Redes de Computadores

IF 975

Sistemas de Informação
Universidade Federal de Pernambuco
2019.1

Marcos Antonio Tavares Galvão
(matg@cin.ufpe.br)

Repositório do projeto:

www.github.com/Galtvam/projeto-de-redes

Camada low-level

Nesta seção será explicada as estruturas e bibliotecas de baixo nível da rede, implementadas para prover suporte as demais funcionalidades de nível superior, aqui também será visto o protocolo ***HLPCT (High Level Protocol for Command Transfer)***, protocolo criado e utilizado para definir as mensagens que transitam na rede.

Biblioteca TCP e UDP

Foram desenvolvidos dois módulos (Bibliotecas de auxílio), a fim de garantir de maneira mais facilitada as instruções de instanciamento, envio, recepção e fechamento de Sockets, para tal foram utilizados os métodos da biblioteca ***sockets*** de **Python**. Os dois módulos possuem funções que permitem com poucas linhas de código instância, receber e/ou enviar pacotes para um endereço, sendo assim as camadas mais superiores do jogo não se preocupam com detalhes de codificação de tais sockets.

Todos os sockets UDP seguiram o padrão de instanciação:

socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

Todos os sockets TCP seguiram o padrão de instanciação:

socket.socket(socket.AF_INET, socket.SOCK_STREAM)

HLPCT

(High Level Protocol for Command Transfer)

Eis o protocolo utilizado pela aplicação, o **HLPCT** foi pensado para ser minimalista, contendo apenas as principais informações necessárias para o diferenciamento e definição de instruções, em suma ele é composto por três partes:

1. CommandID:

   Compreende os primeiros 5 bites do pacote, nele é carregada a combinação de tais bits que identificam a qual commando ele corresponde.

2. Flag:

   Corresponde a 1 bit, que vem logo em seguida do CommandID, em determinados pacotes de um mesmo CommandID é útil haver um bit para possível variação do significado da instrução.

3. Carga Útil:

   Corresponde ao restante dos bits que vem no pacote e dependendo de qual CommandID foi utilizado tal campo corresponde a finalidades diferentes.

Exemplo de uma sequência de bits utilizando HLPCT:

00100 *1* 0100111000110…
[1] [2] [3]

abaixo segue a listagem dos CommandID's e suas descrições, seu funcionamento com demais módulos serão explicados posteriormente.

CommandID	Flag	Carga Útil	Descrição
00001	Não Usa	O ID (nickname) do usuário que está se apresentando	Pacote de apresentação de um novo peer na rede.
00010	Não Usa	**ID do usuário**, codificado em 80 bites, 8 bits para cada letra. **inRoom**, 1 bit que informa se o player está numa sala. **idRoom**, ID da sala que o jogador está.	Pacote de ping.
00011	Não Usa	Carrega 112 bits de dados de cada peer que existir na rede, os primeiros 32 bits para o IP de um peer e os 80 seguintes para o ID dele, e novamente se repete para os demais peers.	Pacote de resposta ao 00001, destina-se a passar informações dos demais peers da rede.
00100	Não Usa	Não Usa	Corresponde a uma solicitação para entrar na sala

			hosteada pelo jogador receptor.
00110	**0 >** O jogador foi aceito na partida e ficará aguardando o início do jogo. **1 >** O jogo foi iniciado pelo dono da sala.	**0 >** Não Usa **1 >** Carrega a informação do número máximo de players na sala.	**Flag 0 >** Pacote informando que a solicitação para entrar na sala (00100) foi aceita. **Flag 1 >**
00111	Não Usa	Não Usa	Indica que o seu pedido para entrar na sala (00100) foi rejeitado.
01000	Não Usa	ID de quem está recebendo o voto.	Pacote de votação para líder da rodada.
10000	Não Usa	Carrega 3 informações, um inteiro que indica o tamanho da palavra, sem seguida a palavra e por fim a divisão silábica da mesma.	Indica o início de uma rodada, é enviado pelo líder, ganhador da votação.
10001	Não Usa	Carrega a divisão silábica da palavra da rodada.	Pacote que informa aos demais players de uma sala qual foi a resposta de um jogador, todos, menos o líder, enviam ele ao fim de uma rodada.

Módulos de Auxílio

Para completude das funcionalidades de low-level foram implementados muitos auxiliares de funções, encoders de listas para binário, conversores de binário para lista, tradutores de nomes, IDs de salas e muito mais, nada de funcionamento independente, sendo no geral peças do quebra-cabeça. Ademais as bibliotecas mais utilizadas nos módulos anteriormente citados foi ***sockets, time e Struct.***

Estrutura P2P

Este pode ser dito como o coração da aplicação, nela veremos todas as nuances que compõem o descobrimento, manutenção e transações do jogo, bastante importante afirmar que tal módulo se preocupa com os clientes (terminais) que executaram o jogo e não na conexão restrita de uma sala. Tal módulo foi idealizado como uma estrutura que pudesse em tempo real prover dados sobre a composição da rede do jogo, por exemplo, quem está online, quem está jogando, quem deixou de estar online, quais pacotes recebi?, atendimento a novos peers, etc. Veremos com mais detalhes a seguir.

Descobrimento da Rede

Para tal finalidade analisaremos dois pontos de vista, um novo usuário que abriu seu jogo e um usuário com o jogo já aberto, começaremos pelo primeiro.

Inicialmente ao abrir o client (terminal rodando o jogo) a estrutura buscará identificar se existem peers já conectados na rede, para tal um pacote de CommandID 00001 será enviado, utilizando um socket UDP, para o IP de broadcast/multcast 224.0.0.1 em uma porta especifica, sendo esta numerada como 5554, logo após tal socket UDP será fechado e um novo, sendo este agora TCP, será aberto utilizando a porta 5555, tal socket ficará escutando durante 1 segundo uma tentativa de conexão, de um player que já esteja jogando, agora abriremos as possibilidades, caso não haja ninguém jogando, o socket TCP, obviamente, não receberá solicitações de conexão e, sendo assim, ele tentará descobrir qual o seu próprio IP utilizando o DNS do Google, segue o código:

```
def getIp():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.connect(("8.8.8.8", 80))
    ip = s.getsockname()[0]
    s.close()
    return ip
```

Caso o novo peer não possua conexão com a internet tal método falhará e manualmente o usuário informará seu IP, essa técnica é necessária pois muitas vezes os computadores não dispõem do arquivo ***hosts*** atualizados para que seja viável capturar o IP pelo socket com facilidade. Ao fim deste processo ele entrará na fase que citaremos adiante, o **hearing**.

A primeira situação foi explicada, não existem peers online, agora analisaremos o que acontece quando há players online. Todos os peers online rodam numa **Thread** um método chamado **hearing**, este é responsável por escutar qualquer pacote que chegue em UDP na porta 5554, assuma que quando o nosso novo usuário mandou o pacote 00001 em broadcast para porta 5554 um peer, que chamaremos de *Reginaldo,* capturou tal mensagem, então tratou de estabelecer uma conexão TCP com o remetente, que agora utiliza a porta 5555. Apesar das alusões aqui feitas a finalidade é Reginaldo fornecer a lista de IPs e IDs de todos que estão Online, neste pacote (00011) Reginaldo também informa o IP do remetente a fim de economizar o descobrimento via DNS do Google.

Em suma através do passo-a-passo citado acima, um novo peer descobre os players online e todos da rede salvam quem mandou a solicitação de dados e seu ID (vide tabela com o conteúdo da mensagem do pacote 00001), sendo assim a descoberta de novos usuários.

Hearing

Tal funcionalidade incluída na estrutura P2P é de tamanha importância que necessita de uma breve explicação. Já anteriormente citado o **hearing** é um método da estrutura P2P que é executado em uma thread durante toda a execução da aplicação, ele é o responsável por capturar todas as tentativas de comunicação com o player, sempre na porta 5554. Ao receber um pacote o **hearing** olha seu CommandID e/ou chama uma função ou repassa para o destinatário.

Checagem de Offlines

A estrutura P2P possui uma lista de todos os players online, cada player é representado por uma lista com 4 informações, [IP, nickname, inRoom, idRoom], sendo os dois primeiros autoexplicativos e os demais já foram explicados na tabela. A cada 0.5 segundos cada usuário envia um pacote de ping 00010 para os demais, vide tabela, ao receber um ping o receptor atualiza suas informações na lista de peers e anota num dicionário qual foi o time.time() daquele ping para aquela pessoa em específico, de tempos em tempos, também 0.5 segundos, há uma checagem do tempo do último ping de todos que compõem a listagem geral de peers, caso alguém não tenha pingado nos últimos 2.5 segundos ele é considerado Offline e é removida das listas de peers, portanto o jogo passa a desconsiderá-lo.

Game Dashbord

Tal estrutura é responsável pelas funcionalidades de controle de uma sala, além da sua integração e comunicação com a estrutura P2P a fim de garantir um funcionamento em tempo real da aplicação.

Objeto Sala

Este objeto tem como finalidade fazer a representação de uma sala, guardando em seu interior dados relevantes sobre a partida, veremos adiante, entretanto sem utilização e preocupação da integração com a estrutura P2P, tal parte se mantém na mão do módulo **Game Dashborad**. Cada sala possui um ID único, este sendo um inteiro, o mesmo é utilizado por exemplo para se entrar numa sala.

As principais informações que a sala armazena são flags, estados e listas, sendo:

- ID da Sala
- Número máximo de players
- Se a partida já começou
- Quais players estão vivos (não foram eliminados)
- Quais os players que integram a sala
- O número do Round
- O mestre da rodada
- O último mestre da rodada
- Um contador de votos (dicionário) para as eleições de líder.
- Várias e várias flags com finalidade de manter controle e permissoes sobre ações em momentos específicos do jogo.

Novamente, tal objeto é meramente armazenador, tendo como únicas funções registrar informações e estados da partida.

Controlador da Partida

Este é o segundo módulo mais importante da aplicação, perde somente para estrutura P2P, e consiste em todas as funcionalidades relacionadas a manipulação e execução de ações em uma sala, tal controlador por exemplo é responsável por receber os pacotes 00100, 00110, 01000, 10000 e 10001 do P2P e executar as ações associados a eles, vide tabela.

Inicialmente exploraremos a criação de uma sala, para tal o jogador deverá fornecer duas informações, ID para a sala, caso já esteja em uso o usuário deverá inserir outro, e o número máximo de players, após isso a sala será criada e através do ping tal informação será difundida. Através do ping o **Game Dashborad** consegue manter uma lista atualizada de salas existentes e qual player pertence a ela. Após uma sala ser criada o dono mantém-se de maneira passiva aguardando jogadores, que quando enviarem, ao dono, o pacote 00100 (solicitação para entrar na sala) o mesmo verificará a possibilidade de aceitar ou não este novo jogador, baseado em quantas vagas existem na sala e caso esteja cheia serão rejeitados, e então enviará, em resposta, o pacote 00110 flag 0 ou 00111. Quando o líder desejar ou a sala encher o jogo será iniciado, logo o pacote 00110 flag 1 será enviado aos integrantes da lista dos players que integram a sala, que só está completa até então para o líder da sala, e os mesmos saberão do início do game.

Um jogador após receber o pacote 00100 flag 0 ele altera sua informação inRoom para True, porém não informa qual o ID da sala que ele integra, portanto ele ainda não sabe quem são os jogadores que também compõem a sala, ele só conhece o líder. Ao receber o pacote 00100 flag 1 todos os peers que foram aceitos mudam seu idRoom para o da sala e buscam na lista de peers da estrutura P2P, que já está atualizada por conta do ping, quem também integra a sala, de tal forma são necessárias menos informações nos pacotes. Passados 2 segundos de warmup, tal preparação, a partida realmente se inicia seguindo as definições do projeto.

Para o jogador entrar numa sala é muito mais simples do que criá-la, tal player simplesmente fornecerá o ID correspondente a sala e enviará um pacote 00100 para o dono da mesma, como citado anteriormente ele receberá ou o pacote 00110 flag 0 ou 00111, no caso do primeiro ele ficará somente aguardando o contato do líder com o próximo pacote, o 00110 flag 1, para então executar as ações citadas no parágrafo anterior.

Analisaremos mais 2 pontos, as votações de líder e as respostas enviadas. Na votação de líder cada player descobre, através do **Game Dashboard** quais são os candidatos válidos para se votar, e após selecionar seu candidato um pacote 01000 contendo na carga útil o ID de para quem seu voto será mandado para todos que integram a lista de participantes da sala, todos os jogadores da sala receberão tal pacote e contabilizaram internamente quem, ao fim dos 10 segundos, é o vencedor e consequentemente o líder da rodada. Após saber quem é o líder da rodada os demais jogadores ficarão aguardando um pacote 10000 chegar, o mesmo contém a palavra da rodada e sua divisão silábica correta, definida pelo líder, após seu recebimento um contador de 15 segundos se inicia e o jogador pode dizer sua resposta, tal divisão, de cada jogador exceto o líder, será enviada via pacote 10001 para todos os outros que integram a sala e

novamente, ao fim dos 15 segundos, a contabilização dos eliminados será feita internamente em cada cliente.

Instruções para Jogar

Obs: Existe a possibilidade da rede bloquear tal broadcast de descobrimento 224.0.0.1, as do Cin não bloqueiam.

Para jogar é só executar o arquivo ***play.py*** contido na pasta do jogo.