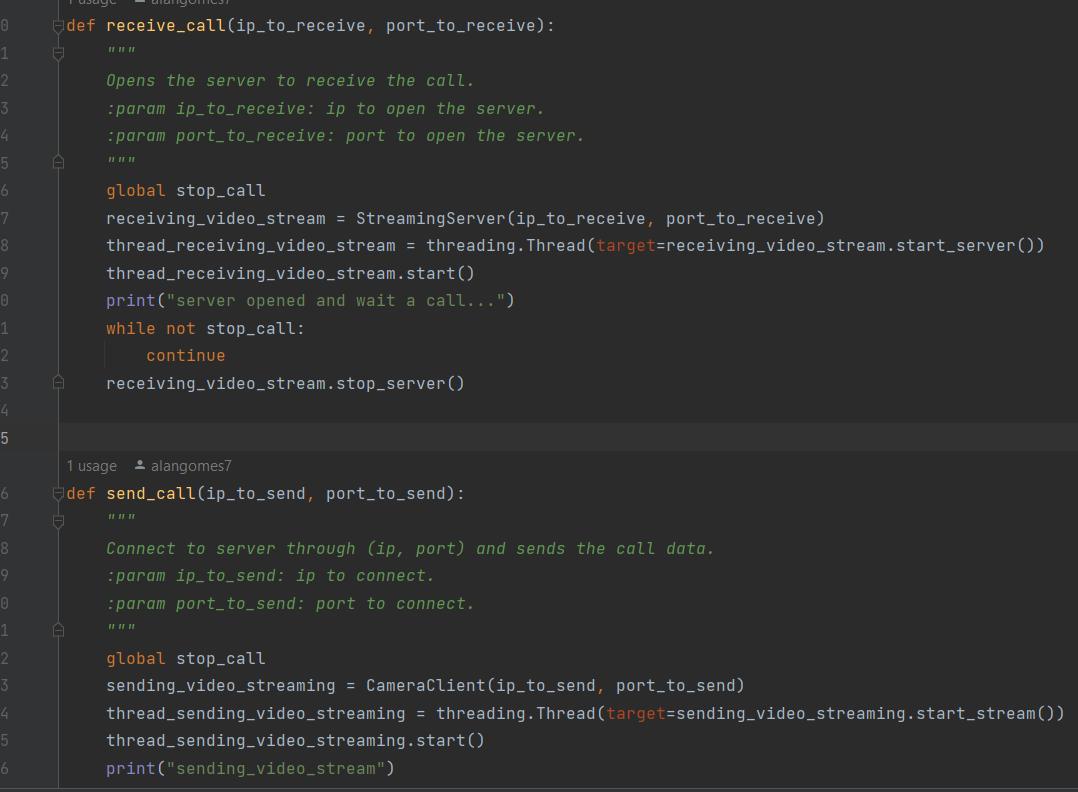
Para não perder as mensagens que poderiam chegar no cliente enquanto este executa outra ação, foi criada uma fila de mensagens. Dessa forma, evitamos de perder mensagens:



Essa classe está no script do cliente e ela faz o buffer de todas as mensagens que chegam. A partir desse buffer de mensagens o script do cliente executa as ações. Essa função está sempre esperando por mensagens. Ela recebe as mensagens e as coloca no buffer. Após esse recebimento elacria uma thread para processar a mensagem recebida.



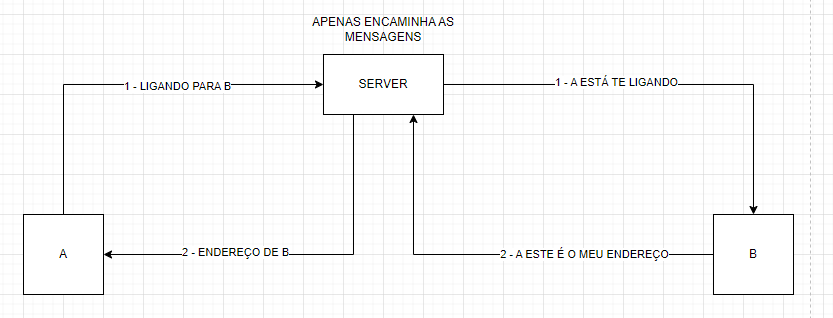
Agora falando dos métodos para a video chamada. Foram criadas duas funções através do videostream que recebem e enviam os dados de vídeo.



Para que fosse possível garantir o envio de mensagens para um servidor existente foi pensado o seguinte pipeline:

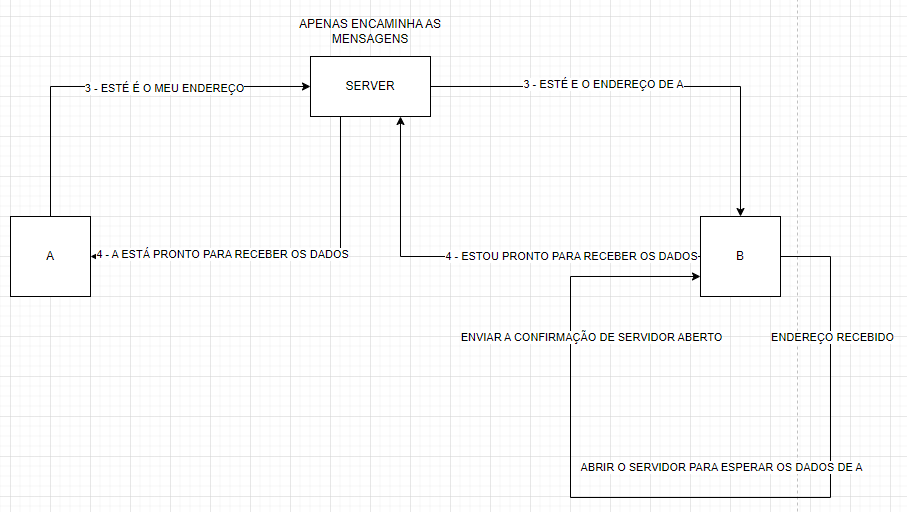
1 – O cliente A liga para B

2 – B recebe a ligação e envia o seu endereço para A

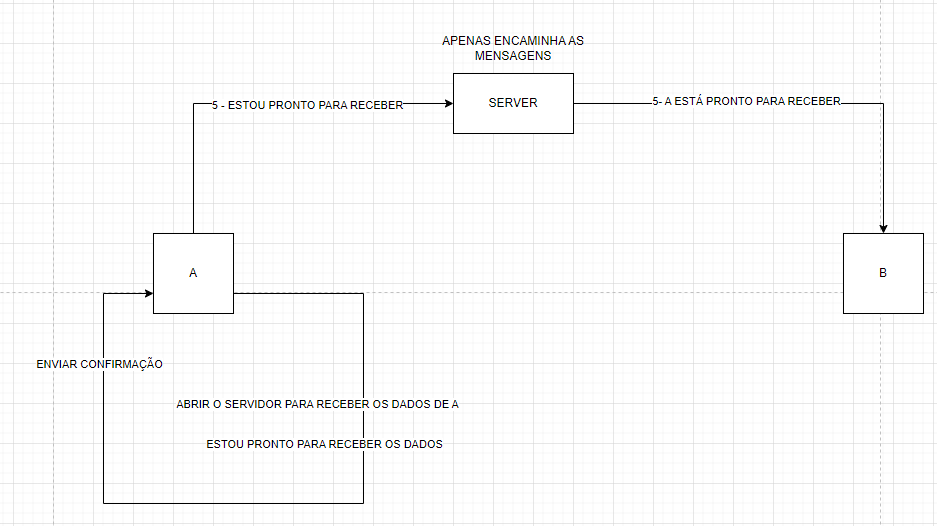


3 – A responde com o seu endereço

4 – B recebe o endereço, abre o servidor e espera por A e envia a confirmação que está pronto para receber os dados de A

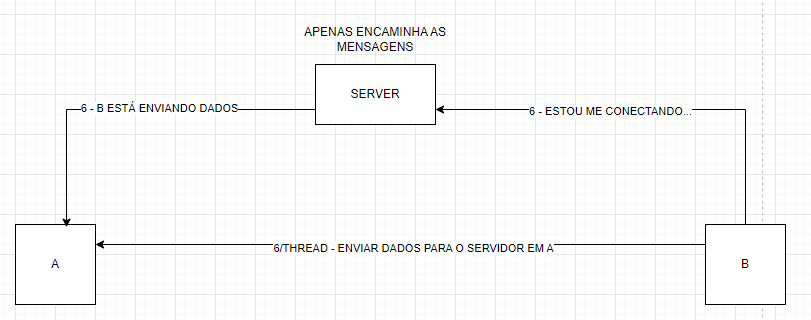


5 - A abre o servidor e envia a confirmação que está proto para receber os dados de B

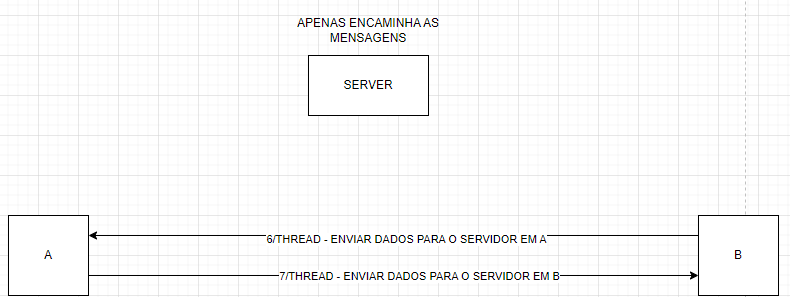


6 – B envia uma confirmação para A indicando que irá enviar os dados

6 – B dispara uma thread e envia os dados para A



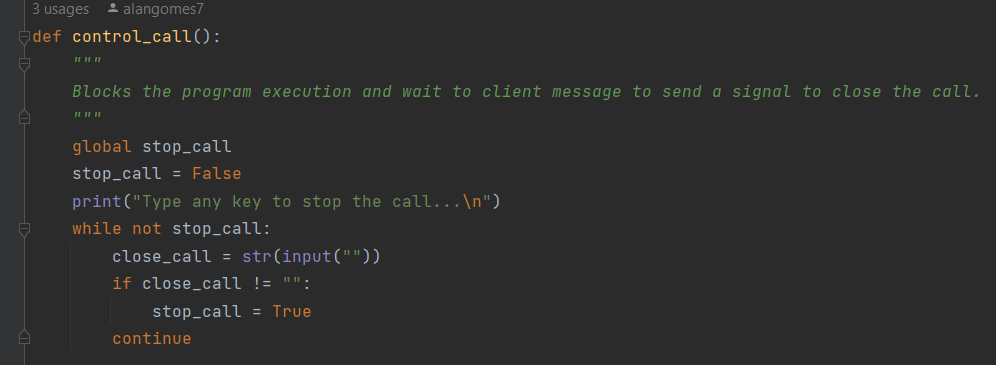
7 – A recebe a confirmação de que chegará dados e também envia os dados para B



Nessa operações um cliente fica esperando a mensagem de outro cliente. De acordo com a mensagem recebida a conexão da ligação prossegue até a ligação acontecer. Quando ocorre a ligação ela é feita diretamente entre os clientes através do (IP/PORTA). Nesse momento os dados não passam mais pelo servidor.

Para estabelecer uma conexão enquanto os cliente executam outra tarefa como o

envio/o recebimento de ligação, foram usadas threads. Na ligação não foi necessário usar Async, pois a operação de transferência de dados não foi tão bloqueante como o menu.



Para encerrar a ligação basta que um dos clientes digite qualquer coisa no prompt de comando para encerrar as threads de transferência de dados da ligação.

Para executar o programa e testar a ligação é necessário usar dois computadores diferentes. Não consegui testar usando apenas uma mesma máquina por causa da disponibilidade de recursos físicos (câmera, mic).

Na interface do server, tive muitos problemas com a interface congelando. Para resolver isso alterei o botão para após iniciar o servidor, ter apenas a opção de fechar o programa. Dessa forma o programa fecha e o servidor também.

Server interface:

