Dear Tsuru Capital,

Not sure if you are used to getting fan-mail but as a technology lover and markets enthusiast I was baffled when I stumbled upon Tsuru Capital. As a curious student who finds fun in problem solving, and is especially interested in the intersection of technology and finance, I was very intrigued by Tsuru Capital's work. If a glance at the attached resume should reflect anything, I hope that it is my dedication to solving problems, eagerness to learn and gain experience, in personal, academic, and professional context.

To briefly introduce myself, I am currently finishing my undergraduate in Computer Science and Applied Mathematics at the University of Edinburgh, and am currently looking for an opportunity where I can showcase what I have learnt, and simultaneously be challenged and learn new skills while working on problems and tasks that excite me! Here at university I have taken many interesting courses that I believe can be essential for this line of work, such as Time Series, Game Theory, Algorithms and Data Structures, and Risk Neutral Asset Pricing.

Talking to a friend who also happens to be one of the biggest Haskell fans, I was introduced to your company. I must admit I did not know much about High Frequency trading, but after researching on my own, I have become very interested in this topic, especially because it requires creativity as well as analytical thinking. While it is important to be able to analyse large quantities of data in an efficient and thorough way, creativity is necessary to craft new interesting trading strategies and find new ways of combining the obtained data.

Interning at JP Morgan's Credit Risk team has helped me develop various skills, especially related to quantitative analysis and modelling. My task was to develop a chat-bot for traders, which could assist them by providing them with the desired data and predictions.

Last summer I had a chance to intern at the Analytics Department at an energy trading firm Gen-I. As a Data Scientist Intern at Gen-I, I assisted in analysing large sets of data and preparing models for future forecasting. Through this experience, I gained an appreciation of the tools and techniques, which are essential for producing reliable and reusable algorithms.

Although I am especially interested in the trading internship opportunities at Tsuru, I wanted to try and prepare a solution for your challenge for developers. Sadly I do not know enough Haskell to code the challenge, but I really liked the task so decided to try and do it in a language that I am more comfortable with, Python. For my solution (accessible here: https://github.com/Pompey21/Tsuru-Prep) I also prepared a report so that it can be better understood. I am now especially curious what can be done with the information gathered from the packets once they are analysed?

I wish to contribute to Tsuru Capital with my knowledge and skills, but I also want to take this opportunity and learn from the leaders in the industry, which would be a truly rewarding experience and I would consider myself lucky to be given such an opportunity.

I look forward to hearing from you. Thank you for your time and consideration.

Kind regards, Marko Horatio Mekjavic

Introduction

As stated before, due to my limited knowledge of Haskell, I decided to try and implement the solution in a language I feel more comfortable with - Python. I am completely aware that Python is a much simpler language which makes it also considerably slower than Haskell or any other similar language. However, the main goal of this task for me was to learn the basics about networks and packets, how to handle large sums of data, and write a system that can handle .pcap files.

For that reason I tried to limit the usage of any sort of libraries that are available for Python and try to implement the majority of algorithms on my own. This turned out to be a very interesting self-learning experience and I have gained a lot of valuable knowledge. It has also introduced me to various fields I previously did know exist or had very limited knowledge about. While it was very exciting trying to play with these packets, I have to admit, I am beyond curious what exactly you do after these packets are intercepted and analysed?

Run the Code

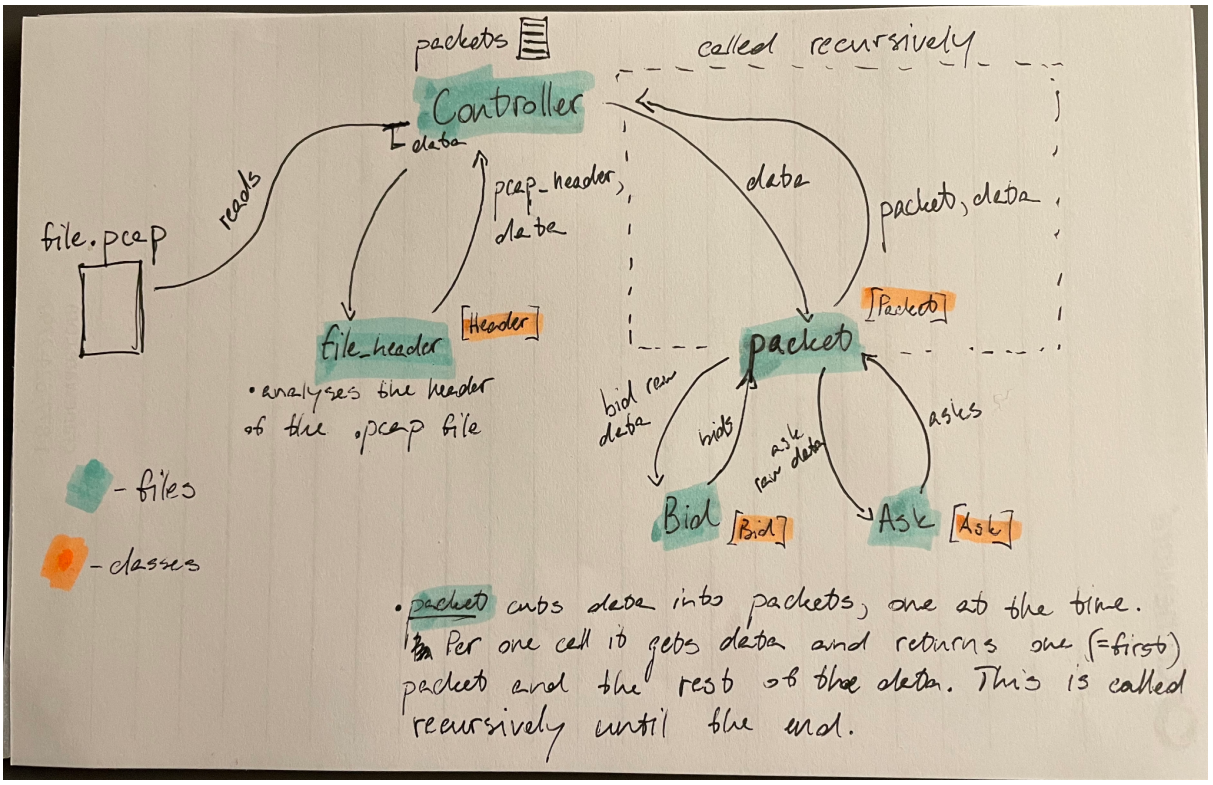
The setup for running the code is fairly simple. Avoiding the unnecessary usage of off-the-shelf solutions in a form of various libraries allows for using the programme without any special environments.

Code runs on Python version 3.7.10 and the only libraries that I have used were the ***sys*** library, which allows for the arguments to be read from the command line, and the ***datetime*** library, which allowed me to time my code as well as to translate UNIX (or Epoch) time to Date Time format that we are used to.

To run the code please navigate to the folder. Once in the folder, run the following command: ***python3 Controller.py -r*** where the last argument is optional - it allows for the packets in the .pcap file to be arranged based on the *quote accepted time*.

Architecture

My code is broken into five different files. I followed a format where I have a main file - named Controller.py - which is responsible for calling all the major operations, from reading a *.pcap* file to ordering and scheduling analysis on the data. Other files, file_header.py, packet.py, Bid.py and Ask.py carry out the execution of the commands scheduled by the Controller.py. The simple architecture is shown below in ***Picture 1***.

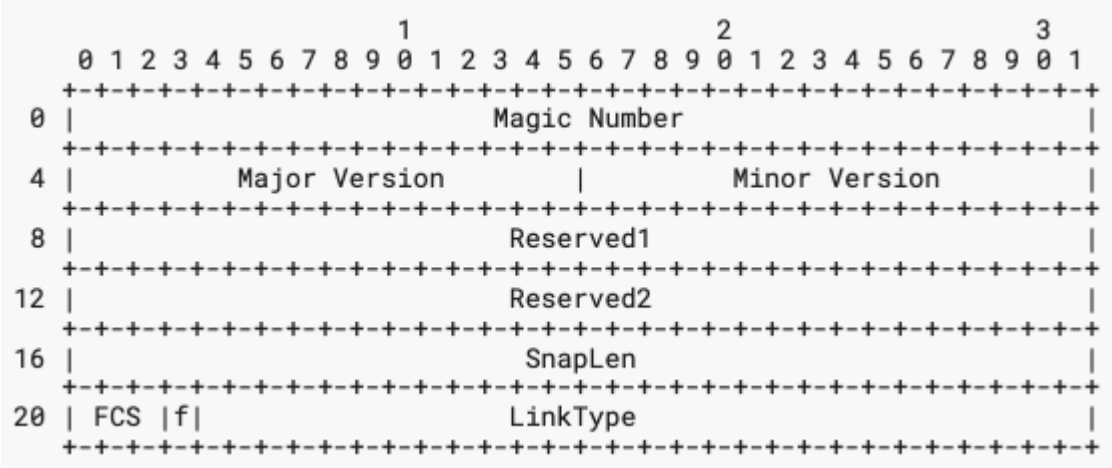


Picture 1: Shows the architecture of my code.

Here is a brief explanation of each class:

<u>Header class</u>

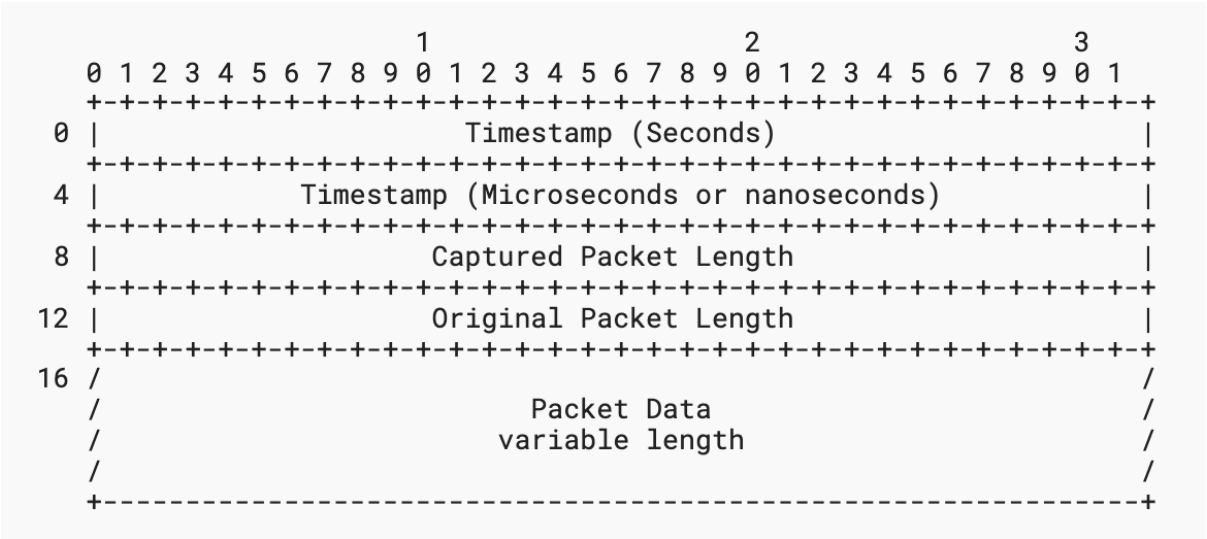
Incorporates the attributes of the .pcap header. So far I only implemented the *Magic number* of the .pcap header, which tells us whether the packet times are recorded in nanoseconds or microseconds. However, if need be, more attributes can easily be implemented here.
The architecture of the .pcap file's header can be seen in *Picture 2* below.



Picture 2: Shows the header break-down of the .pcap file

Packet class

This class is the most important one as it incorporates the structure of a UDP packet. What is interesting about my implementation is the duality in the structure of the UDP packet class. I realised that there is a lot of data inside each packet - such as each bid and ask information - which of course takes time and space to be collected and later on stored. That is why I implemented a class in such a way that it first only acquires the information from the packet's header, such as *package_time, captured_packet_length, original_packet_length, packet_data,* and *flag*, where the *flag* indicates whether the packet contains the **B6034** quote inside its data. Except for the quote information, everything else is contained in the packet record's header, which architecture can be seen in *Picture 3* below.



Picture 3: Architecture of the Packet Record, which is a standard way of storing information about packets coming from the network.

This allows me to easily filter the packets without actually spending more time breaking them down. This is especially important because quite many packets do not contain that quote or are damaged, that is the captured length does not match the original length.

Once the packets are filtered, the ones that remain are then further broken down, that is the necessary information is obtained from their data, such as *bids, asks, data_type,*

market_status_type etc. While the majority of them were fairly easily obtained, I decided to create two additional classes to help me organise the data in an efficient format, the **Bid** and the **Ask** classes. What is more, the timestamp information is recorded in UNIX time, which is seconds and nano- or micro-seconds after 01-01-1970 00:00:00. Since python's library *datetime* already has a method for translating the UNIX time to Date Time format, I decided to use it.
Note, when reading the information from the packet's header, I presumed that it was written in the Big Endian format.

Bid class
This class is responsible for obtaining the information about all the bids. Since each packet carries information about five bids, this class contains five attributes, one for each bid. Each bid is recorded as a tuple containing the quantity and the price of the bid.

Ask class
This class is responsible for obtaining the information about all the asks. Since each packet carries information about five asks, this class contains five attributes, one for each ask. Each ask is recorded as a tuple containing the quantity and the price of the bid.

Solution
As stated, the Controller.py file acts as a main file that calls all the necessary procedures. The first step was to open and read the .pcap file. This was done using simple in-built methods. Learning about the .pcap file and their structure, I realised that it can be separated into a header and body (=data).
Once the data is read from the .pcap file and separated into the header and the body, the Controller.py calls the Header class from the file_header.py, which reads the necessary information as shown in *Picture 2.* Here the most important information is the Magic Number, which states whether the time was recorded in nano- or micro-seconds.
Once that is completed, the Controller.py then proceeds to generate packets from the rest of the data (=body). This is accomplished with the *generate_packets(data)* method which takes the rest of the data (apart from the header!) and calls the Packet class from the *packet.py* file. The Packet class reads all the necessary information of each packet.
Once the packets are filtered, if the code was run with the ***-r*** command, the Controller.py will sort the packets according to their *quote accept time*. Otherwise, it will just proceed with writing the information about the packets into a text file *packets_info.txt.*

Conclusion
This was a very fun and exciting task. From not understanding what the challenge is asking the first time I read it, I managed to learn a great deal and create a somewhat alright solution. If this task even mildly represents daily work at Tsuru Capital, I must admit I am very excited and would very much like to apply for your 3 month interview in the form of an internship. While I am mostly interested in trading internship, I enjoy coding and will definitely try and master Haskell to the extent that I can meaningfully contribute.