CS 171 Fall 2024
Programming Assignment I

Assigned: October 2, 2024
Due On: October 16, 2024, 11:59 PM

A Dictionary is a simple data structure that stores a list of pairs (key, value), where each key is unique and has a value associated with it. For example, a dictionary of student phone numbers would have the perm number as the key and the student's phone number as the value. A simple dictionary service could store the dictionary in an array and a server and would support 2 operations: insert (**int** perm, **int** phone), which would insert this pair in the dictionary and lookup(**int** perm), where the server returns the phone associated with this perm number, if the perm number is in the dictionary. The main challenge is that one server might not have enough storage to store all the pairs, hence a **scalable** solution would implement this service using multiple servers, typically using a hash function to determine which key should be stored in which server (more on this later in the course)

1 Project Overview

In this assignment, you will develop a simplified version of a scalable dictionary service. The Dictionary Service will be implemented using 2 servers: a primary server and a secondary server. We will assume 2 clients. A client sends commands (insert or lookup) to the primary. When the primary server receives (perm,phone), it checks if perm is odd, then it stores it locally, otherwise it forwards it to the secondary server, who stores it locally. When the primary receives lookup(key), it checks if the key is odd, it then searches locally and if found, returns the corresponding phone to the client, otherwise returns NOT FOUND. If the key is even, the lookup operation is forwarded to the secondary server, who checks and if found returns the phone to the client, otherwise send NOT FOUND.

2 Implementation Details

- You can decide how to implement the data structure for storing the dictionary at both the primary and secondary servers. Needless to say, this data structure must support both insert and lookup. For this assignment, we do not care how efficient each of these operations is implemented.

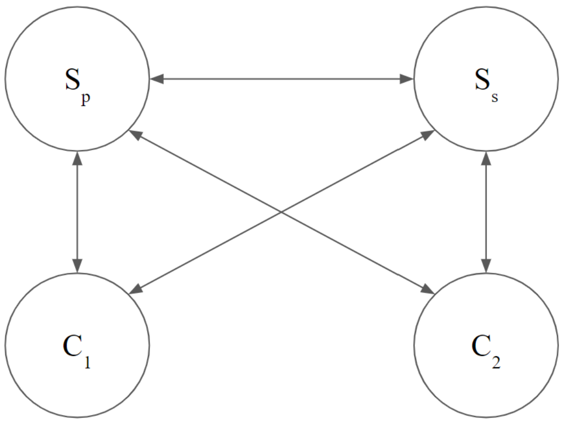

Figure 1: Two clients connected to a primary and secondary server.

3 System Configuration and User Interaction

1. There are exactly 2 client processes and 2 server processes as shown in Figure 1. All processes should run on the terminal. No front-end UI is necessary.

2. Each client process has a unique id: $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$. The primary server has id $\boldsymbol{S_p}$ while the secondary server has id $\boldsymbol{S_s}$. When starting a client process, it should connect to both the servers at pre-defined ports. The two servers should also be connected.

3. Each client must handle the following input from the user:

   - *lookup* + the perm of interest. The client forwards lookup to the primary, who either searches locally or forwards to the secondary. The server responsible for the perm (if odd then the primary, if even then the secondary) returns the corresponding phone or NOT FOUND. The client displays response on its terminal.
   - *insert* + perm + phone. The client forwards the insert request to the primary server, who inserts it (either locally or at the secondary server) and returns Success. The client then displays the response on its terminal.

   The primary server should handle the following input:

   - *dictionary*: The server simply displays the current state of **both** dictionaries stored on the primary and secondary servers sorted by key value

4. To simulate a distributed environment, you can run all 4 processes on the same machine. To simulate message-passing delay over the network, you should **add a 3 second delay before processing a received message**.

5. Use message-passing primitives TCP/UDP for inter-process communications. This will be discussed in detail during the discussion section this week.

6. There is no restriction on the choice of programming language. However, you must use socket programming for communication to simulate an actual network environment.

4 Example Scenarios

When running the four entities – S_p, S_s, C_1, C_2 as separate terminal processes, the output in each terminal should appear as shown below for the specified sequence of commands.

4.1 Sequence of commands

Note that the prefix before each command denotes where it is run. (C1 denotes Client1 and C2 denotes Client2, PS denotes Primary Server)

```
C1: insert 1111111 999999999
C1: insert 2222222 888888888
C2: insert 3333333 777777777
C2: insert 4444444 666666666
C1: lookup 3333333
C1: lookup 4444444
C2: lookup 1111111
C2: lookup 2222222
C2: lookup 5555555
C2: lookup 6666666
PS: dictionary
```

4.2 Expected output

4.2.1 Primary Server S_p

```
Cmd: insert 1111111 999999999
Output: Successfully inserted key 1111111
Cmd: insert 2222222 888888888
Output: Forwarding to secondary server
Cmd: insert 3333333 777777777
Output: Successfully inserted key 3333333
Cmd: insert 4444444 666666666
Output: Forwarding to secondary server
Cmd: lookup 3333333
Output: 777777777
Cmd: lookup 4444444
Output: Forwarding to secondary server
Cmd: lookup 1111111
Output: 999999999
Cmd: lookup 2222222
Output: Forwarding to secondary server
Cmd: lookup 5555555
Output: NOT FOUND
Cmd: lookup 6666666
Output: Forwarding to secondary server
```

```
Cmd: dictionary
Output: primary {(1111111, 999999999), (3333333, 777777777)}, secondary {
(2222222, 888888888), (4444444, 666666666)}
```

4.2.2 Secondary Server S_s

```
Cmd: insert 2222222 888888888
Output: Successfully inserted key 2222222
Cmd: insert 4444444 666666666
Output: Successfully inserted key 4444444
Cmd: lookup 4444444
Output: 666666666
Cmd: lookup 2222222
Output: 888888888
Cmd: lookup 6666666
Output: NOT FOUND
```

4.2.3 Client1 C_1

```
insert 1111111 999999999
Output: Success
insert 2222222 888888888
Output: Success
lookup 3333333
Output: 777777777
lookup 4444444
Output: 666666666
```

4.2.4 Client2 C_2

```
insert 3333333 777777777
Output: Success
insert 4444444 666666666
Output: Success
lookup 1111111
Output: 999999999
lookup 2222222
Output: 888888888
lookup 5555555
Output: NOT FOUND
lookup 6666666
Output: NOT FOUND
```

5 Submission

Details on how to submit your assignment will be posted next week.