CS 211 course project proposal: Remote access to OpenWRT

Zhehao Wang
zhehao@cs.ucla.edu

Haitao Zhang
haitao@cs.ucla.edu

Jeffrey Chen

1. LITERATURE SURVEY

1.1 What is OpenWrt

OpenWrt [**?**, **?**] is a Busybox/Linux based embedded platform which is developed following GPL license. It minimizes its own functions so that it fits for lots of memory constrained devices. Specifically, it builds the appropriate toolchain for devices, compiles appropriate kernel with patched and options, as well as provides software as IPKG packages.

1.2 OpenWrt System Structure

OpenWrt System Structure covers four aspects: directory structure, packages and external repositories, toolchain, and software architecture.

There are four key directories in the base: tools, toolchain, package and target. Tools and toolchain refer to common tools which will be used to build the firmware image, the compiler, and the C library.

In an OpenWrt, almost all the packages are .ipk files. Users can choose what packages to install and what packages to uninstall based on their specific needs. Packages are either part of the main trunk or maintained out of the main trunk. For the second case, packages can be maintained by the package feeds system.

To compile the program for a particular architecture, the toolchain will be automatically created by the OpenWrt system during the cross-compilation process. That will simplify developers tasks. However, when there are needs to create toolchain by hand, OpenWrt also provides an easy way to configure the arguments.

The following figure 1 shows the software stack of OpenWrt. We can see that the common embedded Linux tools such as uClibc, busybox, shell interpreter are used by OpenWrt, so there is no need to learn some complex things.

1.3 How to Develop With OpenWrt

It is easy to port software to OpenWrt. Various fetching methods such as GIT, Subversion, CVS, HTTP, local source can be used to download package source. In a typical package directory, there should always be a `package/<name>/Makefile`. After run "make menuconfig", the new package will automatically show in the menu; and after run "make", the new package will automatically be built.

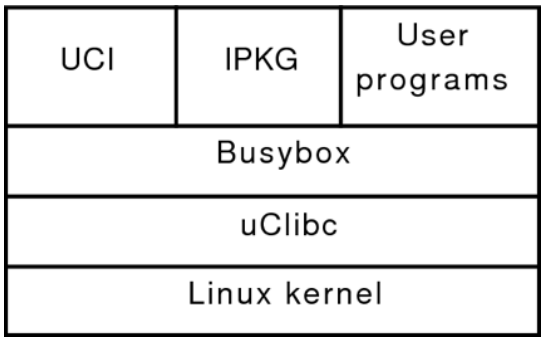

Figure 1: the software stack of OpenWrt

1.4 Web-based Access to OpenWrt

In the OpenWrt system, some important features are provided: a built-in web server with CGI support, an SSH server and a package management tool. We can make use of these existing tools to build our own web-based access application.

Apart from the basic tools, OpenWrt also supports LuCI (https://wiki.openwrt.org/doc/techref/luci), which is really a nice tool to remotely configure the OpenWrt system. Specifically, it provides status visualization functions, system administration functions and network configurations. We can build the web-based access application functions based on their work.

Another useful reference is the netgear (http://support.netgear.com/for_home/default.aspx) mobile application, which designs nice UIs that we can borrow ideas from. We need also to design the web-based access application UIs based on their work.

2. DESIGN AND IMPLEMENTATION ROADMAP

This section covers the functional components of the application, which are organized into three major categories.

- **Network configuration** covers common functionalities in configuration tools that often come with

commercial APs. These configurations include, but may not be limited to, managing network interfaces, DHCP and DNS settings, static routes, and firewall.

- **System configuration** provides interface to customize the OpenWRT box. Common adminstration functions include: system and user configuration (setting device adminstrator password, creating system backup image and restoring system from backup image, generating user SSH keys, etc), software management (installing and configuring software packages), task management (managing scheduled task and startup task)
- **Status/Statistics visualization** offers a mobile-phone friendly view of the system status (Firmware and kernel version, uptime, current time; CPU and memory usage, currently running processes, and system and kernel log) and network-related status (Interface, route, and firewall status, etc). The visualization component could provide real-time graphs of system load and traffic statistics, for example, traffic per interface and traffic per transport layer connection.

The design and implementation effort will be organized by the three function categories, with approximately two weeks dedicated to each.

3. TIMELINE

A rough timeline for the project is given in table 1

Table 1: Project timeline

Week No.	Task
5, 6 (first half)	Implement status/statistics visualization module
6 (second half), 7	Implement network configuration module
8, 9 (first half)	Implement system administration module
9 (second half), 10	Prepare final report and presentation