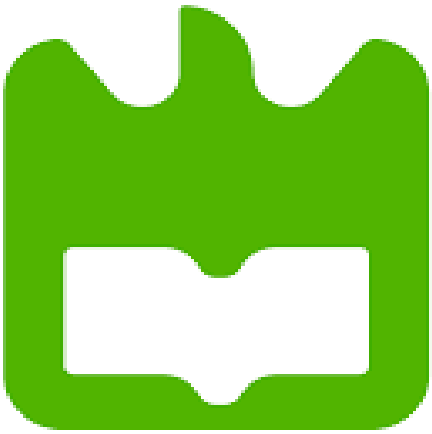
Universidade de Aveiro



universidade de aveiro

Departamento Eletrónica, Telecomunicações e Informática

Relatório Distributed Object Detection

Engenharia Informática

Trabalho Realizado por:

- Diogo Cunha nºmec: 95278

Introdução

Como segundo projeto da disciplina de Computação Distribuída foi criado um programa em que o cliente deve enviar um vídeo em formato MP4 para um servidor central através de protocolo HTTP, sem aguardar qualquer resposta do processamento final do servidor. Do servidor é esperado que analise o vídeo e emita uma mensagem para a consola (do servidor) sempre que o número de pessoas num frame do video ultrapassou um dado limite fornecido por linha de comandos. Para esta tarefa é fornecido este guião e um código seminal disponível

1.1 Servidor

O servidor quando é iniciado fica à espera de estabelecer uma ligação com um worker no /workers ou então de receber um vídeo na sua raiz (/).

Quando recebe um post de um worker ele regista-o num dicionário chamado workers_list em que a chave do dicionário e a porta do worker e o valor e o seu estado (Disponível ou Ocupado).

Quando recebe um post de um cliente com um vídeo particiona esse vídeo em frames guardando cada frame num dicionário (frames). Esse dicionário tem como chave um número que representa a contagem dos frames, de valores tem o frame em si e o seu estado (Working ou Waiting).

Depois de ter os workers registados e o vídeo particionado em frames o servidor prossegue a distribuição dos frames pelos workers registados que estejam disponíveis.

Ao fim de receber toda a informação sobre todos os frames vindo dos workers, o servidor começa a tratar a informação imprimindo no terminal os frames processados, tempo médio de processamento de um frame, pessoas detetadas, total de classes detetadas e o top 3 de objetos detetados.

Imprime também os frames em qual foi ultrapassado o limite máximo de pessoas.

1.2 Worker

O worker quando e iniciado executa a parte de ler as classes e executa o Setup tensorflow, keras and YOLOv3 ect e faz um post para o servidor (/workers) a registar-se na lista de workers do servidor como estando “Disponível”.

Apos se registar fica à espera de receber um frame para tratar. Ao fim de o receber e retirar toda a informação desse frame o worker envia para o servidor a informação acerca do mesmo e depois atualiza o seu estado para "Disponível”.

De notar que para executar o worker tem de ser especificado a sua porta sendo a porta 5001 a default (Ex: $ python3 worker.py --worker-port 5002).

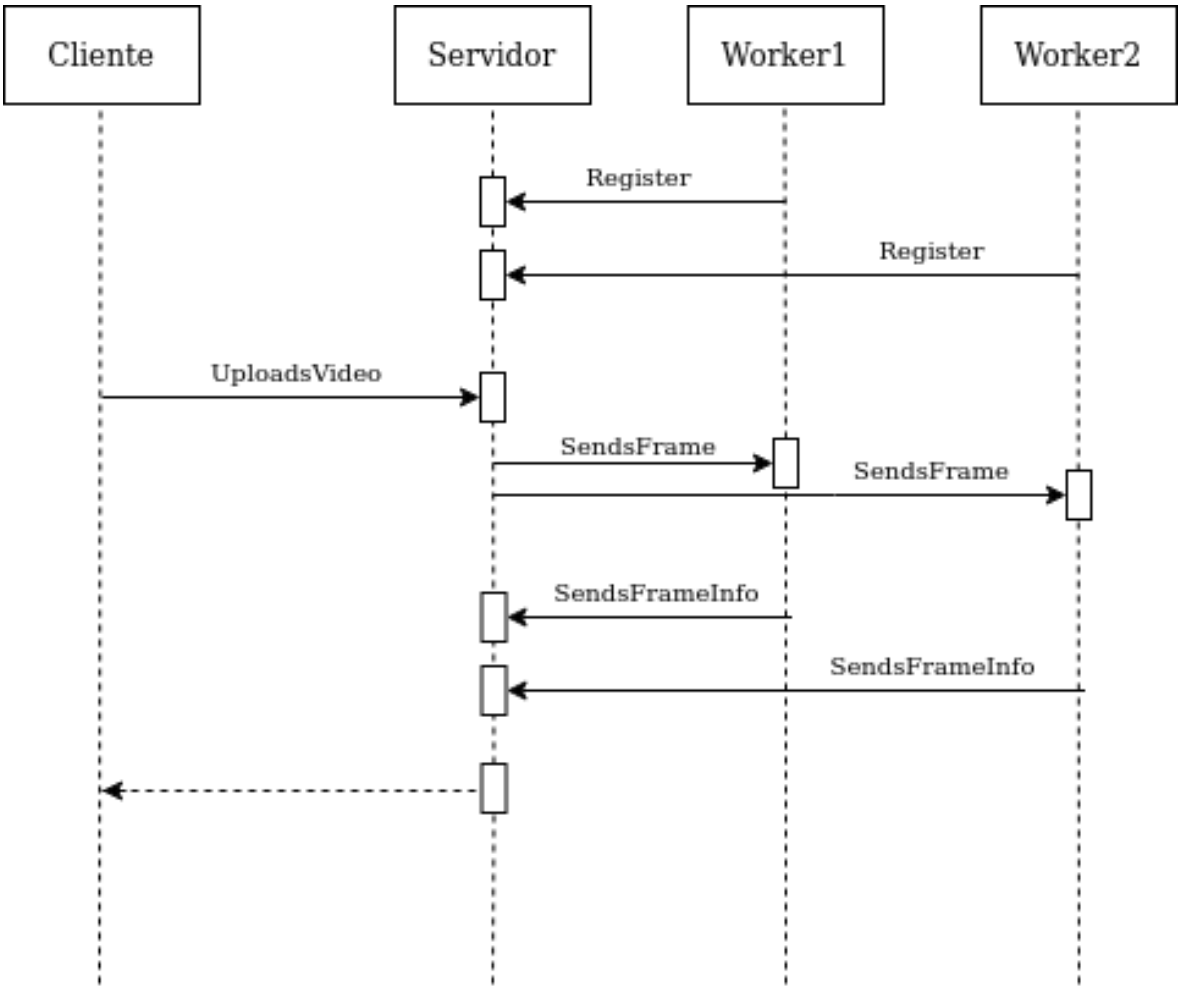
1.3 Worker e Servidor

O worker e o servidor comunicam entre si através de protocolos HTTP e fazendo uso do framework flask. Nos posts entre o worker e o servidor o conteúdo enviado é passado para formato json antes de ser enviado e depois o inverso quando recebido.

Server faz post de “files” que contem o frame que esta guardado no dicionario “frames”.

O worker quando é inicializado e quando acaba de tratar um frame, faz post de um dicionário com o seu estado e a sua porta em formato JSON, ao fim de tratar o frame para alem de enviar o seu estado envia tambem em formato JSON um dicionário com a informação do frame tratado.

Sequence Chart



Resultados

Resultados de 1 worker

```
Processed frames:  431
Average processing time per frame:  1.9174452652234213
Person objects detected:  8557
Total classes detected:  6
Top 3 objects detected:  ['person', 'boat', 'truck']
127.0.0.1 - - [25/Jun/2020 21:54:38] "POST /workers HTTP/1.1" 200 -
127.0.0.1 - - [25/Jun/2020 21:54:38] "POST / HTTP/1.1" 0 -
^C
real    14m51,169s
user    14m28,788s
sys     0m2,940s
```

Resultados de 2 workers

```
Processed frames:  431
Average processing time per frame:  3.6012551419419645
Person objects detected:  8557
Total classes detected:  6
Top 3 objects detected:  ['person', 'boat', 'truck']
127.0.0.1 - - [25/Jun/2020 23:00:41] "POST /workers HTTP/1.1" 200 -
127.0.0.1 - - [25/Jun/2020 23:00:41] "POST / HTTP/1.1" 200 -
^C
real    13m50,396s
user    12m49,058s
sys     0m3,611s
```

Resultados de 4 workers

```
Processed frames:  431
Average processing time per frame:  6.115446382221656
Person objects detected:  8557
Total classes detected:  6
Top 3 objects detected:  ['person', 'boat', 'truck']
127.0.0.1 - - [25/Jun/2020 21:28:38] "POST / HTTP/1.1" 0 -
^C
real    11m55,251s
user    8m44,296s
sys     0m3,655s
```

Nos resultados são tambem apresentados os frames que ultrapassam o limite de pessoas.