Issue #1: RedisClientWorker may throw UnhandledMessageException

In the preStart method of RedisClientSession the following execution

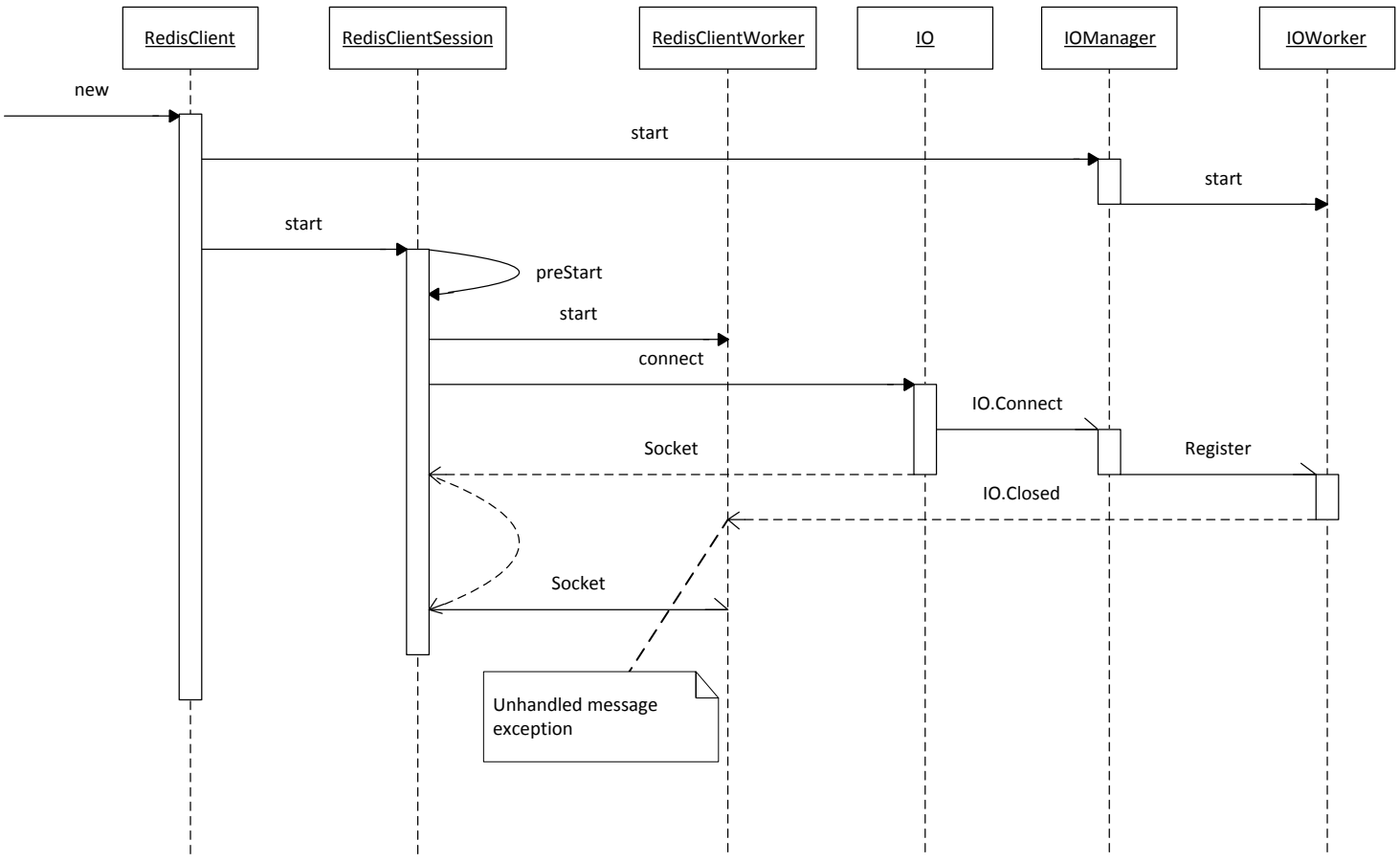
```
socket = IO.connect(ioManager, host, port, worker)
worker ! Socket(socket)
```

may lead to a sequence of events, which will cause the exception.

RedisClientWorker only responds to an IO.Closed message if the socket variable is the same as the handle from the message. Thus if the worker receives the IO.Closed message before the Socket message the exception will be thrown. An example scenario is a refused connection.

Github: https://github.com/alexdoth/fyrie-redis/issues/1

Message Sequence



Issue #2: Request and result callbacks may stall or kill RedisClientSession/Worker

Both RedisClientSession and RedisClientWorker execute the callback functions in their own thread

```
RedisClientSession:
    def onRequest(req: RequestMessage): Unit = {
    if (config.retryOnReconnect) waiting enqueue req
    requests += 1L
    if (requestCallbacks.nonEmpty) {
      val atTime = System.currentTimeMillis
      requestCallbacks foreach (_(requests, atTime)) // <-- CALLBACK INVOCATION
    }
  }

RedisClientWorker:
  def onResult() {
    if (config.retryOnReconnect) sendToSupervisor(Received)
    results += 1L
    if (resultCallbacks.nonEmpty) {
      val atTime = System.currentTimeMillis
      resultCallbacks foreach (_(results, atTime))  // <-- CALLBACK INVOCATION
    }
  }
```

There are a few problems with this:

- The callback may throw an exception, which is not caught in any of the invocations. Thus unexpectedly shutting down the active actor.

- The callback may take unspecified amount of time. Thus the actor will stall until all the callback complete. Possibly triggering future timeout exceptions if the callbacks take long time to complete.

- The callback will be executed by the current Actor thread, thus if the callback uses shared data-structures there may be potential race-conditions.

Github: https://github.com/alexdoth/fyrie-redis/issues/2

Issue #3: RedisClientWorker fails when received data is invalid type

If the received type is none of RedisString, RedisError, RedisInteger, RedisBulk, or RedisMulti subsequent client requests will result in future timeout exceptions.

As an example if the RedisClientSession actor receives the following messages

```
val future1 = worker ? Run
worker ! IO.Read(handle ByteString("invalid redis type" + EOL)
future1.get // throws FutureTimeoutException

val future2 = worker ? Run
worker ! IO.Read(handle ByteString("+valid redis string" + EOL)
future2.get // throws FutureTimeoutException
```

Then the current request and any subsequent requests result in FutureTimeoutException thrown to the sender.

Github: https://github.com/alexdoth/fyrie-redis/issues/3

Issue #4: RedisClientSession may fail to complete client request

To complete the sender's future RedisClientWorker needs to receive Run before IO.Read. If the order is not met, the sender will receive a future timeout exception for the request.

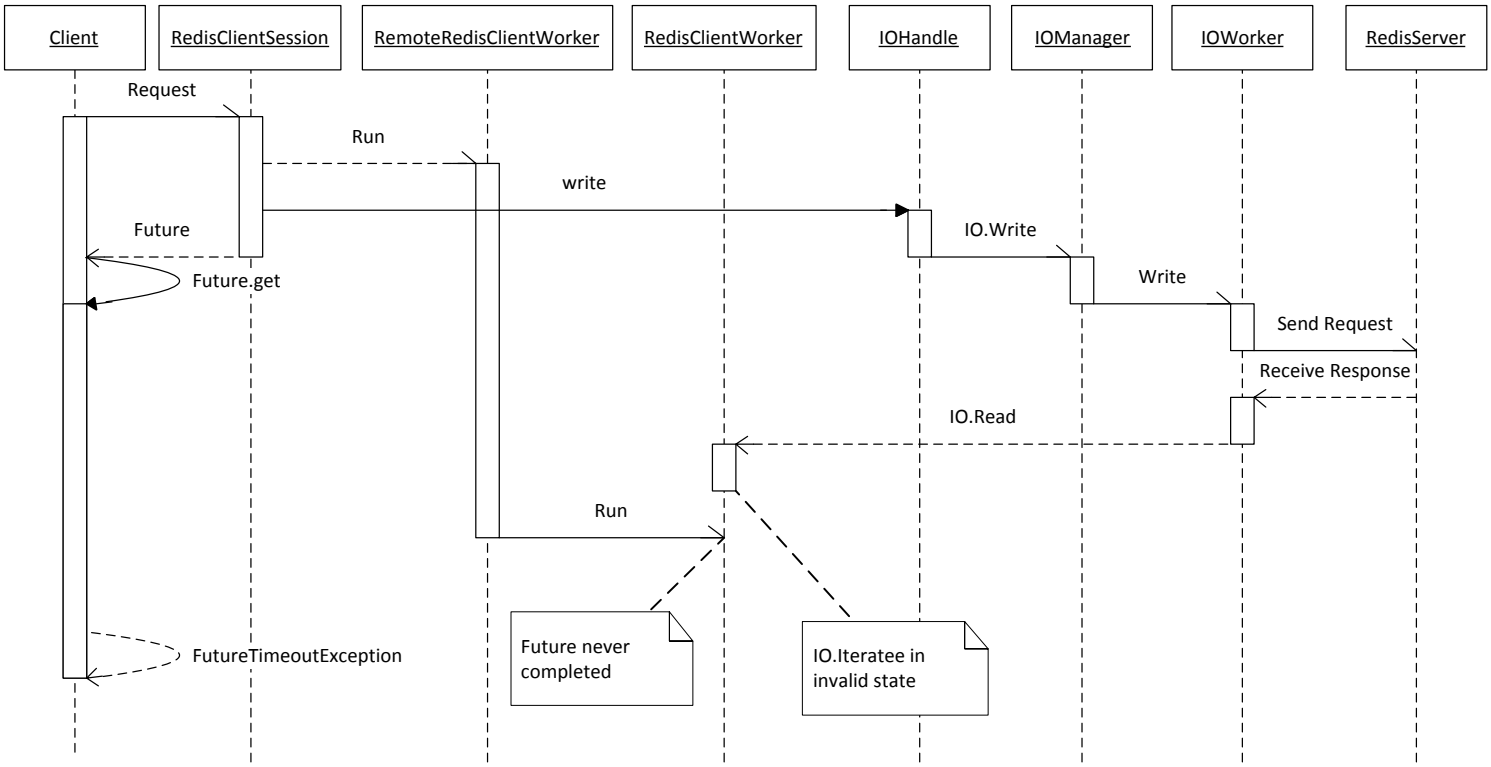
In RedisClientSession the response to a request involves the following

```
def receive () = {
....
    case req: Request ⇒
      worker forward Run
      socket write req.bytes
      onRequest(req)
....
}
```

It can happen that the IOManager sends IO.Read before the Run message has been forwarded to the worker.

Github: https://github.com/alexdoth/fyrie-redis/issues/4

Message Sequence



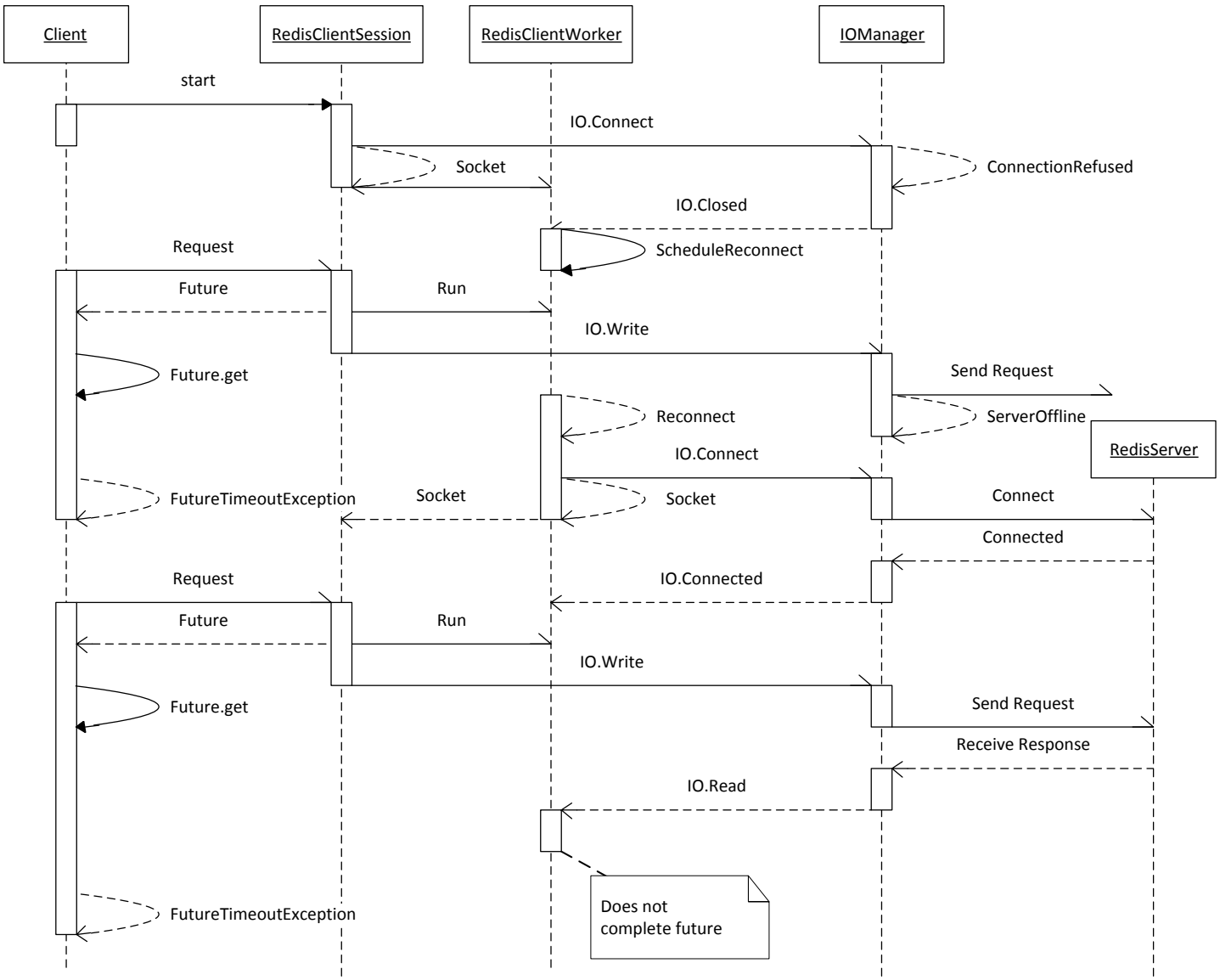
Issue #5: FutureTimeoutException thrown when retryOnReconnect is disabled

When retryOnReconnect is set to false and the following scenario occurs:

- Start session
- Submit a request
- Connection refused
- Retry connecting
- Get result from first request -> Expected FutureTimeoutExpcetion
- Connection to server successfully established
- Submit another request
- Get result from request -> should complete successfully but unexpected FutureTimeoutException is thrown again

Github: https://github.com/alexdoth/fyrie-redis/issues/5

Message Sequence



Issue #7: RedisSubscriberSession doesn't shutdown its worker

Sending a "Disconnect" message to the RedisSubscriberSession actor leaks and doesn't shutdown the RedisClientWorker actor initialized and allocated in RedisSubscriberSession . Resources are wasted and need to be reclaimed by gracefully shutting down the worker actor.

Github: https://github.com/alexdoth/fyrie-redis/issues/7

Issue #8: RedisSubscriberSession doesn't accept new socket from worker

Upon closing the connection the worker will retry to connect if auto-reconnect is enabled.
It will also send the supervisor the new socket to be used. However RedisSubscriberSession doesn't handle Socket messages. Thus any subsequent requests to RedisSubscriberSession will not complete successfully.

Github: https://github.com/alexdoth/fyrie-redis/issues/8

Message Sequence