Paper Template for COMP90049 Report

Zichun Zhu 784145
zichunz@student.unimelb.edu.au

September 7, 2018

1 Introduction

This project is aimed to develop a multi-clienteed dictionary. In more detials, it should allow concurrent clients to search the meanings of a word, add a new word, and remove an exiting word. The preoject contains two part of system: the server and the client. The server have to be a multi-threaded that accepts multi-clients.

2 Architecture

2.1 Server system

The server is implemented as a thread-per-request architecture by Runnable. The reason mainly is the use of UDP protocol. The components in the server system for following:

- Server: main function.
- Utilities: contains all constants and utilities methods.
- Dictionary: base dataset of dictionary, in format of json.
- RequestObj: the construct of request object, including reading method.
- ResponseObj: the construct of response object, including reading method.
- Response: been implemented with Runnable. Once the socket received a message, one thread is created and each has one instance of response.

Comparing with TCP packets, UDP packets have shorter header. It led that UDP usually is a fast and efficient protocol. This benefit is significate in this project. Each request from the clients does not contain much data, and also the responces are generally less or equaled the length of words' definations that are short. So this implementation is more similar to the DNS server, which requires highly efficiency.

On the other hand, the implementation with UDP also has some drawbacks. The largest issue it faced is dealing with the idempotent methods. It is important to note that UDP is an unreliable connection. Lossing or modifed packets could be happened during transmission. There are three methods for the clients, and two of them, add and remove, have to be idempotent. So how to make sure that these methods only runs once, and regardless with network impact is the problem. The second issue is the limitation on sockets. In TCP, a socket is consisted of the address and port from the server, plus the address and port from the client. Thus it is unlikely that multiple processes take the same socket concurrently. While in UDP, because of the nature of connectionless, the socket is unable to initially bind with the destination. In a multi-thread program, the problem is to map different messages from the same socket to different thread.

2.1.1 Subsubsection

Text of the subsubsection.(see Table 1).

Corpus	Features
AAA	1M words
BBB	spoken corpus (expensive)
CCC	2M words
	free (to academics)

Table 1: The caption of the table

3 Conclusions

Concluding text.[**?**]

References