Trabalho 1 de Sistemas Distribuídos

Abraham Banafo Ampah - 117074396

Cristian Diamantaras Vilela - 118109047

Gilberto Lopes Inácio Filho - 115173699

Ricardo Kaê Bloise - 116039521

16 de junho de 2022

Introdução

O trabalho realizado pelo grupo consiste numa aplicação de comunicação instantânea distribuída de acordo com os elementos do projeto definidos em acordo com a turma. Foram estabelecidas uma arquitetura de software, de sistema e um protocolo de camada de aplicação que procuramos seguir na implementação da aplicação.

Funcionamento

A aplicação é composta por um programa para a execução pelos usuários, um para o servidor central, além de utilitários para a exibição do texto utilizando formatação rica no terminal.

O usuário é capaz de se conectar ao servidor central, se registrando como ativo. Também é capaz de mandar mensagem diretamente para outros usuários que estejam ativos no mesmo servidor central, além de receber mensagens destinadas a si.

A interface que implementamos entre o usuário final e as funcionalidades disponíveis no programa executa diretamente no terminal, usando formatação rica de texto para destacar mensagens de serviço, instruções e informações sobre comando, e destacar o identificador em uma mensagem que representa o remetente. Toda a interação é feita através de comandos de texto, e todas as informações geradas no programa são exibidas da mesma forma.

Sendo assim, abaixo encontram-se algumas imagens que exibem esse funcionamento da aplicação, quando três usuários em máquinas distintas desejam-se comunicar por esse chat.

Tais usuários logam-se sobre os usernames: @Usuário A, @Usuário B e @Usuário C. Sendo postas as figuras 2,3 e 4, para exibir seus respectivos logs. Bem como, a figura 1 para exibir o log do Servidor Central e a figura 5, de um usuário D, para ilustrar o menu de comandos da aplicação.

Figura 1. Log do Servidor Central

Figura 2. Log do Usuário A

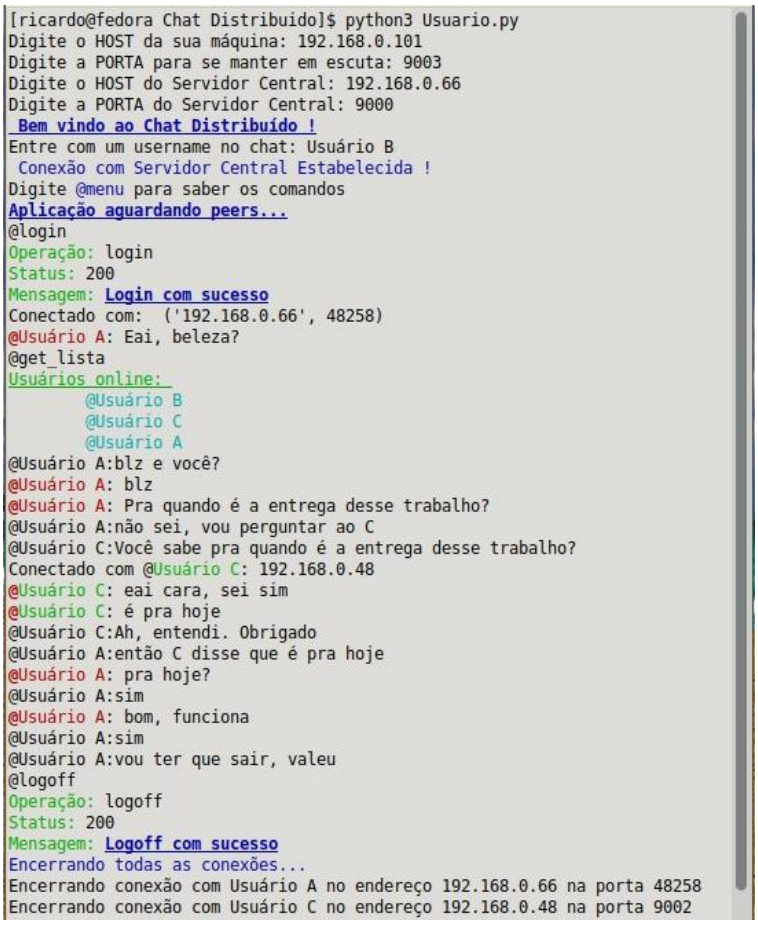

Figura 3. Log do Usuário B

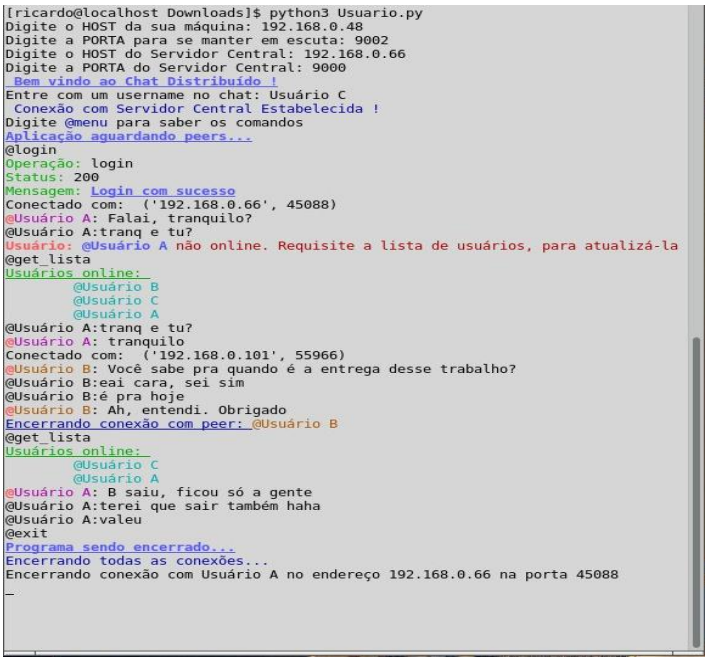

Figura 4. Log do Usuário C

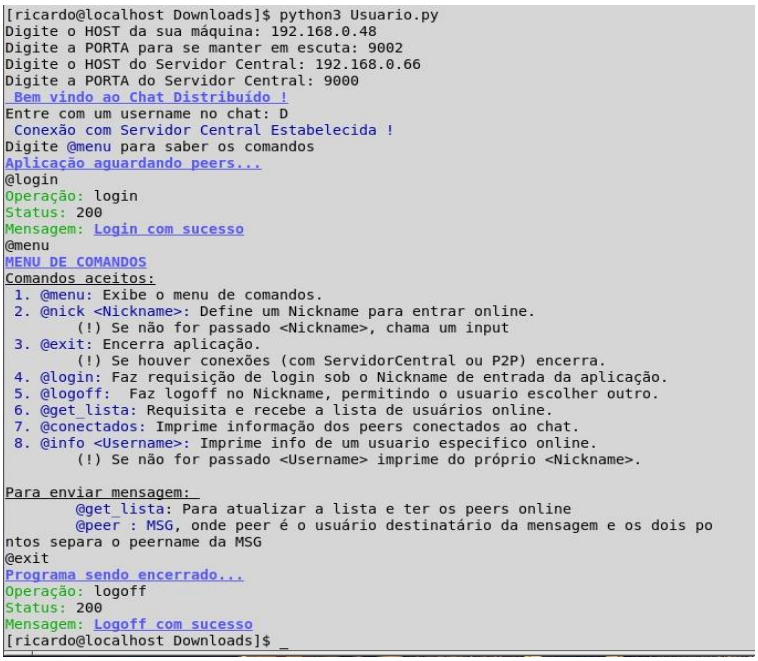

Figura 5. Menu de comandos em D

Pela análise da Figura 2 (log do Usuário A) pode-se perceber grande parte do funcionamento da aplicação. Sendo as outras figuras, apenas um auxílio a essa análise.

Na Figura 2 percebe-se que quando não há usuários logados no Servidor Central, o comando **@get_lista** retorna uma lista vazia (primeiro comando dado após a aplicação de A ter iniciado e estar aguardando *peers*). Quando B e C dão **@login**, eles se registram (conectam) no Servidor Central (como pode ser visto na Figura 1), logo ficam disponíveis para receberem conversas.

Com isso, o Usuário A pede a lista novamente e inicia uma conversa com o Usuário B e com C. Como A não tinha ainda efetuado o **@login**, a aplicação foi robusta o suficiente para forçar a entrada de A no Servidor Central e assim, permitir a troca de mensagens entre esses *peers*.

Na sequência, o Usuário A pôde executar o comando @info, um dos comandos disponíveis pela aplicação (ilustrados na figura 5) e imprimir informações suas na mesma tela em que recebe as mensagens de B e C, pois como mencionado no relatório do projeto deste trabalho, o objetivo do grupo era desenvolver uma aplicação de janela única, com todas as informações sendo impressas na mesma tela.

Os usuários A,B,C puderam continuar conversando até que B e C fizeram, respectivamente, **@logoff.** (linhas finais das Figuras 3 e 4), onde tais deslogamentos foram comunicados no terminal de A e no Servidor Central, permitindo assim A finalizar a aplicação corretamente.

Vale destacar ainda, que na Figura 4, no log de C, assim que B faz **@logoff** e C pede a lista, o usuário B já não aparece mais, indicando que ele foi desconectado do Servidor Central corretamente.

E assim nota-se, portanto, que a funcionalidade da aplicação atende os requisitos especificados. Ela é capaz de enviar requisições e receber respostas do Servidor Central e ainda trocar mensagens com outros usuários ativos (online) no mesmo Servidor Central.

Ajustes

Em relação ao projeto da aplicação, relatado na 1º etapa deste trabalho, foram feitos em essência 3 grandes ajustes:

- Ajuste do funcionamento do Servidor Central
- Ajuste na comunicação P2P do Usuario (threads e socket)
- Comando de envio de mensagem para os *peers*

Fora a funcionalidade acrescida na interface que permite um usuário trocar seu username, sem a necessidade de sair da aplicação. Para tal funcionalidade (**@nick** <username>) não tinha sido atribuído um comando no menu da aplicação, quando o projeto havia sido elaborado. Sendo assim, esse foi um pequeno ajuste que foi acrescido à medida que o projeto se desenrolava.

Para os demais ajustes, teve-se:

Funcionamento do Servidor Central.

Na etapa de projeto, não havia sido elaborada uma proposta para o desenvolvimento do Servidor Central. Tanto que no 1º relatório, o grupo descreveu esse componente de forma muito resumida, já que tinha-se somente, nessa etapa, uma ideia básica de como deveria ser seu funcionamento.

Durante o desenvolvimento da aplicação, o grupo tomou a decisão de desenvolver um servidor concorrente, que designa uma thread por conexão (socket) estabelecida, podendo então atender requisições e enviar respostas desse/para socket, sendo portanto um ajuste no projeto inicial.

Comunicação P2P do Usuario (Threads e Socket).

No diagrama estrutural do relatório do projeto, menciona-se a existência de um componente modular chamado Servidor Peer, que seria responsável por aguardar conexões de outros *peers* na classe Usuario. Na descrição que havia sido feita, tal componente modular seria executado, concorrentemente, em threads diferentes da thread principal, na qual refere-se a thread em que o usuário pode executar comandos na interface da aplicação. Sendo, portanto, designado no projeto, threads para se manter em modo passivo, aguardando por conexões (de outros clientes, usuários ou *peers*).

Acontece que durante a implementação da aplicação, percebeu-se que esse foi um equívoco do grupo e que não havia necessidade de se ter threads distintas aguardando por conexões. Logo, o ajuste que foi feito quanto a isso foi manter esse modo passivo na mesma thread principal da aplicação e assim que uma conexão chega, designa-se apenas uma thread para receber as mensagens (executar o recv) vindas desse peer que se conectou.

Logo, ao contrário do que havia sido escrito: ao invés de se ter 3 tipos de threads em Usuario, representadas pelas cores verde, laranja e vermelho no diagrama estrutural, passou-se a ter somente 2: as verdes e as vermelhas. A thread principal e uma classe de threads para receber mensagens.

Além disso, outro ajuste que foi feito na organização P2P do programa foi quanto ao socket usado para enviar e receber mensagens. No diagrama estrutural, ilustra-se apenas um único socket para enviar e receber (executar o recv - sob ação de uma thread) mensagens do/para o usuário. Enquanto que na prática, utilizou-se dois: um para atuar em modo ativo e se conectar a um peer online no Servidor Central e outro para aguardar conexões e designar um novo socket, responsável por executar o tal recv sob uma nova thread lançada.

<u>Comando de envio de mensagem para os peers.</u>

Este ajuste não foi tão grande quanto aos outros dois citados mais acima. Na realidade, em termos de código, o ajuste dele foi menor do que a funcionalidade acrescida no comando **@nick**, mas vale destacá-lo como um grande ajuste por dois motivos:

1. A alteração foi feita no comando mais importante do menu, aquele no qual o usuário envia mensagem para os outros *peers.* Antes, no projeto, o envio de mensagens seria feito da seguinte forma:

   ~~@peer mensagem~~

   Agora com o ajuste, as mensagens são enviadas conforme abaixo:

   @peer : mensagem

   onde o símbolo ":" separa o nome do peer (destinatário) da mensagem a ser enviada.

2. Tal ajuste permitiu a conexão com *peers,* com nomes de usuários compostos (incluindo o uso de espaços), que foram escolhidos por muitos grupos da turma.

Testes

Durante as rodadas de teste, realizamos diversos ajustes também, mas não quanto ao projeto inicial, como foi descrito na seção anterior, mas ajustes quanto a implementação das versões da nossa aplicação, até a versão final, que se mostrou razoavelmente bem na última rodada de teste, em que pudemos nos conectar e trocar com mensagens com praticamente todos os grupos lá presentes, onde pode-se citar alguns membros: Thierry, Lorena, Rufino/Luiz e Nando.

Apresentando um pouco desses ajustes.

1. Na primeira rodada de teste, o grupo não tinha entendido a ideia dos dois primeiros bytes para sinalizar o tamanho das mensagens enviadas e recebidas pelos outros clientes e servidores da turma, impedindo a nossa conexão com os mesmos e motivando demandas a serem ajustadas pelo nosso sistema.
2. O grupo conseguiu acertar os dois primeiros bytes, mas havia pouco tratamento a falhas nos métodos de recebimento de mensagens, sejam vindas dos servidores da turma ou dos clientes. O que ainda dificultava muito a comunicação com outros usuários e fazia a aplicação quebrar muito.
3. O grupo conseguiu tornar tais métodos de recepção e envio de mensagens bem robustos, fazendo uma reorganização interna de suas funcionalidades, incluindo métodos auxiliares e cláusulas try-except, que permitiram segurança na conexão com outros usuários da turma e a prevenção contra bugs e o travamento da aplicação, quando exceções (ou erros) ocorriam.

Logo, para mostrar os efeitos de tais ajustes, o grupo apresenta uma imagem abaixo de três aplicações se comunicando: a aplicação do grupo do Nando, do grupo do Rufino/Luiz e a nossa, rodando em máquinas distintas no servidor do Rufino.

Em tais figuras, pode-se notar que as três aplicações funcionam quanto ao envio e recebimento de mensagens, iniciando conversas (modo ativo) e recebendo conversas (modo passivo)

Aplicação do grupo do Rufino

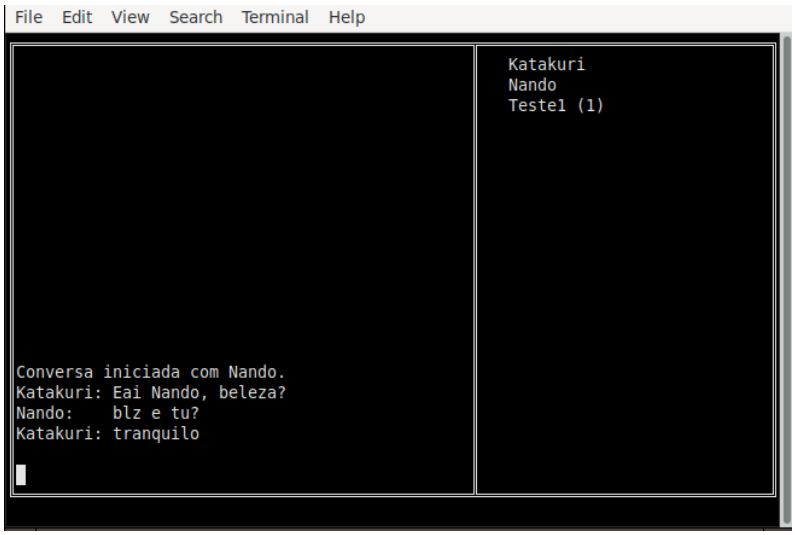

Figura 6. Iniciando conversa com Nando

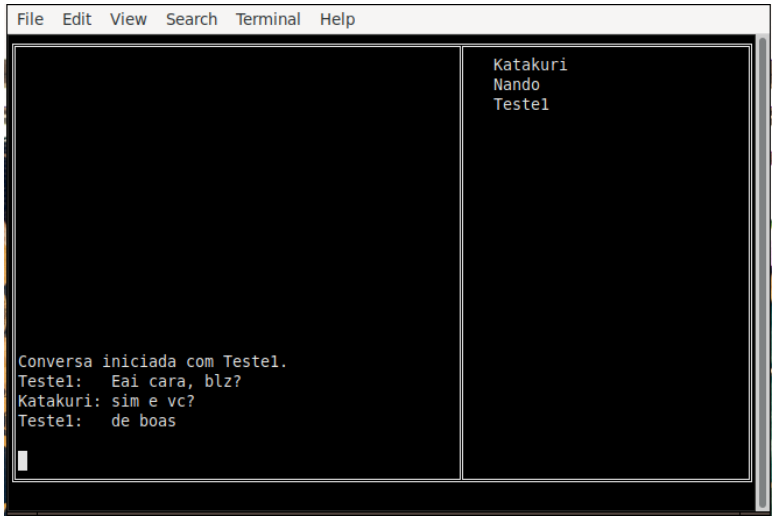

Figura 7. Recebendo conversa da nossa

Aplicação do grupo do Nando

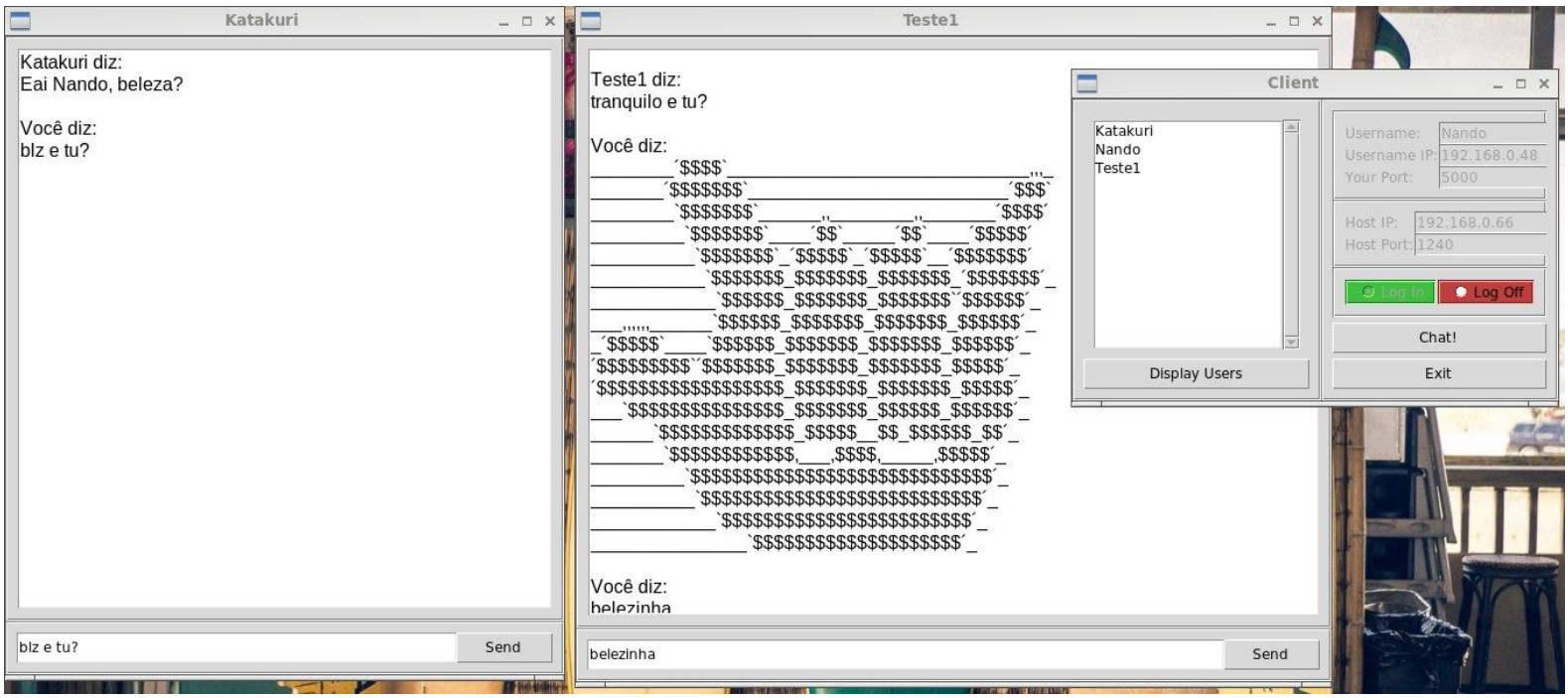

Figura 8. Ao lardo esquerdo, mostra a recepção da aplicação quando alguém inicia o chat com a mesma. E ao lado direito, a aplicação do Nando inicia o chat com a nossa.

Aplicação do grupo

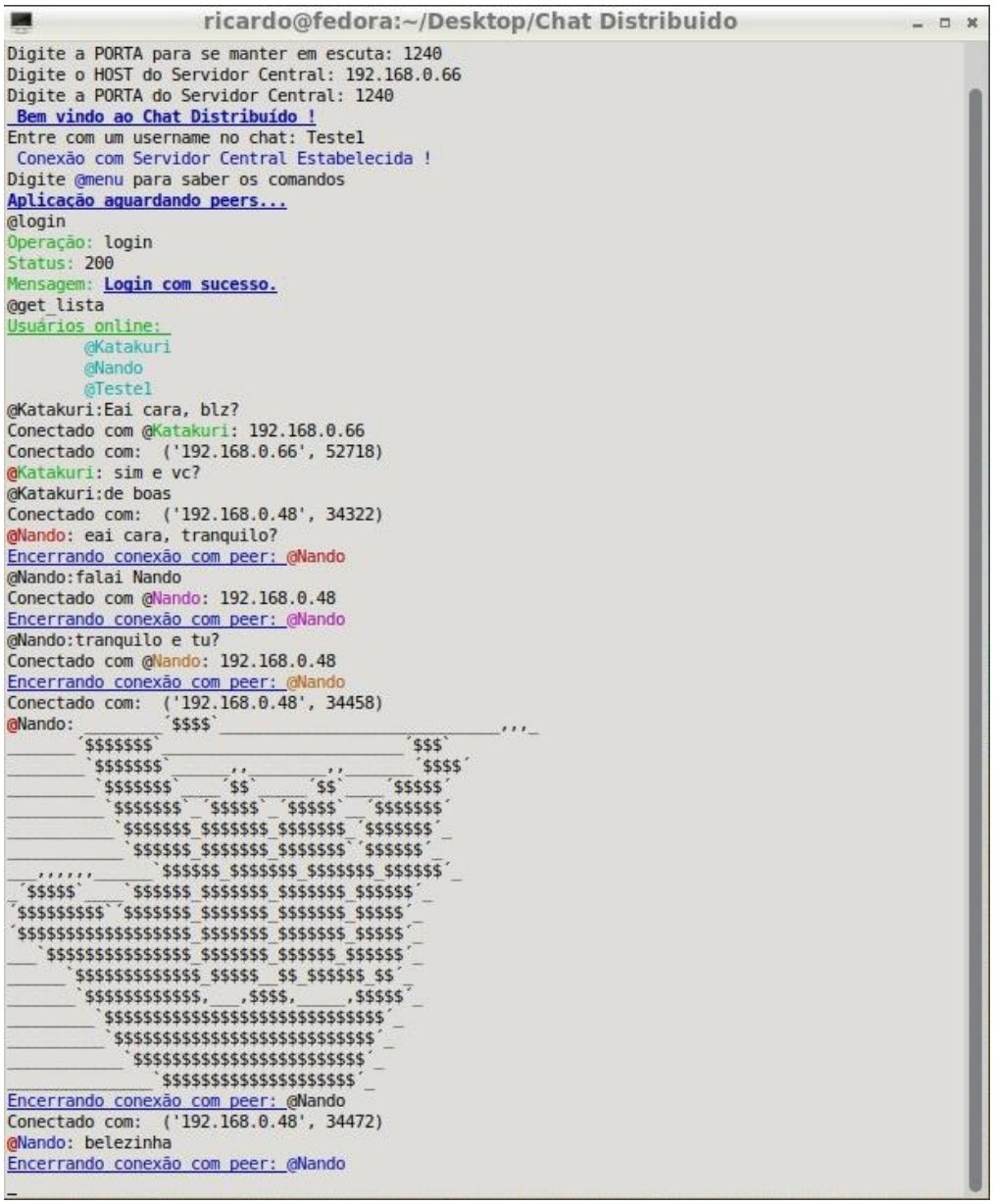

Figura 9. Aplicação do grupo

A Figura 9 mostra a aplicação do grupo enviando e recebendo mensagens, onde @ (o arroba vermelho) ilustra mensagens recebidas e pode-se notar que o grupo do Nando, de fato, iniciou uma conversa com a aplicação desse trabalho. E o @ (arroba em preto) precedendo o nome do usuário ativo, indica mensagens enviadas pela aplicação e então nota-se que iniciamos a conversa com o grupo do Rufino (usuário Katakuri).

Conclusão

Apesar de dificuldades com aspectos particulares do Python, além de imprevistos de saúde com um dos membros, a execução do trabalho nos permitiu um grande aprendizado sobre conceitos de sistemas distribuídos, principalmente em relação aos desafios de se projetar e implementar um protocolo de aplicação ad hoc, sem a utilização de middlewares mais robustas para solucionar problemas clássicos da comunicação entre processos através do TCP.