Beernet's API

Boris Mejías

May 31, 2011

1 Introduction

> **WARNING!** This document corresponds to the API of Beernet-0.8. It does not integrate the new API with secrets, but Beernet-0.9 is still compatible with this document.

Beernet is provided as a Mozart-Oz[1] library. To use it, the program needs to import the main functor to create a peer to bootstrap a network, or to join an existing one using the reference of another peer. Importing the main functor and creating a new peer works as follows:

```
functor
import
   Beernet at 'beernet/pbeer/Pbeer.ozf'
define
   Pbeer = {Beernet.new args(transactions:true)}
```

Interacting with the peer is done by triggering an event as follows:

```
{Pbeer event(arg1 ... argn)}
```

We list now the different events that can be triggered on Beernet's peers. Even though Beernet's architecture is organized with layers where Trappist is the upper most one, the architecture does not prevent the access to lower layers because their functionality is important to implement applications.

2 Relaxed Ring

The following events can be used to get access to the functionality provided by the relaxed ring layer. It mostly provides access to peer's pointers and other information of the structured overlay network.

[1] The Mozart Programming System, http://www.mozart-oz.org

2.1 Basic Operations

- join(RingRef) Triggers joining process using RingRef as access point.
- lookup(Key) Triggers lookup for the responsible of Key, which will be passed through the hash function.
- lookupHash(HashKey) Triggers lookup for the responsible of HashKey without passing HashKey through the hash function.
- leave Roughly quit the network. No gently leave implemented.

2.2 Getting Information

- getId(?Res)
  Binds Res to the id of the peer
- getRef(?Res)
  Binds Res to a record containing peer's reference with the pattern
  pbeer(id:<Id> port:<Port>)
- getRingRef(?Res)
  Binds Res to the ring reference
- getFullRef(?Res)
  Binds Res to a record containing peer's reference and ring's reference with the pattern ref(pbeer:<Pbeer Ref> ring:<Ring Ref>)
- getMaxKey(?Res)
  Binds Res to the maximum key in ring's address space
- getPred(?Res)
  Binds Res to the reference of peer's predecessor
- getSucc(?Res)
  Binds Res to the reference of peer's successor
- getRange(?Res)
  Binds Res to the responsibility range of peer with the pattern From#To, where From and To are integers keys.

2.3 Other Events

- refreshFingers(?Flag) Triggers lookup for ideal keys of finger table to refresh the routing table. Binds Flag when all lookups are replied.

- injectPermFail Peer stop answering any message.
- setLogger(Logger) Sets Logger as the default service to log information of the peer. Mostly used for testing and debugging.

3 Message Sending

This section describe the events that allow applications to send and receive messages to other peers.

- send(Msg to:Key)
  Sends message Msg to the responsible of key Key.
- dsend(Msg to:PeerRef)
  Sends a direct message Msg to a peer using PeerRef.
- receive(?Msg)
  Binds Msg to the next message received by the peer, and that it has not been handled by any of Beernet's layer. It blocks until next message is received.

4 DHT

Beernet also provides the basic operations of a distributed hash table (DHT). None of this uses replication, therefore, there are no guarantees about persistence.

- put(Key Val)
  Stores the value Val associated with key Key, only in the peer responsible for the key resulting from applying the hash function to key Key.
- get(Key ?Val)
  Binds Val to the value stored with key Key. It is bound to the atom `'NOT_FOUND'` in case that no value is associated with such key.
- delete(Key)
  Deletes the item associated to key Key.

5 Symmetric Replication

The symmetric replication layer does not provides an interface to store values with replication, but it does provides some functions to retrieve replicate data, and to send messages to replica-sets.

- bulk(Msg to:Key)
  Sends message Msg to the replication set associated to key Key.

- getOne(Key ?Val)
  Binds Val to the first answer received from any of the replicas of the item associated with key Key. If value is `'NOT_FOUND'`, the peer does not bind Val until it gets a valid value, or until all replicas has replied `'NOT_FOUND'`.

- getAll(Key ?Val)
  Binds Val to a list containing all values stored in the replica set associated to key Key.

- getMajority(Key ?Val)
  Binds Val to a list containing the values from the replica set associated to key Key. It binds Val as soon as the majority is reached.

6 Trappist

Trappist provides different protocols to run transactions on replicated items. Due to their specific behaviour, they have different interfaces.

6.1 Paxos Consensus

- runTransaction(Trans Client Protocol)
  Run the transaction Trans using protocol Protocol. The answer, commit or abort is sent to the port Client. Currently, the protocols supported by this interface are twophase and paxos, for two-phase commit and Paxos consensus with optimistic locking. For eager locking, see the interface in Section 6.2.

- executeTransaction(Trans Client Protocol)
  Exactly the same as runTransaction. Kept only for backward compatibility.

Inside a transaction, there are three operations that can be used to manipulate data.

- write(Key Val)
  Write value Val using key Key. The new value is stored at least in the majority of the replicas. Updating the value gives a new version number to the item.

- read(Key ?Val)
  Binds Val to the latest value associated to key Key. Strong consistency is guaranteed by reading from the majority of the replicas.

- remove(Key)
  Removes the item associated to key Key from the majority of the replicas.

6.2 Paxos with Eager Locking

- getLocks(Keys ?LockId)
  Get the locks of the majority of replicas of all items associated to the list of keys Keys. Binds LockId to an Oz name if locks are successfully granted, and to error otherwise.

- commitTransaction(LockId KeyValuePairs) update all items of the list KeyValuePairs which must be locked using LockId. Each element of the list KeyValuePairs must be of the form <key>#<value>

6.3 Notification Layer

- becomeReader(Key)
  Subscribes the current peer to be notified about locking and updates of the item associated to key Key. Notification are received using the receive event described previously.

6.4 Key/Value-Sets

- add(Key Val)
  Adds the value Val to the set associated to key Key.

- remove(Key Val)
  Removes value Val from the set associated to key Key.

- readSet(Key ?Val)
  Binds Val to a list containing all elements from the set associated to key Key.