ECE419 M1 Report

Thomas Kingsford
Marcel Mongeon
Zhihao Sheng

28 January 2018

1 Design and Decisions

Architecture Refer to Figure 1 in the Appendices for an architecture diagram.

KVClient

KVStore

KVServer

CommMod The CommMod class handles client and server communications. The server registers as a listener and receives KVMessages by way of a callback. The server must respond to each received KVMessage with a response message. Clients Connect() then send messages via SendMessage(), which returns a KVMessage response or else times out if the server fails to respond quickly enough.

TLVMessage The TLVMessage is an implementation of the KVMessage interface. it implements a modification of tag-length-value encoding. It (un)marshals a KV message as a sequence of bytes in which:

1. The first byte is the ordinal value of the StatusType enum, referred to as a 'tag'.
2. The second byte is the length of the key $L_K \in [0, 255]$. This protocol imposes an upper limit on key size of 255 bytes.
3. For messages containing a value (the existence of a value is fully determined by the tag), the third byte is the length of the value $L_V \in [0, 255]$. This protocol imposes an upper limit on key size of 255 bytes. This could be trivially extended - for instance, the use of four bytes would give a maximum length of $2^{3}2 - 1 \approx 1$ billion bytes
4. The following L_K bytes are the key.
5. If there is a value, the following L_V bytes are the value.

LRUCache

LFUCache

FIFOCache

LockManager

FilePerKeyKVDB

2 Performance Evaluation

3 Test Cases

3.1 CacheTests

3.2 CommModTests

3.3 ConnectionTest

3.4 InteractionTest

3.5 KVDBTests

3.6 LockManagerTest

3.7 SocketTest

3.8 StoreServerTests

3.9 TLVMessageTest

4 Appendices

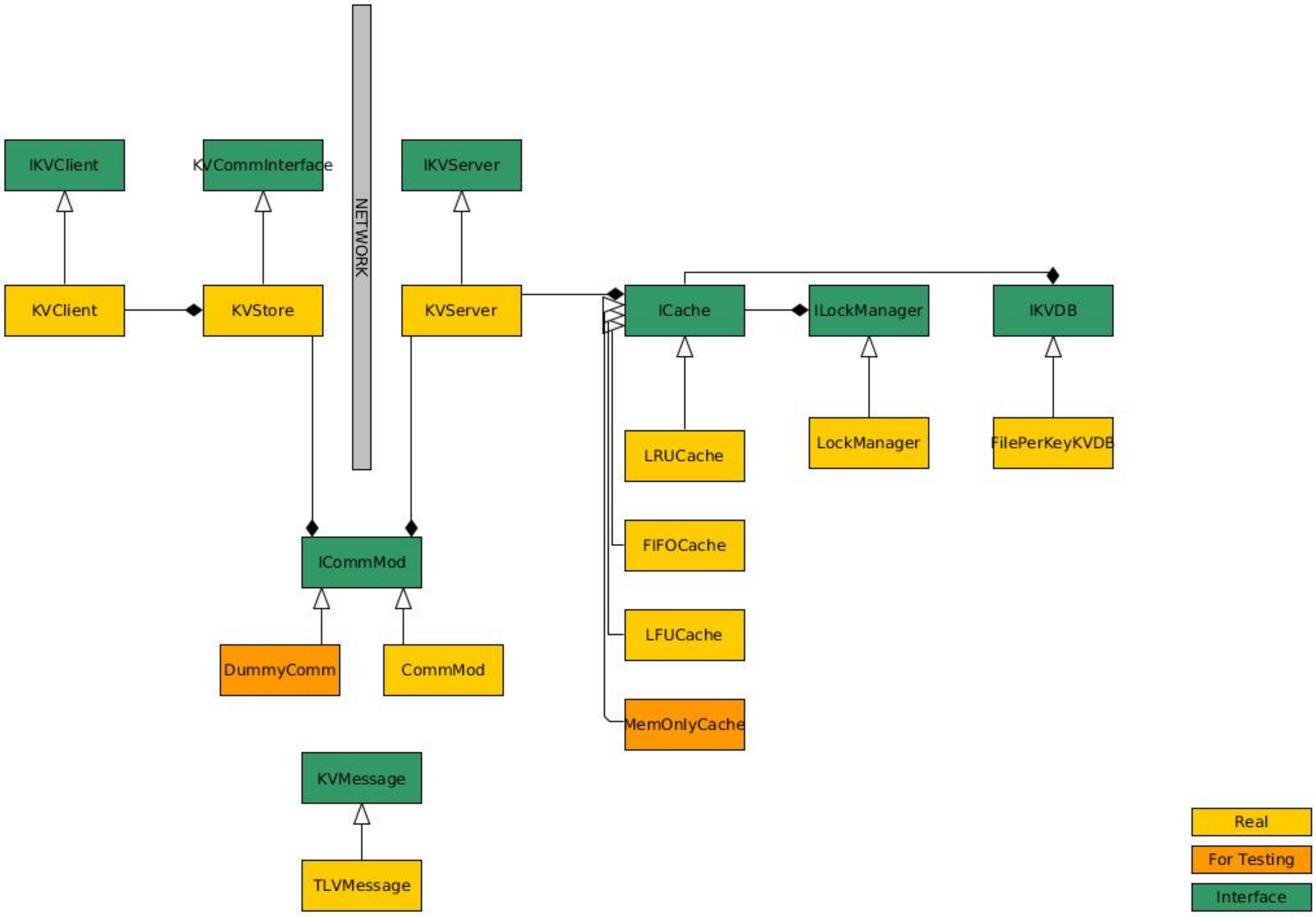

Figure 1: Architecture Diagram