Relatório do AP2

Universidade de Aveiro

Bruno Gomes, César Malhão

VERSAO

Relatório do AP2

Departamento de Telecomunicações e Informática
Universidade de Aveiro

Bruno Gomes, César Malhão
(103320) brunofgomes@ua.pt, (104279) cesarmalhao@ua.pt

28/05/2021

Resumo

Este trabalho teve como principal objetivo criar um servidor que suporte a geração de um número inteiro aleatório (entre 0 e 100), que vamos designar por número secreto, bem como o número máximo de tentativas (entre 10 e 30) concedidas para o adivinhar. E um cliente que permita adivinhar esse número secreto. Ou seja um jogo de adivinha o número secreto.

O servidor nunca deverá aceitar dois clientes com a mesma identificação a jogar simultaneamente e deverá criar e atualizar um ficheiro designado por report.csv onde vai escrevendo os resultados dos diversos clientes quando estes terminam o jogo. O cliente pode desistir em qualquer altura e o jogo acaba quando ele adivinha o número secreto ou quando esgota o número máximo de tentativas que dispunha para jogar. Caso o cliente exceda o número de jogadas de que dispunha o jogo será considerado sem sucesso mesmo que ele tenha adivinhado o número. Quando o jogo acaba corretamente o cliente deve escrever no monitor uma mensagem a indicar se adivinhou ou não o número secreto e quantas jogadas efetuou. Por sua vez o servidor acrescenta ao ficheiro a informação relativa ao jogo: cliente; número secreto; número máximo de jogadas; número de jogadas efetuadas; e o resultado obtido pelo cliente (desistência ou sucesso ou insucesso).

A comunicação entre o servidor e o cliente deverá ser feita através de funções já fornecidas, que permitem enviar e receber dicionários. E para aumentar a segurança no envio de dados, para que outros clientes não consigam descobrir o número secreto, foram criadas funções para encriptar e desencriptar informação.

Agradecimentos

Queremos agradecer aos nossos professores da cadeira de LABI, António Manuel Adrego da Rocha e João Paulo Barraca por nos ensinarem como utilizar as ferramentas deste software. Queremos também agradecer aos nossos colegas do 1ºano do curso, por nos ajudarmos mutuamente e também aos nossos colegas de anos superiores, por nos passarem a sua experiência que acumularam com os seus próprios projetos.

Índice

Capítulo 1

Modelo Cliente/Servidor

Nas comunicações entre aplicações é usual distinguir-se dois tipos de intervenientes: o cliente e o servidor. O cliente é aquele que inicia a comunicação e faz um pedido, enquanto o servidor é aquele que aceita um pedido e realiza uma ação, podendo emitir uma resposta. A Figura 1 representa uma simples interação entre vários clientes e um servidor.

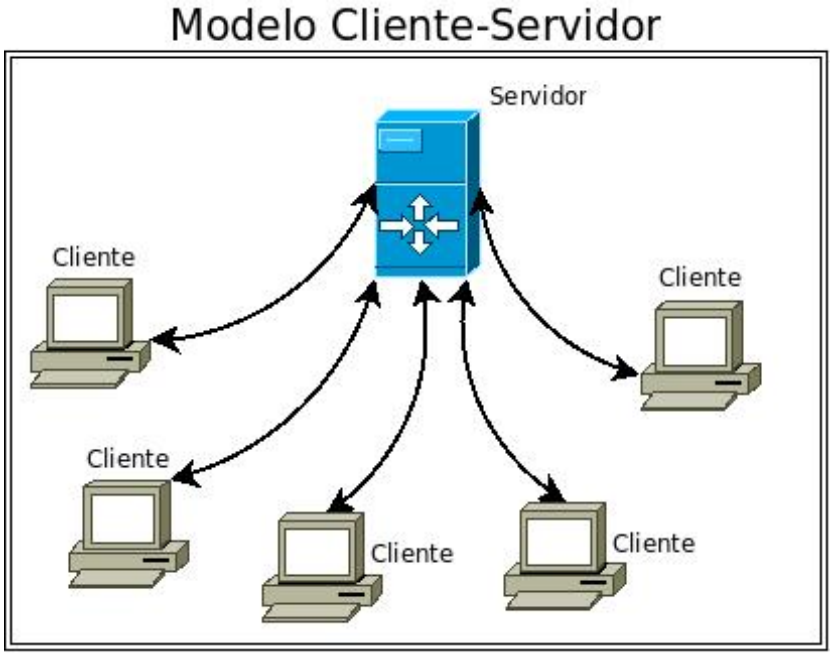

Figura 1.1: Interação de um servidor com vários com clientes

Esta separação de funções é importante porque implica que as aplicações do cliente e do servidor são distintas: o fluxo lógico (algoritmo) do cliente é diferente do fluxo lógico do servidor. Outra característica deste modelo é que vários clientes se podem ligar a um único servidor.

Como o servidor pode-se conectar com vários clientes, para que as comunicações entre clientes e servidor sejam seguras, isto é, para que os clientes não consigam intercetar a informação transmita entre outro cliente, recorremos á utilização da criptografia.

A criptografia é um conceito antigo de transformação de conteúdos, que até há cerca de duas décadas era sobretudo usado em ambientes onde a segurança é um elemento fundamental (ambientes militares, serviços de informações, etc.). Hoje em dia, em virtude da massificação do uso da informática e da Internet para os mais variados fins, a segurança criptográfica faz parte do dia-a-dia do cidadão comum, mesmo que disso ele não se aperceba.

No decorrer do trabalho foi utilizado a manipulação de documentos, em dois formatos muito comuns, usados para a troca de informação entre aplicações e dispositivos, o formato Comma Separated Values (CSV) e o formato JavaScript Object Notation (JSON). Os dois utilizam representações textuais para a codificação da informação, o que tem a vantagem de permitir que os dados sejam interpretados ou até gerados por humanos.

A motivação deste trabalho foi colocar em prática os conteúdos lecionados nas aulas relativamente á comunicação entre aplicações, criptografia e manipulação de documentos em python.

Este documento está dividido em quatro capítulos. Depois desta introdução, no Capítulo 2 é apresentada a metodologia seguida, no Capítulo 3 são apresentados os resultados obtidos, sendo estes discutidos no Capítulo 4. Finalmente, no Capítulo 5 são apresentadas as conclusões do trabalho.

Capítulo 2

Metodologia

A comunicação entre os clientes e o servidor foi suportada por sockets TCP, porque estes têm a vantagem de permitir detetar a falha de um interlocutor aquando do uso do socket com a ligação para com o mesmo.

O funcionamento de programa foi estruturado em quatro operações: START, QUIT, GUESS e STOP. A cada operação que o cliente inicializa, este envia um dicionário para o servidor com a operação e a informação adequada. Assim sendo os dicionários enviados do cliente para o servidor tem a seguinte forma:

- {op = "START", client_id, [cipher]}
- {op = "QUIT"}
- {op = "GUESS", number }
- {op = "STOP", number, attempts}

Por sua vez, o servidor ao receber um destes dicionários entra na função new_msg, que deteta a operação que o cliente está a pedir e pede a uma função especializada para obter a informação a enviar de volta para o cliente. Sendo estas funções as seguintes: new_client, quit_client, guess_client e stop_client. E cria e atualiza um ficheiro designado por report.csv onde vai escrevendo os resultados dos diversos clientes quando estes terminam o jogo.

Na função new_client é detetado um novo cliente através cliente_id que foi enviado pelo dicionário da operação START. Este é adicionado a um novo dicionário do servidor chamado gamers, onde ficaram guardadas as informações relativas ao jogo do cliente. Caso o cliente já esteja nesse dicionário a função vai retornar um novo dicionário com o erro de o cliente já existir. De seguida, são atribuídos valores ao número secreto e ao número máximo de tentativas

através da função random. Por fim é retornado o dicionário { "op": "START", "status": True, "max_attempts": nº máximo de jogadas }.

Na função quit_client é atualizado o ficheiro report.csv com o resultado QUIT e é eliminado o cliente e sua informação do dicionário gamers, retornando o dicionário { "op": "QUIT", "status": True }

Na função guess_client é determinado se o número passado pelo cliente é maior, menor ou igual ao número secreto e é retornado o dicionário { "op": "GUESS", "status": True, "result": "smaller"/"larger"/"equals"}.

Na função stop_client é atualizado o report.csv com o resultado SUCCESS caso o cliente tenha conseguido acertar o número secreto antes de ultrapassar o número máximo de tentativas, senão o resultado será Failure. Retornando o dicionário { "op": "STOP", "status": True, "guess": número secreto }.

As operações recv_dict e send_dict, já fornecidas no ficheiro common_comm.py são utilizadas pelo servidor para, respetivamente, receber o pedido da operação (START, GUESS, STOP e QUIT) que o cliente pretende executar e para enviar-lhe a respetiva resposta com o resultado de execução da mesma. Por sua vez a operação sendrecv_dict é utilizada pelo cliente para enviar um pedido de execução de uma operação aguardando pela respetiva resposta do servidor.

Na questão de segurança, os clientes comunicam secretamente com o servidor fazendo encriptamento dos dados numéricos. O cliente contacta o servidor para dar início ao jogo enviando um dicionário com o seguinte formato: { "op": "START", "client_id": nome do cliente, "cipher": cipherkey }.

Capítulo 3

Resultados

Para puder avaliar a execução do programa, isto é, testar as funcionalidades do servidor e do cliente, criamos uma nova pasta com os ficheiros base: server.py, client.py e common_comm.py. De seguida, alteramos o ficheiro client.py para enviar dicionários com informação para testar cada operação.

Para testar a operação **START**:

{"op":"START","client_id":client_id}

Para testar a operação **QUIT**:

{"op":"QUIT"}

Para testar a operação **GUESS**:

{"op":"GUESS","number":25}

{"op":"GUESS","number":50}

{"op":"GUESS","number":75}

Para testar a operação **STOP**:

{"op":"STOP","number":50,"attempts":3}

Após cada dicionário enviado com a função sendrecv_dict fizemos print do dicionário enviado, que por sua vez, agora continha a informação devolvida pelo servidor.

Capítulo 4

Análise

Depois de fazer os testes através dos métodos demonstrados no capítulo anterior, obtivemos os dicionários enviados pelo servidor. Sendo eles:

{'op': 'START', 'status': True, 'max_attempts': 16}

{'1': 'op': 'GUESS', 'status': True, 'result': 'larger'},

'2': {'op': 'GUESS', 'status': True, 'result': 'larger'},

'3': {'op': 'GUESS', 'status': True, 'result': 'equals'}

{'op': 'STOP', 'status': True, 'guess': 75}

{'op': 'QUIT', 'status': True}

Isto demonstra que as funções do servidor estão a funcionar devidamente, ou seja, recebem a informação dos clientes, processam-na e dependo da informação passada dão uma resposta adequada. Quanto á parte da segurança não conseguimos testar pois estávamos a obter erros ao executar o cliente.

Capítulo 5

Conclusões

Ao realizar este trabalho e através da prática dos conhecimentos lecionados nas aulas, nós ficamos com uma melhor perceção de como são feitas as comunicações num modelo de servidor-cliente e apesar de não termos conseguido alcançar o objetivo na questão da segurança do servidor, sentimos que estamos lá perto e sem dúvida com ajuda dos professores, no futuro, seremos capazes de compreender melhor esse assunto.

Contribuições dos autores

Neste trabalho o aluno César Malhão demonstrou maior destreza a utilizar as ferramentas de python para realizar a conexão do servidor com o cliente, enquanto o aluno Bruno Gomes revelou um pouco de dificuldades (pois este limitou-se a trabalhar no client.py e um pouco no server.py). O relatório final foi feito por ambos os alunos. Portanto a nota final deve ser de 60% para o César e 40% para o Bruno.

Bibliografia

[1] https://profsalu.com/2014/05/18/arquitetura-clienteservidor/

[2] https://realpython.com/python-sockets/

[3] https://docs.python.org/3/library/socket.html

[4] https://www.geeksforgeeks.org/socket-programming-python/

[5] https://www.tutorialspoint.com/python/python_networking.htm

[6] https://www.w3schools.com/python/python_json.asp

[7] https://docs.python.org/3/library/csv.html

[8] https://www.w3resource.com/JSON/structures.php