Project proposal
Open-source implementation of DCXP

Jonas Bäckström, Henrik Hagsved
Department of Information Technology and Media
Mid Sweden University

November 23, 2011

Contents

Chapter 1

Introduction

In a near future the amount of sensors linked to electronical equipment will increase rapidly and thus also the need for distributing this new information to consuming applications and systems. In such an environment a centralized solutions is not applicable due to the massive amount of connecting nodes, which would bring any centralized server to its knees. This is where a real-time distributed system of context information with DCXP is a possible solutions - which we know from earlier research - performs better than other available solutions. [1][2][3]

The purpose of this project is to create a fully open-source solution without depending libraries. In addition the research-team's current solutions are lacking certain core features or are not fully distributed and are thus inappropriate to use as bottom layer in a big scale system.

1.1 Functional demands

Here follows the functional demands identified in the project.

- REGISTER_UCI - A CUA uses REGISTER to register the UCI of a CI with the CS.
- RESOLVE_UCI - In order to find where a CI is located, a CUA must send a RESOLVE to the CS.
- GET - Once the CUA receives the resolved location from the CS, it GETs the CI from the resolved location.
- SUBSCRIBE - SUBSCRIBE enables the CUA to start a subscription to a specified CI, only receiveing new information when the CI is updated.
- NOTIFY - The source CUA provides notification about the latest information to subscribing CUAs every time an update occurs or if asked for an imediate update with GET
- SET - Sets the CI at the resolved location.

1.2 Non-functional demands

Here follows the non-functional demands identified in the project. The primary requirements must be implemented for the project to be considered successful while the secondary requirements will be included give time.

1.2.1 Primary

- Scalability - Logarithmic (or better).
- No External Licenses - We control the license and any exploitation opportunity.
- Extendibility - So new features can be added without redistributing the whole project.

1.2.2 Secondary

- Android - A node as a service on an android phone.
- Stability - Manage volatile nodes joining and leaving (With high churn rate).
- Reliable - All queries should return an answer.

Chapter 2

Project plan

We intend to use the following tools developing the application:

- Eclipce - Opence source IDE.
- Android SDK - Software development kit connectable to Eclipse.
- Github - Version managements system and code-sharing forum.

2.1 Project management

In this project we intend to use a agile SCRUM-like [4] project management approach.

- Iterative - Short iterations with prototype testing.
- External communication - Close connection to project owners (weekly meetings).
- Internal communication - Daily meetings for the project group.
- Pair-programming - To ensure higher quality on critical parts.

Chapter 3

Timeplan

See separately attached Gantt-schedule ('Ganttschedule.png') describing the workflow and estimated time distribution.

Chapter 4

Expected Deliverables

In the end of the project the five functions raised in the functional demands should be merged to an implementation of the DCXP-core and then connected to a DHT (e.g. Chord, P-GRID). In addition the DCXP-implementation should fulfill all primary non-functional demands specified.

Another important part - to easily continue on this project - is extensive documentation with detailed explanations describing the system architecture, relations with adjacent layers and future work.

The final product will be handed over to the project owners in three parts;

- The project report - Sent via email and/or in printed format.
- The documentation - Can be found in the Github repository.
- The code - Can be found in the Github repository.

4.1 Verification

The functional demands of the software will be tested using existing sensor implementations provided by the project owner. If all functional demands work, this requirement is considered a success.

Scalability will be verified by simulating multiple nodes on some computers and measuring the time it takes to find a requested Context Information (CI). The measured time should increase logarithmically compared to the increase in nodes.

A review of the code will reveal any external licenses and if no external licenses exist it is considered a success. It will still be considered a success if external licenses exist and all of them have a satisfactory motivation for being included and the software can be modified to not need them in the future.

Verification of extendibility will be performed by code review and if all parts are working fully towards interfaces, then this requirement is considered a success.

Android compatibility will be tested by using the software on an actual Android phone. If the software can be run as a service on an Android phone, this secondary requirement is considered a success. A partial success can be achieved by the software not containing anything that is not supported by Android.

Stability verification will be conducted by having computers simulate multiple nodes and having them join and leave. For this secondary requirement to be considered a success, 20% of the nodes shall be able to join or leave at any one time.

Verification of reliability will be done by computers simulating multiple nodes which send messages to each other. If all nodes receiving a message sends out an acknowledge message, which does not require an ACK. of its own, this secondary requirement is considered a success.

Bibliography

[1] Theo Kanter, Patrik Österberg, Jamie Walters, Victor Kardeby, Stefan Forsström and Stefan Pettersson, "The mediasence framework", *In Processings of 4th IARIA International Converence of Digital Telecommunications (ICDT)*. Colmar, France, July 2009.

[2] Theo Kanter, Stefan Forsström, Victor Kardeby, Jamie Walters, Patrik Österberg and Stefan Pettersson, "Chapter 16: Ubiquitous Mobile Awareness from Sensor Networks", *Enabling Context-Aware Web Services Methods, Architectures, and Technologies Edited by Quan Z. Sheng , Jian Yu and Schahram Dustdar*, Pages 449–468. Chapman and Hall/CRC, 2010.

[3] Theo Kanter, Stefan Pettersson, Stefan Forsström, Victor Kardeby, Roger Norlin, Jamie Walters, Patrik Österberg, "Distributed Context Support for Ubiquitous Mobile Awareness Service", *Fourth International Conference on Communications and Networking in China*. ChinaCOM, 2009.

[4] Wikipedia [EN], "SCRUM (Development)", *http://en.wikipedia.org/wiki/Scrum_(development)*, Aug 2011.