Conclusive Notes

Varain

Helsinki December 3, 2006

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Kankaanniemi Marko
Lampila Pekka
Ruottu Toni
Viita Antti

Instructor

Yrjänäinen Sampo

Client

Manner Jukka

Project Masters

Taina Juha

Homepage

`http://www.cs.helsinki.fi/group/varain`

Change Log

Version	Date	Modifications
1.0	1.12.2006	First version

Contents

1 Introduction

This document has been written at the end of software project Varain. It's purpose is to give information about all the products that have been created during this project. The beginning of the document introduces our final product, that is a patch set to KOM RSVP package that adds LRSVP support. After the numerous documents created during this project are listed and their purposes discussed. Additionally there is a section that gives final analysis of the project and another section for the final conclusion.

2 Glossary

RSVP Resource Reservation Protocol. Protocol for reserving resources in a IP network.

LRSVP Local Resource Reservation Protocol. Modification of the RSVP protocol, that provides the means to reserve local resources.

KOM RSVP RSVP implementation made by Multimedia Communications Lab at Technische Universität Darmstadt.

3 Introduction of Project Varain

Varain is a patch set to KOM RSVP [Kar] to add a LRSVP functionality to it. LRSVP add-on is specified in RFC draft [Man06]. Basically it is used to make bandwidth reservations on either end of the route. The reservation can be done on local ends access network or remote ends access network or on both ends. The program has two main executables, one for RSVP daemon and one for LRSVP proxy. There are also some small utilities to test the software or to make manual reservations.

3.1 How LRSVP Works

In the LRSVP scheme the local end host is expected to be RSVP aware and then we add an LRSVP proxy to a router in the local access network. The proxy is located somewhere within the local access network. A good place would be the access network gateway. For local reservations, the proxy acts on behalf of correspondent nodes. Now, in order to distinguish local reservations from end-to-end reservations, we use one bit in the unused Flags field in the RSVP Session Object. The Local Indication (LI) bit is used to differentiate reservations that are internal to the access network. When the bit is set the RSVP message is part of local resource signaling and the RSVP router running the proxy will not forward the message to the next hop but instead give the message to the RSVP application running on the router.

When the local end host wants to make a resource reservation for a downstream flow, it needs a Path message from a node on the data path. If the data sender is not RSVP aware, the local end host can trigger the LRSVP proxy to send the Path message on behalf of the data sender. A new message type called "Path Request" is used to request a Path message from the local RSVP proxy. This message has the same structure as a standard Path message.

A second message called "Path Request Tear" is used to tear down a downstream reservation. Note that due to the new bits and message types, all RSVP routers inside the access network must be upgraded with the LRSVP extension.

When a local end host wants to reserve resources in the local access network, it uses the LI flag in RSVP messages to indicate a local reservation. The structure of the RSVP messages follows the RSVP standard. When the router running the LRSVP proxy receives an RSVP message with the LI bit set it will notice that the flag was set and does not forward the message further to the next hop. The RSVP daemon on the router gives the message to the local RSVP application.

4 Summary of Project Documentation

During our project we produced documents about this project. Our documentation consist following documents: project plan, requirements specification, implementation plan, test plan, implementation description, system administrators guide and conclusive notes. We also created a presentation for other groups how participated this course and a presentation for our client.

4.1 Project Plan

Project plan includes definition of project goals and objectives. It will specify tasks and activities how goals will be achieved. There are listed resources which are needed and time lines for completion of the project.

4.2 Requirements Specification

Requirements specification is based on client meeting where requirements for our project were specified. It also defines priorities for all requirements.

4.3 Implementation Plan

Implementation plan describes the architecture of existing software and how we are going to modify it.

4.4 Test Plan

Test plan describes testing system and test cases. It also describes how to perform test cases and what should be the outcome of them.

4.5 Implementation Description

Implementation description describes what is implemented and how. It also tells differences between the plan and the actual implementation.

4.6 System Administrators Guide

System administrators guide is intended for people how will administer and configure this software. It describes how to use the program itself and how to use most common utilities.

5 Project Analysis

Our project goal was to create a patch set that adds LRSVP functionality to the KOM RSVP package. KOM RSVP is a RSVP implementation written in C++ language and consist over 20000 lines of code. The implementation had almost no documentation and the code didn't carry any comments. Although it compiled and seemed to work as intended most of the time.

We also got 2000 lines code written earlier by Adrien Bauffe for the same purpose. Sadly he wasn't able to get it work. Bauffe made some documentation about his work, but did not either use comments in his code. He wrote some useful test cases that we reused.

We decided to fix Bauffe's implementation, instead of creating a new one. To archive all this we had to do lots of reverse-engineering on the old implementation. We also discovered that the software had some design flaws and hidden bugs. Therefore all modifications had to be made with care so nothing would break.

At the end of the project we successfully ran all nine of Bauffe's test cases in our test lab, which consisted of four IBM laptops.

6 Conclusion

Because of the non-typical topic, we found it very hard to follow good software engineering principles in our project. It was e.g. not possible to write unit-tests, as we were missing specifications that would tell us what code was supposed to do.

Lots of our time went into discussing how to exactly adapt a vanilla software engineering process to fit our project. We realized we could not fix entire KOM RSVP code base and decided to just do the modifications that were absolutely necessary. E.g., a task as simple as trying to make the LRSVP proxy read configuration information from command line instead of a file failed because of some hidden side-effects.

The project met many problems. Both technical and software engineering process related. Most of the time it was very hard to predict how close we were to our goal, if we in fact were going to reach it at all. Uncertainty got many team members frustrated from time to time. The project got done very little compared to the time and effort used, but it succeeded to reach the goals it had been set and the goals were reached in time.

References

Kar Karsten, M., KOM RSVP engine. URL `http://www.kom.tu-darmstadt.de/en/downloads/software/kom-rsvp-engine/`.

Man06 Manner, J. et all, Localized RSVP. URL `http://tools.ietf.org/wg/tsvwg/draft-manner-tsvwg-lrsvp-00.txt`. 2006.