SINF2345 Languages and Algorithms for Distributed Applications
Project report

Martin TRIGAUX
SINF22MS
martin.trigaux@student.uclouvain.be

Bernard PAULUS
SINF22MS
bernard.paulus@student.uclouvain.be

May 6, 2012

Contents

Introduction

The aim of the project is to build a distributed banking system. The system is distributed in a way such that any bank node is award of the full banking system state at all time but only atomics messages are transfered between nodes.

To build this project, we decided to use the programming language Erlang. We believe this language is highly adequate for concurrent applications and that motivated us to use it for this project. This language handles also very well events and made easier the implementation of proposed algorithm in the reference book.

1 Architecture

To realise the distributed system, we used the reference book "Reliable and Secure Distributed Programming" by C. Cachin, R. Guerraoui and L. Rodrigues second edition. We used a architecture based on layers and encapsulation of messages. Each module is connected to other and behave upon events (recieved messages from other modules).

Links

We used the **perfect link** module[1] for basic point to point communication. These are at the lowest level of our architecture and simply transmit messages to above modules. The perfect link ensures *reliability* and the *no duplication* property.

Broadcasts

We used the **best effort broadcast** module[2] for implement the broadcast messages between nodes. This module transmit messages to the adequate perfect links. As we want to handle failing nodes, we used a **reliable broadcast** algorithm[3]. To ensure the *agreeement* property, we implemented the **eager reliable broadcast** algorithm[4].

Failure detector

To handle failing nodes in a partially synchronous system, we used an **eventual failure detector** module[5]. This was implemented using the **increasing timeout failure detector** algorithm[6].

Leader detector

monarchical eventual leader detector eventual leader detector

Consensus

read write epoch change leader based epoch change leader driven consensus consensus

Total order broadcast

consensus based total order broadcast total order broadcast

2 Manual

3 Conclusion

[1]Module 2.3 p37 in reference book
[2]Module 3.1 p75 in reference book
[3]Module 3.2 p77 in reference book
[4]Algorithm 3.3 p80 in reference book
[5]Module 2.8 p54 in reference book
[6]Algorithm 2.7 p55 in reference book