PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
(PUCRS)

ESCOLA POLITÉCNICA

Jonathas Hernandes e Wagner Santos

Programação Distribuída - P2P

Porto Alegre

2021

Sumário

1 Introdução

Para o segundo trabalho da cadeira de Programação Distribuída, foi solicitado o desenvolvimento de uma aplicação p2p. A principal característica de tal modelo de arquitetura se dá por cada um de seus pontos funcionarem tanto como cliente quanto por servidor, permitindo compartilhamento de serviços e dados, sem a necessidade de um servidor central. Diferente de sistemas centralizados, os sistemas P2P oferecem recursos localizados dentre todas as máquinas conectadas em sua rede, de forma não centralizada.

O projeto desenvolvido utiliza-se de tal modelo de arquitetura. Através de dois tipos de nós denominados super nodos, os demais nodos conectados entre si através destes são capazes de trocar recursos sem a necessidade de um agente centralizador para gerenciar a transação entre eles. Embora tais super nodos sejam responsáveis pela centralização dos recursos disponíveis entre os diversos nós conectados, a conexão existente entre eles se dá de forma totalmente autônoma, não precisando sequer que ambos estejam ligados a rede a partir do mesmo super nodo.

Para o desenvolvimento da aplicação, optou-se pela utilização da linguagem Python. A escolha desta para o presente trabalho se deu pela familiaridade dos membros do grupo com a linguagem, bem como a simplicidade que este apresenta. Além destas benesses, compreende-se também a facilidade que existe a utilização desta para o desenvolvimento utilizando sockets. Percebendo ser esta uma parte central do desenvolvimento, procurou-se uma opção que tornasse viável o desenvolvimento em alto nível, comparado com outras opções como C/C++ onde é necessária a compreensão profunda do desenvolvimentos de pacotes de rede a nível de bits.

2 Arquitetura

Conforme descrito anteriormente, o projeto tem como principal característica a presença de dois tipos diferentes de nós. Dentre eles cita-se a presença dos denominados super nodos, capazes de manter a conexão com os demais nodos, bem como serem capazes de conversar entre eles. Estes são responsáveis por manter um controle dos recursos e nodos ativos entre a rede, fazendo a ponte entre os nodos comuns, denominados clientes, entre os recursos e endereços dos demais conectados. Já os nodos comuns, conectados a rede, fazem o papel de clientes. A partir da conexão com um dos diversos super nodos, são capazes de realizar operações, como informar os recursos disponíveis ou requisitar um

recurso específico.

Na figura 1 podemos observar um modelo da rede a ser desenvolvida a partir da arquitetura descrita até o momento. Como pode-se observar, esta parte de super nodos centralizadores, capazes de se comunicarem entre si, bem como a ligação de nodos clientes a estes. Da mesma maneira, existe também a possibilidade de, em um determinado momento, a fim de ser realizada a transação de recursos.

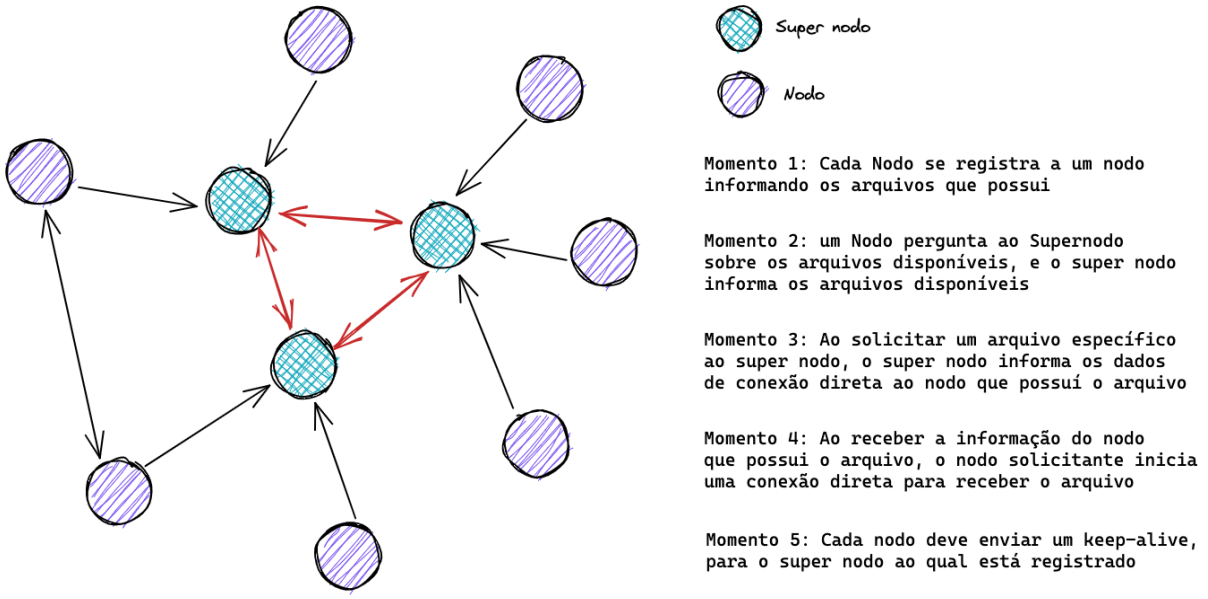

Figura 1. Diagrama da arquitetura contendo Nodos e Super Nodos

3 Super Nodos

Os super nodos se conectam entre si e entre os nodos através de conexões Socket. A partir de cada uma das conexões realizadas com ele, mantém uma thread capaz de oferecer seus serviços caso sejam solicitados pelos clientes conectados. Para tais serviços, foi desenvolvido um protocolo próprio, cujo programa é capaz de realizar o parse. Tendo em seu cabeçalho a palavra chave realizada no serviço, este é capaz de direcionar a requisição para o método cujo qual atende a solicitação atribuída. Entre eles, inclui o retorno da lista de nodos conectados, a busca de um arquivo através de seu hash, bem como a adição de um arquivo a partir de seu nome e hash e, por fim, um método de heart beat, a fim de manter o controle dos nodos ainda conectados.

3.1 Gerenciamento de requests recebidos

Após a iniciação do super nodo, este fica a aguardar a conexão de demais nodos, a serem realizadas a partir da implementação de sockets. Após o bind do socket ao super nodo, este gera uma thread para o nodo que acabou de se conectar, onde aguarda as mensagens deste. A partir do pacote recebido, é realizado o parsing, para então ser ativado o método referente ao tipo de chamada definido pela primeira linha de dados recebida.

4 Nodos Cliente

Estes são capazes de se conectar a um determinado super nodo a partir de um socket, atribuindo-lhe o caminho e a porta para a conexão. A partir deste, é apresentado ao usuário as diversas operações válidas a partir do nodo com um super nodo. Além destes, existe também a possibilidade de se conectar brevemente a um outro nodo da rede, a fim de realizar o recebimento do arquivo solicitado. Além disso, durante a execução do programa, este envia em intervalos periódicos mensagens de heartbeat para o super nodo cujo qual está conectado, informando que permanece "vivo" na rede.

5 Descrição de processos

5.1 Enviar arquivo:

A fim de enviar um arquivo, a primeira operação realizada pelo nodo é o processo de digestão do hash. Utilizando a biblioteca hashlib, do Python, é criada uma instância do algoritmo de hash SHA-256. Tendo este em mãos, é realizado o processo de leitura do arquivo por blocos, que serão passados ao algoritmo, que será atualizado com o novo valor, para, no final deste, serem digeridos, gerando o hash referente ao arquivo a ser enviado. Então, é montado a mensagem. Esta mensagem é composta de seu primeiro parâmetro, o tipo de mensagem, seguido pelo nome do host, sua porta e então, o valor hash digerido pelo algoritmo de hashing e o nome do arquivo. Sendo estes valores encaminhados para o super nodo, só resta a este aguardar a confirmação de chegada.

5.2 Enviar arquivo para peer:

Diferente do processo anterior, onde é encaminhado apenas dados para indexação, neste caso, deve-se encaminhar o próprio arquivo para o cliente que aguarda o arquivo. Para

tal, é necessário que o cliente receba as informações acerca do host que faz a requisição, bem como seu endereço e requisição. Na requisição, encontraremos as informações referentes ao endereço do host e sua porta, a fim de estabelecer uma conexão. Então, busca-se o arquivo a partir do filename, a fim de pegar seu path. A partir destes, o arquivo é aberto e lido em bytes, a fim de que chunks de dados sejam enviados através do socket, até que ele chegue por completo no destino que o aguarda.

5.3 Receber arquivo de peer:

Tal método está relacionado diretamente ao envio do arquivo por um peer. Neste caso, a fim de ser realizada a solicitação, o cliente deve informar o nome do arquivo, bem com o endereço e porta do host de destino, em posse do recurso desejado. Estas informações são importantes para a montagem do protocolo de solicitação, bem como a abertura de um socket, por onde será encaminhado o arquivo.

5.4 Keep Alive

Um método simples, que a cada cinco segundos, envia uma mensagem tipo heartbeat para o super nodo cujo qual o cliente está relacionado. Esta tem como finalidade informar ao super nodo que ele ainda está disponível e deve manter os dados relacionados a ele em seu sistema.

5.5 Find

Faz uma requisição ao super nodo cujo qual o cliente esteja conectado a fim de que este faça uma busca pelo arquivo inserido como parâmetro. Este por sua vez, ao receber a requisição irá retornar, caso tenha encontrado, os dados do arquivo, como seu hash, host e porta para conexão e download dele.