HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

FHO Fachhochschule Ostschweiz

roadster advanced web scada system

Bachelor thesis

Roadster High Availability

Manuel Schuler, Patrik Wenger

for industry client

mindclue GmbH

supervised by

Prof. Farhad Mehta

Fall semester 2016

Abstract

TODO introduction

TODO approach and technologies

TODO result

Declaration of Originality

We hereby confirm that we are the sole authors of this document and the described changes to the Roadster framework and libraries developed.

TODO any usage agreements or license

Thank You

TODO anyone we'd like to thank

Management Summary

Initial Situation

TODO describe initial situation, not too technical

Roadster is a next generation monitoring application.

Software Development Process

TODO describe decision to use RUP/Scrum TODO maybe describe what project management tools we'll be using

Project Phases

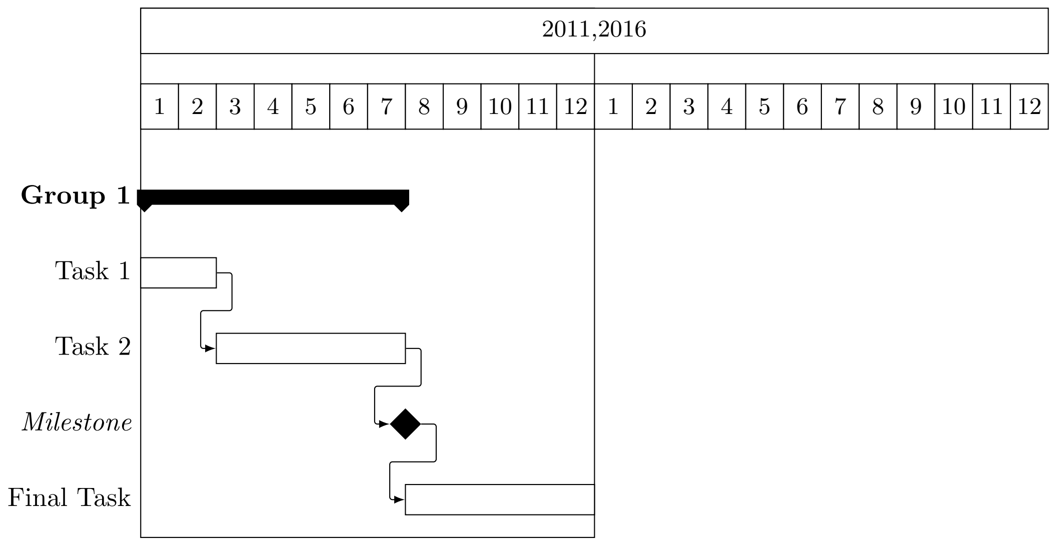
TODO describe this phase in retrospection

Inception

TODO include Gantt chart for this phase TODO describe this phase in retrospection

Elaboration

TODO include Gantt chart for this phase TODO describe this phase in retrospection



Construction

TODO include Gantt chart for this phase TODO describe this phase in retrospection

Transition

TODO include Gantt chart for this phase TODO describe this phase in retrospection

Results

TODO describe results

Contents

List of Figures

List of Tables

Listings

Part I

Technical Report

Chapter 1

Context

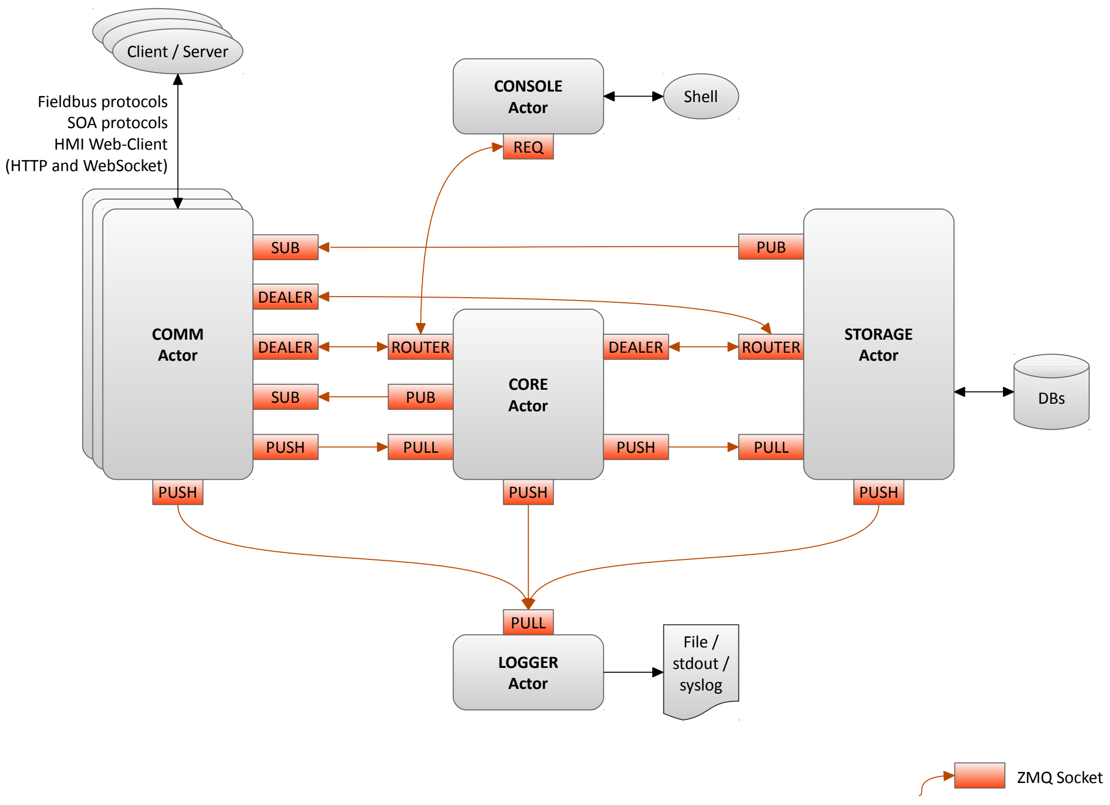
TODO what's this thesis about (general scope)

Initial Situation

TODO What's Roadster and its goals

Software Architecture

TODO Roadster architecture



Goals

TODO mandatory goals

Optional Goals

TODO optional goals

Requirements

TODO the requirements

In descending priority:

1. multi-node CSP
2. single-level HA
3. multi-level HA
4. persistence synchronization
5. security
6. OPC UA HA (optional)

The following sections explain the requirements in greater detail.

Cluster

This could also be called "Multi-node CSP".

- this is to allow running Roadster in a hierarchical setup
- new COMM actors for inter node communication
- usually 2 (or 3) levels of Roadster nodes
- common cases:
  - - single level, single node (legacy)
  - - single level HA
  - - multi level, HA at root node only
- exotic cases:
  - - multi level, HA at bottom
  - - multi level, HA in middle
- every subtree can live on autonomously
- only node A has write access to values on A (to avoid uncertain situations involving race conditions), e.g.:
  - - a forced value coming from the web UI comes through a command,
  - - routed to the relevant node, where it is applied,
  - - and then synced (up via DEALER and down via PUB, we suppose)
- KISS

```
        C
      /   \
     A     B
```

Single Level HA

This is where there's a node pair directly connected to a PLC. Both nodes have read/write access to the PLC, but only one of the nodes (the active one) must do so. The nodes must automatically find consensus on who's active. The passive one must automatically take over in case the active one is confirmed to be dead.

Multi level HA

This is where a node pair is the parent of one or more other nodes (subnodes).

Persistence Synchronization

This is about the synchronization of the TokyoCabinet databases. Data flow is from south to north (towards the root node), so the root node collects and maintains a replication of the persisted data of all subnodes, recursively.

- autonomous
- not same as CHP
- data only flows from bottom to top

Security

- transport needs to be secure (encrypted and authenticated)
- TODO verify requirements with Andy (we didn't really discuss this during the meeting)

OPC UA HA

- provide standardized interface upwards from HA pairs

Non-Functional Requirements

TODO the NFRs

Coding Guidelines

- basically Ruby style guide[1]
- method calls: only use parenthesis when needed, even with arguments (as opposed to [2])
- 2 blank lines before method definition (slightly extending [3])

[1]`https://github.com/bbatsov/ruby-style-guide`
[2]`https://github.com/bbatsov/ruby-style-guide#method-invocation-parens`
[3]`https://github.com/bbatsov/ruby-style-guide#empty-lines-between-methods`

- YARD API doc, 1 blank comment line before param documentation, one blank comment line before code (ignoring [4])
- Ruby 1.9 symbol keys are wanted (just like [5])
- align multiple assignments so there's a column of equal signs

Task Description

TODO our tasks

[4]https://github.com/bbatsov/ruby-style-guide#rdoc-conventions
[5]https://github.com/bbatsov/ruby-style-guide#hash-literals

Chapter 2

About ZMQ

TODO crash course in ZMQ/CZMQ/CZTop
TODO strong abstraction (one socket for many connections, connection handling transparent, transport and encryption transparent, no concept of peer addresses)
TODO brokerless/with broker, up to you
TODO basic patterns
TODO extended patterns
TODO not only a "MOM", but a multi threading library (Actor pattern)

Chapter 3

Port

Concept

TODO explain binding options out there, why CZTop
TODO explain concept of exchanging ffi-rzmq with CZTop

Implementation

TODO show certain excerpts how it was done

Chapter 4

Cluster

Concept

TODO explain planned multi node setup
TODO election/design of appropriate protocol
TODO explain Clustered Hashmap Protocol (I guess)

```
* PCP: use DIM to know node tree and determine next hop for (dialog or fire+↩
    ↪ forget) messages

* decide on sync variant
  - variant 1
    - always sync on self-subtree only
    - con: no copy of remaining tree

  - variant 2:
    - always sync on complete tree
    - get snapshot and merge own subtree

  - variant 3:
    - make it configurable: either sync on subtree or complete tree

* node topology in DSL, static file shared on all nodes, read by each actor on ↩
    ↪ startup
* specific config file on each node (conf.rb) knows its own place in topology
    conf.system_id = "nodes.root" # no HA
    OR
    conf.system_id = "nodes.root_ha.foo" # with root HA, subnode A directly ↩
        ↪ below root level

 * maybe a HA pair is one DIM object, has one name, but two IP addresses (↩
     ↪ primary and backup, in order)
```

Implementation

TODO explain how to implement/integrate CHP

Chapter 5

High Availability

TODO we have two different kinds of HA TODO explain the kinds of failures we want to be able to handle

Single Level HA

Concept

```
 - this is different from what's described in the zguide because the concept of↩
    ↳  client requests is missing here (PLCs don't request anything)
 - life sign from one node to the other through some continually updated PLC ↩
    ↳ value
 - mark active HA peer in DIM, OR PUSH-PULL \& different route back
   - evaluate in elaboration
 - side note: PUSH-PULL is probably not feasible, because message are sent to ↩
    ↳ inactive pull anyway, until queue full
```

Implementation

TODO explain how we implement/integrate it

Multi Level HA

Concept

Finding consensus should be easier here, as it's closely related to the CHP described in the zguide.

```
```

Implementation

TODO explain how to implement/integrate Binary Star

Chapter 6

Persistence Synchronization

Concept

```
  - super node requests for delta of TC periodically
```

Implementation

TODO explain how we implement/integrate it

Chapter 7

Highly Available OPC UA Server

TODO This is the optional goal.
TODO explain new opportunity for OPC UA HA server

Concept

TODO describe whatever needs to be described

- study standard
- use Andy's gem
- according to Andy, this should be a simple thing

Implementation

TODO describe whatever needs to be described

Chapter 8

Conclusion

TODO write conclusion, we're the best and everything is awesome

Part II

Appendix

Appendix A

Self Reflection

TODO how did we perform, completion of goals, accuracy of estimated efforts, efficiency, resourcefulness

Appendix B

Project Plan

Organization

TODO roles, how we organize ourselves and how we communicate with each other

Appendix C

Infrastructural Problems

TODO describe serious problems here, if any

Project Management Software

TODO Github/Trello/Harvest/Everhour/Elegantt/Ganttify/Redmine

TODO add any other appendix chapters here, like detailed ZMQ stuff, usage manuals, ...