Universidade Federal de Minas Gerais

DCC023: Redes de Computadores

TP1: Bridge Defense

Nome: Arthur Nunes Cascardo

Matrícula: 2018013968

SAG: 2018013968:1:1c9324306dd2f0c2e3bc8623dc9df0d9d9b90cc5e0557843073665d069ceed3d+

5a9ba60f795949fac526fe359039e17ce52f6e58f0a4ba3f1a748c98e2ab04fc

Linguagem: Python

**Implementação**

[Codificação das mensagens em JSON](#_Codificação_das_mensagens)

[Autenticação com os quatro servidores](#_Autenticação_com_os)

[Algoritimo de retransmissão](#_Algoritmo_de_retransmissão)

**Execução**

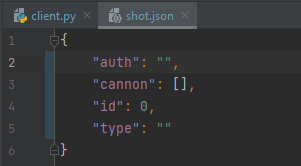
[Algoritmo de execução](#_Algoritimo_de_execução)

Implementação

# Codificação das mensagens em JSON

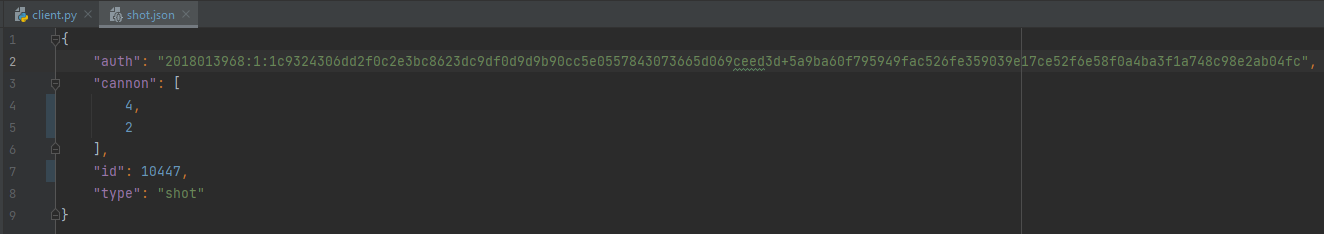
O programa utiliza o tipo de dados “lista” default do python para receber argumentos de cada tipo de comando, e um arquivo JSON com as chaves e o tipo de dado correspondente a cada item de cada chave ja definido. Por exemplo o JSON de request para tiro de um canhão é inicialmente:

Arquivo padrão:



Assim que os parametros são passados, estes são escritos no arquivo e enviados ao servidor:

Arquivo com argumentos:



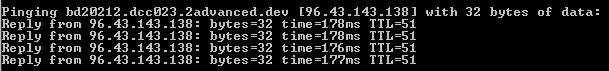
Para tal foram escritas as funções formatRequets(args) e writeSendData(json\_obj). O argumento “args” é do tipo lista e contém as chaves de cada comando em ordem alfabética. O arquivo, cujo nome é o tipo do request, é aberto, então um dicionário “json\_obj” recebe os valores das respectivas chaves. Esse dicionário é enviado em formato string para “writeSendFile” a qual sobrescreve o arquivo padrão com o que contém argumentos.

# Autenticação com os quatro servidores

A autenticação com os servidores é feita através de um único socket na função “authReq”. A fim de otimizar o tempo de autenticação é enviada uma requisição, e caso a conexão não seja bem sucedida o programa tenta conectar-se com o próximo servidor, aplicando pequenas mudanças feitas pelo algoritimo de retransmissão. Uma lista contém os portos não conectados e à medida que cada porto é autenticado esses vão sendo retirados da lista. O programa itera esse processo até que a lista esteja vazia.

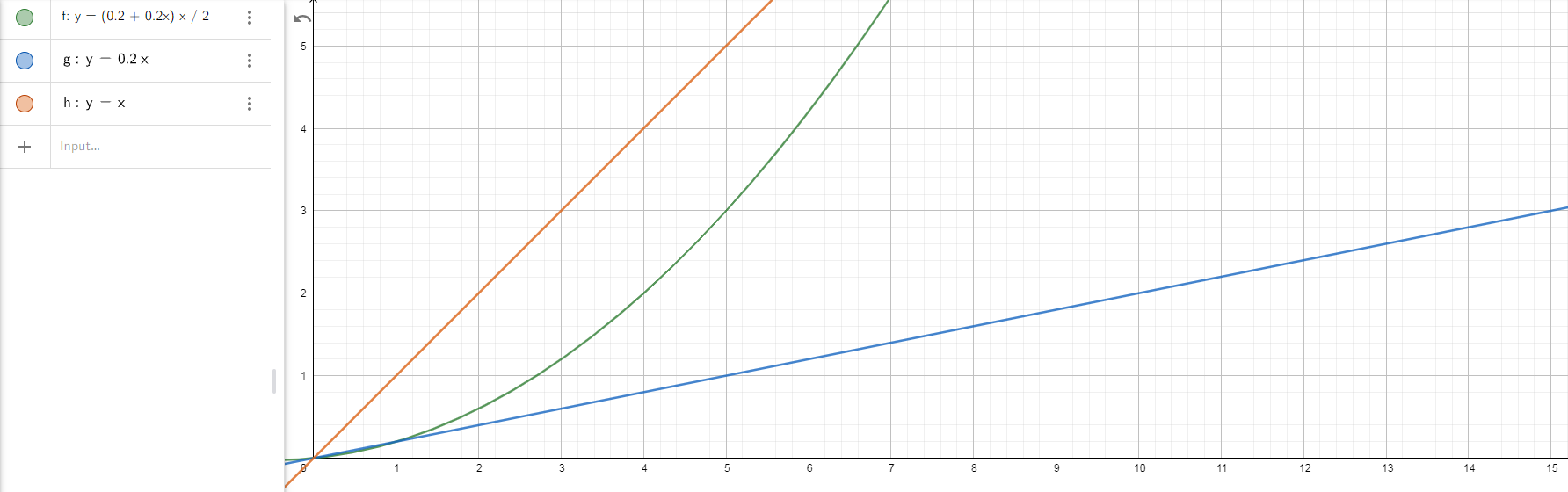
# Algoritmo de retransmissão

Para o algoritimo de retransmissão foi utilizado um timeout inicial na mesma ordem da latência da resposta do servidor ao comando ping do prompt de comando do windows.



F:\Disciplinas\Redes\TP0\Documentation\timeout.PNG

Na ocorrencia de um timeout, o pacote é reenviado e o tempo de timeout é acrescido de 200ms, que é no caso o TO inicial. Dessa forma temos uma progrssão aritimética do tempo de timeout. Assim caso o servidor responda na primeira tentativa o tempo minimo de conexão é 200ms com 1 tentativa, do contrário temos a função T = ((0.2 + 0.2\*n) \* n)/2 sendo n o número de tentativas



No gráfico acima temos f representando o algoritimo, g representando um tempo fixo de 200ms e h representando um tempo fixo de 1s. Observa-se que fixando um tempo de resposta de 1 segundo, e em seguida fixaramos o numero de tentativas em 5 temos:

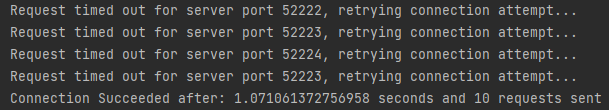
F : Envia 3 requisições e leva 3s para enviar 5 requisições

G : Envia 5 requisições e leva 1s para enviar 5 requisições

H : Envia 1 requisição e leva 5s para enviar 5 requisições

Observa-se que F representa a mediana em ambos os casos apresentados sendo o algoritmo ótimo para situações nas quais o servidor responde rápido ou demora a responder.

Uma análise mais profunda mostra que dada uma função f(x) = Ax sendo A o tempo de timeout e x o número de tentativas. Tem-se que f(x) = Ax é mais rápida que T(n) = ((0.2 + 0.2\*n) \* n)/2 a partir de x = (10A – 1), portanto para X na ordem de segundos, T(n) é consideravelmente mais rápida. Outro ponto importante é número de retransmissões, para coeficientes A abaixo de 0.4 o número de retransmissões é igual em x = (10A – 1) e a partir desse ponto observa-se que a quantidade de retransmissões cresce no mínimo em uma P.A de razão 1, razão essa que aumenta de forma inversamente proporcional a ‘A’

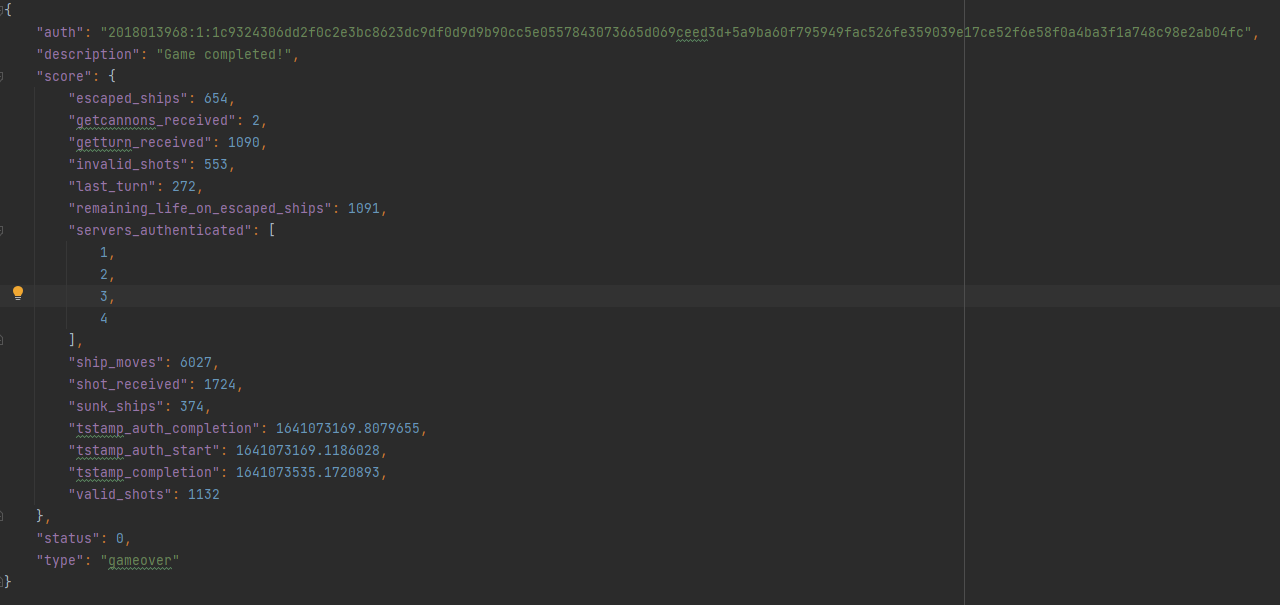


Os tempos e tentativas para autenticação dos 4 servidores varia respectivamente entre 0.6ms a 1s e 7 a 10 tentativas.

Execução do programa

# Algoritimo de execução

A fim de atingir os objetivos propostos do TP foi feito um algoritmo que automatiza a execução do jogo. O algoritimo autentica os 4 servidores e em caso de sucesso faz a requisição de posicionamento dos canhões. A posição dos canhões é salva em uma lista e é enviada a requisição de turno. Ao receber o estado das 8 pontes em cada um dos 4 rios, o algoritimo percorre a lista “ships” de cada ponte de cada rio e salva em outra lista respectivamente o id e posição do barco na tupla (ponte, rio). Em seguida, uma função computa as possíveis coordenadas de tiro de cada canhão e compara com a posição de cada barco. Caso haja um barco dentro da distância de tiro de um canhão, que ainda possua “vidas”, uma requisição de tiro é efetuada. O processo é repetido até que todas as possibilidades de tiro tenham sido consideradas. Posteriormente, é requisitado um novo turno e o algoritmo volta ao inicio.



A taxa de sucesso de tiros varia geralmente entre 95% e 100% e o número de navios afundados fica entre 450 e 550 na maioria dos jogos.