Apresentação EP2 Redes (2021)

Lourenço Sborz (11208005) & Mohamad Rkein (10740130)

20 de junho de 2021

1. Protocolo
   - Início da Conexão
   - Comandos
   - Jogo
   - Tratamento de Erros
2. Extras
   - IA
3. Experimentos e Gráficos
   - Sem clientes
   - Com 2 clientes sem jogar
   - Com 2 clientes jogando
   - Comparação do uso de rede nos testes

Depois que o cliente conecta com o servidor, a primeira coisa que o servidor faz é usar o socket dessa conexão para enviar duas novas portas para o cliente. Uma dessas portas será usada para abrir a conexão criptografada e a outra será usada para abrir a conexão que chamamos de *background server listener*. A conexão criptografada e a conexão normal são usadas para enviar para o servidor os comandos manuais que o usuário escreveu na prompt. O *background server listener* é usado por uma thread separada e fica o tempo todo recebendo e respondendo pings, além de ser responsável por verificar se o usuário foi "desafiado".

Os comandos digitados pelos usuários são parseados da seguinte maneira: primeiro, quebramos o comando nos espaços, para separar os argumentos do nome e depois checamos por qual socket o comando deve ser enviado. Por exemplo, se o usuário digtou `login a b`, sabemos que isso deve ser enviado pelo socket criptografado, então enviamos o comando e seus argumentos pelo socket com SSL. Se o comando não existe, enviamos ele pelo socket criptografado. Fizemos isso por dois motivos:

1. Achamos melhor tratar os erros no servidor, pois um usuário malicioso poderia simplesmente modificar o seu cliente para deixar as falhas passarem caso elas fosse tratadas dentro do código do cliente.
2. Já que vamos enviar os comandos errados para o servidor pelo motivo anterior, é melhor enviarmos eles pelo criptografado, pois pode ser que o cliente tenha escrito *logn username passw*.

Todos os comandos enviados pelo cliente têm algum tipo de resposta do servidor, nem que seja um simples “ok”, indicando que tudo ocorreu corretamente.

Alguns comandos só estão habilitados para o usuário usar em condições específicas, por exemplo: o usuário só pode usar o comando begin se ele está logado.

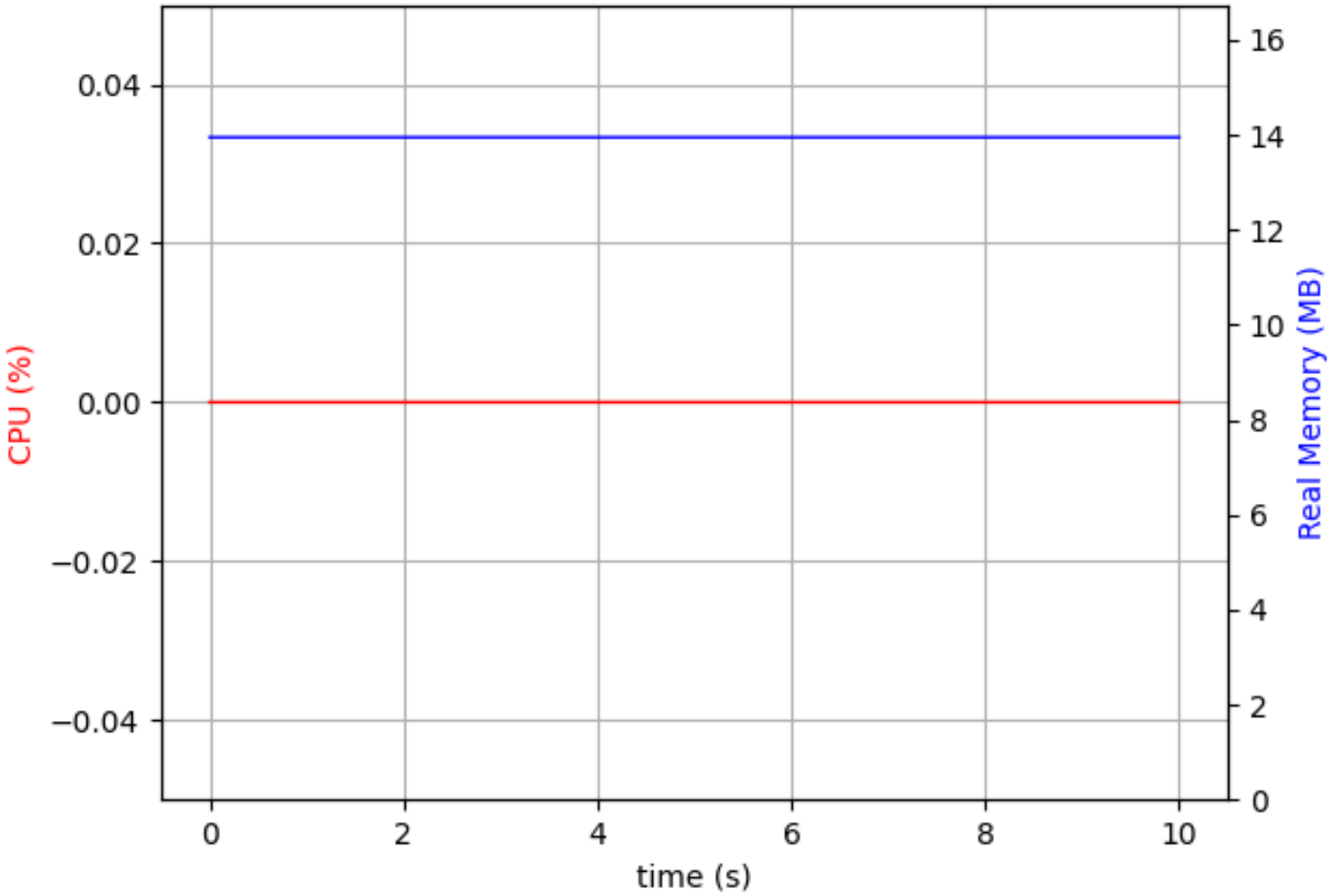
O jogo é feita de maneira *peer to peer* como exigido no enunciado. Suponhamos dois usuários: a e b. O que acontece quando a desafia b usando o comando `begin b` é: o begin é enviado para a thread do servidor responsável por a. Essa thread se comunica com a thread o servidor responsável por b usando o algoritmo dos produtores e consumidores e então a thread responsável por b envia esse comando para o cliente do b. b pode então digitar `refuse` ou `accept` (nesse caso é enviado junto uma porta e o IP de b) para responder. A resposta de b faz o caminho inverso que o begin fez, e chega até o a, que então age da maneira adequada. No caso de `refuse`, o a simplesmente recebe a mensagem de que seu

Como pedido no enunciado, implementamos um sistema de heartbeat. O servidor envia um pacote de "Ping" para o cliente a cada 3 segundos. O cliente irá assumir que sua conexão foi interrompida caso ele fique 3 minutos sem receber o ping e irá fechar os sockets e imprimir uma mensagem de erro. O servidor faz a mesma coisa, caso não receba resposta do ping em 3 minutos, ele irá assumir que o cliente caiu e irá fechar a conexão com o cliente.

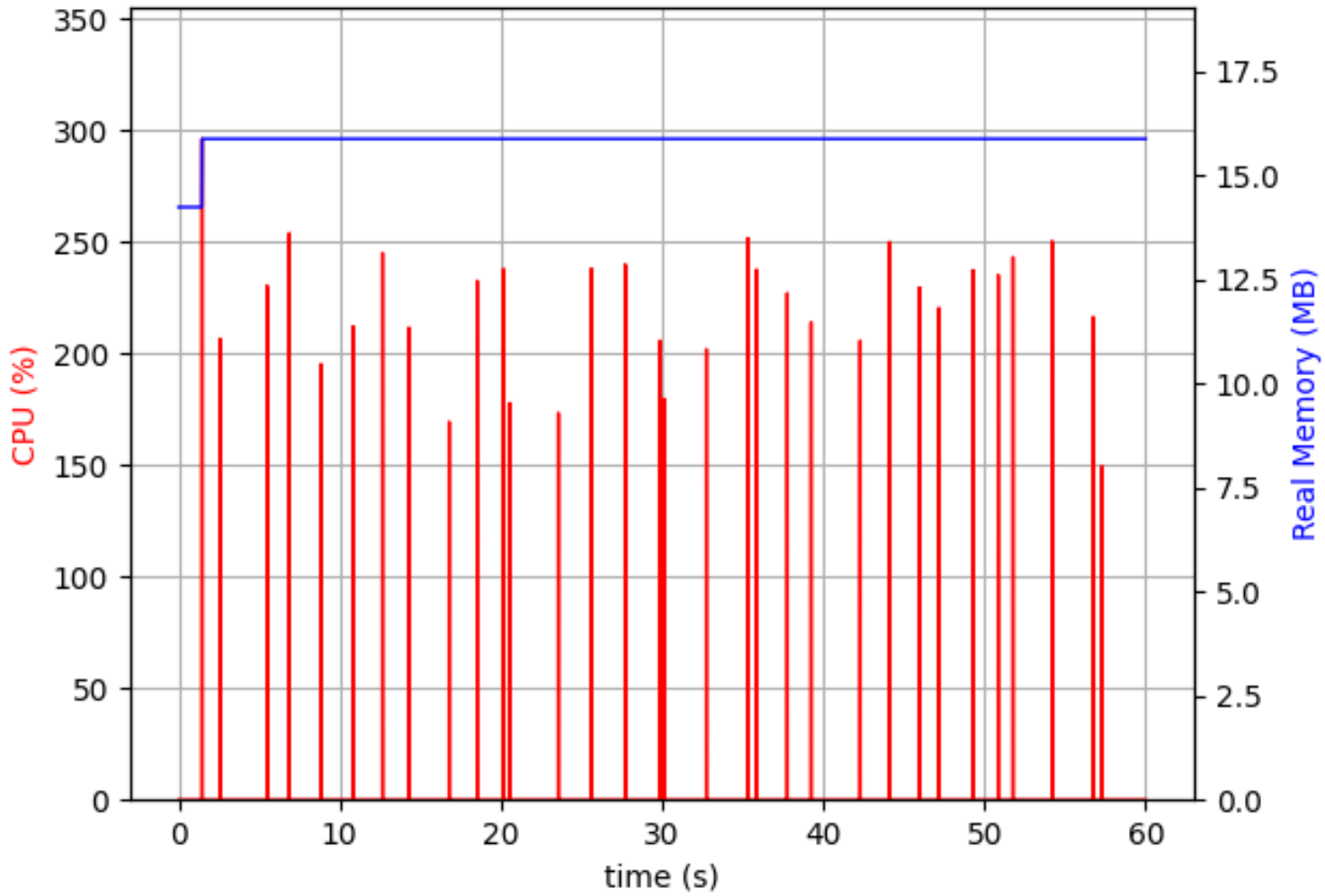
Quando o servidor é finalizado com SIGKILL, os clientes ficam 3 minutos tentando reconectar. Caso o servidor seja reiniciado na mesma porta dentro desse intervalo, os clientes são reconectados automaticamente e os que estava logados, são logados automaticamente também. Caso os 3 minutos passem sem o servidor ser reiniciado, os clientes avisam o usuário e terminam. Consideramos que o servidor servidor foi finalizado com uma falha se ele morreu enquanto ainda tinha algum usuário logado.

Como bônus, implementamos um cliente (`cliente_ia.py`) que é uma inteligência artificial que (teoricamente) nunca perde. Para jogar contra ela basta conectá-la no servidor e desafiar o usuário chamado "velha".

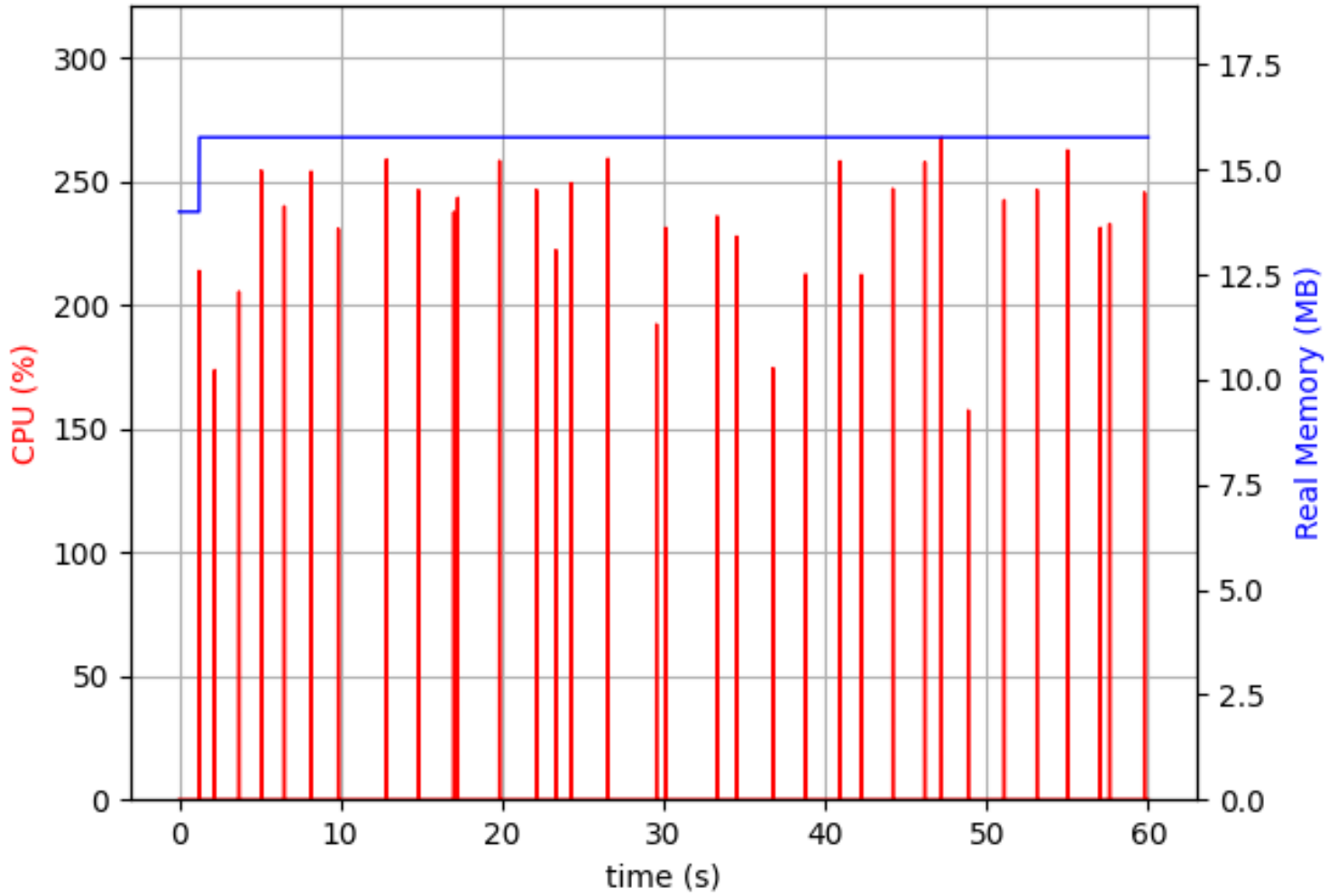
Uso de cpu e memória para o servidor sem clientes:

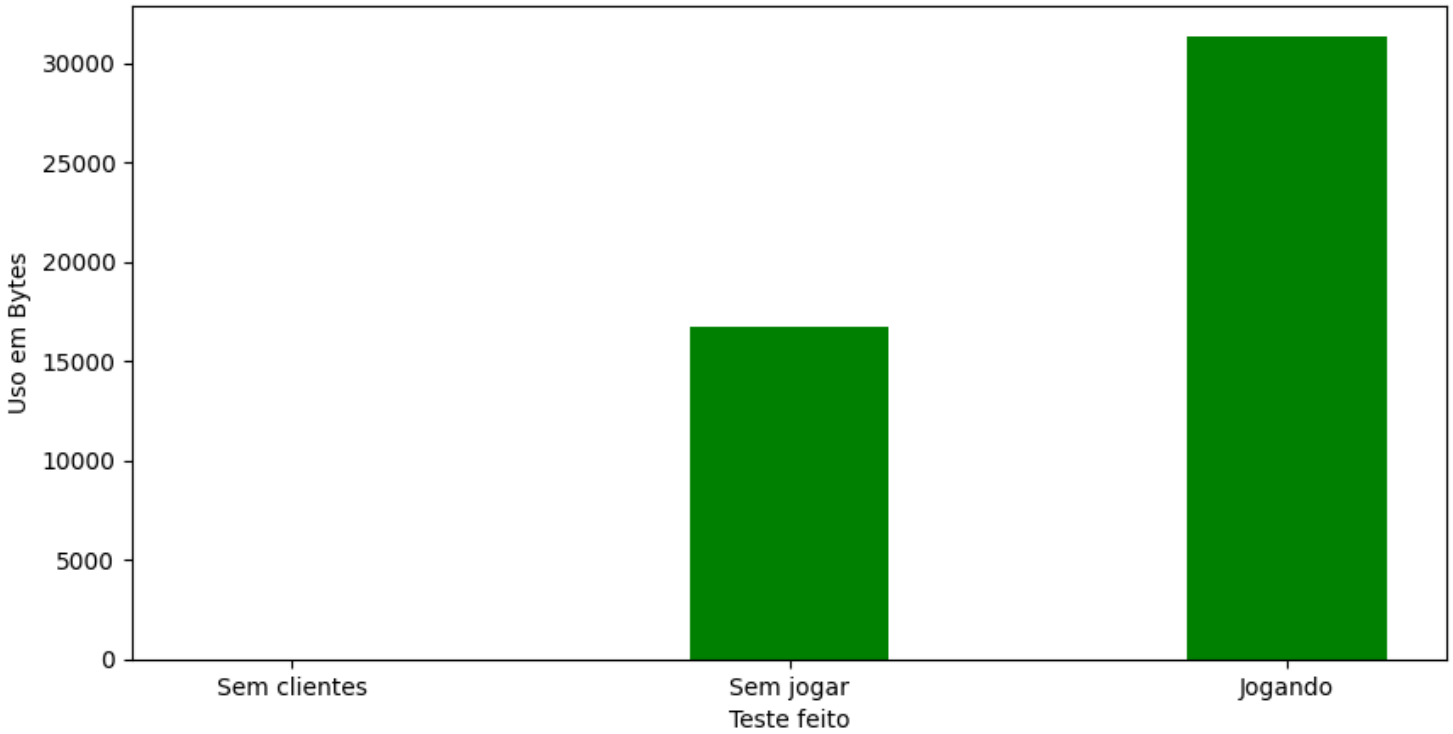


Uso de cpu e memória para o servidor com 2 clientes sem jogar:



Uso de cpu e memória para o servidor com 2 clientes jogando:




30000
25000
20000
15000
10000
5000
0
Uso em Bytes
Sem clientes
Sem jogar
Jogando
Teste feito