Małgorzata Wielgus*, Przemysław Dąbek**, Roman Janusz***,
Tomasz Kowal****, Wojciech Turek*****

PLATFORMA DO AKTUALIZACJI OPROGRAMOWANIA NA URZĄDZENIACH MOBILNYCH OPARTA NA TECHNOLOGII ERLANG

To pisze się na samym końcu

Słowa kluczowe: *Erlang, aktualizacja oprogramowania, urządzenia przenośne*

ERLANG-BASED SOFTWARE UPDATE PLATFORM FOR MOBILE DEVICES

To be written at the very end

Keywords: *Erlang, software updates, mobile devices*

1. Introduction

Fast development of mobile devices creates new domain of applications for large systems composed of many devices communicating over the internet. Those devices can be far from each other, what makes direct access difficult. Our system was designed to simplify software updates on groups of such devices.

There are many areas of applications including:

- Monitoring - with our system we can perform software updates on cameras in the building. Especially if they are equiped with software for face recognition.
- Management of geographically spread devices.
- Security - updates on sensors used in cars or alarms in homes.
- Robotics - autonomic robots used in factories and in industry.

*AGH University of Science and Technology, Kraków, Poland, malgorza@student.agh.edu.pl
**AGH University of Science and Technology, Kraków, Poland, przemyslaw.dabek@gmail.com
***AGH University of Science and Technology, Kraków, Poland, roman@student.agh.edu.pl
****AGH University of Science and Technology, Kraków, Poland, tomekowal@gmail.com
*****AGH University of Science and Technology, Kraków, Poland, wojciech.turek@agh.edu.pl

The need: management and software updates, use cases General usecase of our system is to perform software updates on multiple devices over unreliable network. There is a need of such updates because of growing amount of remote devices such as sensors, mobile phones or network infrastructure devices.

Existing solutions include Mobile Devices Management systems created by Nokia, Motorola, Oracle and by Open Mobile Alliance.

"The tool described in this paper is a response for all the needs world has... "

2. Requirements and assumptions

- Network infrastructure assumptions
  - Possibility of NAT
  - Slow and unreliable connections
- Need for scalability (up to about 10000 devices)
- Simple, intuitive interface
- Managing groups of devices, batch updates

3. Erlang Technology Overview

3.1. Usability of Erlang Technology in distributed systems

Erlang is a technology and a programming language that mixes functional programming with an approach to easily build heavily parallel and distributed, highly available systems. It achieves these goals using a set of unique features, including:

- Virtual machine implementing message-passing concurrency model with lightweight Erlang processes
- Built-in, language-integrated engine for communication in a distributed environment
- Hot code swapping with a fine control over the software upgrade process. The aim of these is to allow an upgrade to be performed automatically without stopping any services.
- Fault-tolerance features like supervisors.
- Takeover and failover mechanisms for cluster systems.

3.2. General description of Erlang OTP

Erlang OTP (Open Telecom Platform) is an Erlang distribution released by Ericsson in 1998 when the language became open source. OTP is a set of standard erlang libraries and corresponding, well-defined design principles for Erlang developers. OTP defines patterns for basic elements that make up the software as well as the general layout of completed, deployed environment.

OTP principles include:

- Supervisors and supervision trees
  Erlang software can be thought of as a set of lightweight erlang processes communicating with each other. In supervision tree principle, these processes form a tree where the leaves are called workers and are doing the actual job, while other nodes are called supervisors. Each supervisor is responsible for monitoring its children and reacting accordingly when any of them crashes. Supervisors allow to design well-structured and fault-tolerant software.
- Behaviours
  Behaviours are a set of basic design patterns used to build common types of software pieces. Fundamental behaviours are:
  - `gen_server` for implementing simple servers and client-server relation between erlang processes
  - `gen_fsm` for implementing generic finite state machines
  - `gen_event` for implementing event handling subsystems
- Applications and releases
  These patterns define general layout of a self-contained, deployable piece of Erlang software. Applications and releases will be described in the next section as the Software Update Platform deals heavily with them.

3.3. Applications and releases in Erlang

An example of deployable package of software written in Erlang is so-called embedded node. An embedded node is a self-contained, configured Erlang environment along with actual software written in Erlang that can be deployed and run using a simple command. An embedded node contains:

- Erlang Runtime System
- A set of Erlang applications, the actual code
- Configuration for ERTS and applications
- An Erlang release

3.3.1. Erlang applications

An erlang application is an independent piece of software that serves some particular functionality. An application is defined by its name, version, code (set of modules), dependencies (other applications) and other more finegrained settings and attributes. These are all configured in an `.app` file. Every application defines a way of starting it, stopping it and possibly upgrading or downgrading it to another version. It also has its own piece of configuration. A running application is often made up of a single supervision tree.

3.4. Upgrading Erlang software

Problems with using the method: 1. system admins are used to apt; 2. management of many devices is hard

4. APT for Erlang

- Why do this
- How was it done
- Results, effects
- Problems and limitations

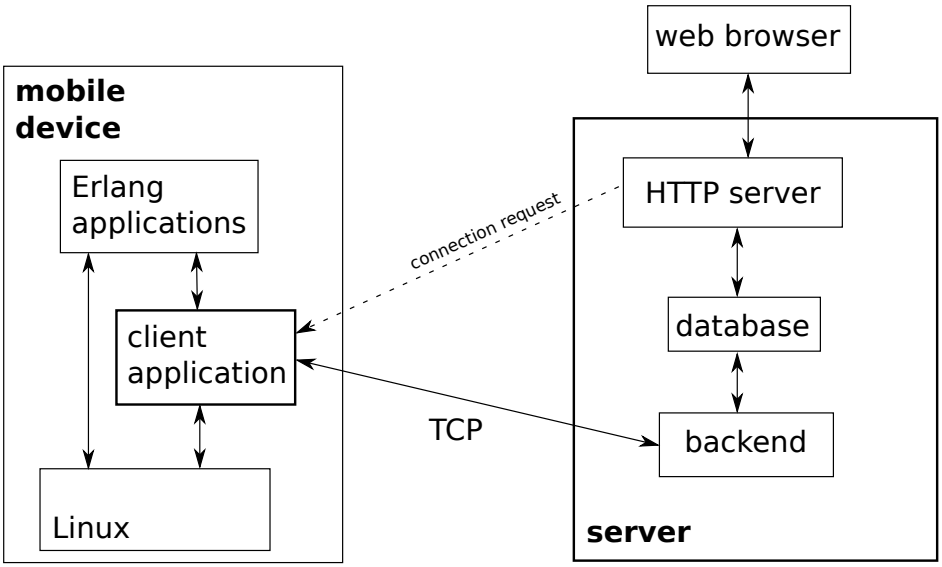
5. Software Update Platform

Software Update Platform is responsible for performing updates on multiple devices and monitoring installed applications. Developers prepare *.deb* packages with applications and add them to repository. When connection with device is established and information about update is written in database, platform will perform software update on that device.

5.1. Architecture

Software Update Platform consists of two main parts:

- client application installed on mobile device,
- server.



Client application

We assume that client application is running on Erlang VM installed on some Linux distribution and *apt* is available. Client application connects with server and perform given operations.

Server

Server consists of 3 components:

- HTTP server
- database
- backend

HTTP server

Mochiweb – our web server of choice is responsible for following tasks. It manages user interaction through web interface (serves content, performs user requests) and it is repository for *.deb* packages.

Database

Mnesia database stores information about:

- device and installed applications on it,
- jobs (e.g. update) for device.

Database is connector beetwen web interface and backend. It stores user requests – jobs to perform on device.

Backend

Backend is the core of platform. It is Erlang application responsible for whole automatic device management. Every new device in platform connected with backend is stored in database. From now on backend can monitor state of that device and performs operations on it.

5.1.1. Device-server session

5.1.2. On-device upgrade logic

5.2. Functionality

- General description of the application: Web interface, technology
- Development model for application, debian packaging utilities
- Direct usage of package manager on remote devices
  gui: 1 or 2 images

6. Conclusions and Further Work

- Support for other package managers
- Communication security
- Development towards more general management platform
  - monitoring
  - configuration
  - maintenance

– diagnostics

7. Acknowledgements

To be added at the very end

Literatura

[1] Palermo D. S., Jenkins J. J.: *Word Association Nouns: Grade School trough College*. 1st ed., Univeristy of Minnesota Press, 1964