Utilizing Transactional Events in Haskell in the Implementation of Message Broker Middleware

Master's Project Proposal and Overview
Christopher R. Wicks
9/4/2017

Introduction

In this project, I will explore the use of Transactional Events, a novel concurrency abstraction in the functional paradigm, in the implementation of message broker middleware. In particular, client and server software utilizing STOMP will be implemented. The performance and convenience of the abstraction will be examined in a real-world implementation scenario.

TE Haskell

Transactional events are a high-level concurrency abstraction that combine first-class synchronous message-passing events with all-or-nothing transactions [1]. Transactional events were first given a formal semantics by Matthew Fluet and Kevin Donnely in 2006. They posit that their approach enables more elegant solutions than traditional approaches to problems in concurrent programming. They have provided an experimental extension to Haskell that implements these semantics, which I will be utilizing in the development of the programs that are to be written for this project.

STOMP

STOMP (the Streaming Text Oriented Messaging Protocol) is a fairly simple, text-based messaging protocol, similar in design to HTTP [2]. The flexibility and expressiveness of Haskell's type system feels like a natural fit in the implementation of a message-passing protocol such as STOMP. Moreover TE Haskell should provide an elegant abstraction in the implementation of both the message broker middleware and client APIs.

Project Goals

For this project, the main goal is to utilize TE Haskell to implement a message broker and client software/APIs to interact with the message broker. I will attempt to identify points in the development of this software in which TE Haskell provides a simpler/more elegant abstraction, or enables an approach that would be impossible or notably more complex given other models of concurrency. Performance metrics will be considered a secondary goal to the implementation and evaluation of the abstraction, but will be taken nonetheless and compared with existing implementations of the protocol. If time permits, an extended goal would be to support other messaging protocols in the software, but this could also be future work.

References

[1] Kevin Donnelly & Matthew Fluet (2006): *Transactional Events.*In: *11th ACM International Conference on Functional Programming*.
[2] STOMP Specification. Retrieved from https://stomp.github.io/.