ULOOP Software Bootstrapping Guidelines

Running ULOOP in wireless Access Points and Android devices

Alfredo Matos and Daniel Romão (Caixa Mágica Software); **Carlos Ballester** (University of Genéve); **Luís Amaral Lopes and Rute Sofia** (SITI, Universidade Lusófona)

The User-centric Wireless Local Loop (ULOOP) project aims at improving the way Wi-Fi networks are used, by providing self-organizing and autonomic aspects to allow these networks to grow on-the-fly, in a robust way. ULOOP has as one of its goals to provide open-source software which when publicly available, must have a user-friendly way to be installed onto user devices.

This whitepaper describes how to bootstrap ULOOP software onto selected hardware and software platforms, so that users can easily obtain and take advantage of the features developed in ULOOP.

Contents

Introduction

The ULOOP project deals with User Centric Wireless Fidelity (Wi-Fi) Networks (UCNs) [5], structures which incorporate heterogeneous end-user devices and/or devices belonging to network operators, which are typically deployed on user premises through devices like gateways. The heterogeneity of such devices can pose a serious challenge when deploying ULOOP software and features, since it can introduce several compatibility and cross-platform issues. In current scenarios, the two most prominent deployment targets for ULOOP are defined by smartphones, devices that the user typically carries, and SoHo APs, which today exist in nearly every household, and are owned either by an Internet Service Provider, by an Wireless ISP (WISP) such as FON, or even by the end-user or its community. Therefore, it is important to consider how to deploy ULOOP software and functionality into such devices, taking into account the (very) different requirements for each platform. In this whitepaper we introduce the two initial proof-of-concept targets[1] for ULOOP software, as specified in the ULOOP architecture [2]: a smartphone operating system, Android[2], and a router-oriented Linux platform, OpenWrt[3]. The ULOOP software consists of four major components (corresponding to the project's building blocks): Trust Management, Resource Management, and Mobility Aspects (which in ULOOP incorporate mobility estimation and mobility anchor point coordination), and an orthogonal block consisting of Integration and Interoperability aspects. The integration of ULOOP software stemming from the technical blocks takes place under the scope of

[1] Even though ULOOP is intended to be cross-platform, it is not feasible to target all major platforms in three years.
[2] http://www.android.com
[3] http://www.openwrt.org

the Interoperability Aspects, which include not only how to integrate the software on the target platforms, but also how to install the software on real devices. The integration is mostly defined by creating a custom OpenWrt firmware that incorporates the different technologies and packages required to run the ULOOP software suite, and a standalone Android application that provides the required features by restoring to the platform's software libraries and development tools.

ULOOP Target Platforms

In Deliverable D2.3 [2], the project has provided specific hardware and software requirements to ensure that albeit only a proof-of-concept would be implemented during ULOOP lifespan, the requirements would fit most of the existing devices today, thus allowing a broad deployment of ULOOP after the project ends. The selected targets were Android and UNIX-like as operating systems for potential end-user devices; OpenWrt as an open operating system for APs. We highlight that, as described in D2.3, ULOOP also intends to release software to other platforms.

ULOOP in End-User Equipment

ULOOP relies heavily on end-user equipment, ranging from notebooks to 3G-enabled smartphones. To assist in a future deployment, ULOOP has set minimum requirements both in terms of hardware and of operating systems to be considered. ULOOP is expected to provide software at least for Android, version 2.x and above; Linux and UNIX systems, where Ubuntu is the default choice, with kernel version 2.36, as well as MacOS X.
The minimum hardware requirements have been set for both smartphones, and laptops. Therefore, when considering smartphones, ULOOP opts for devices that have at least 512 RAM, and at least 16GB storage. The devices must be Wi-Fi enabled (802.11b/g). When considering laptops, these must integrate an Atheros NIC chipset.
For the initial proof-of-concept, ULOOP has first targeted Android, and thus this paper addresses the boot-up procedure in Android.

Android

The Android platform is a customized Linux system with a standardized Software Development Kit (SDK)[4], which enables the creation of mobile applications. The SDK consists of several JAVA classes that run on a modified JAVA Virtual Machine (DALVIK). Accordingly, applications targeting the Android platform should be developed in Java, making use of the provided SDK. For more customized features, it is possible to use the Native Development Kit (NDK)[5], which provides compatibility with C/C++ code. The ULOOP project currently focuses on Android 2.3 for backwards compatibility, but can also run on newer versions. Deploying ULOOP on Android devices implies bootstrapping the software into devices which currently have no Internet connectivity apart from a limited (captive portal) connection to an AP.

ULOOP in Wireless APs

As part of the ULOOP functionality is to go into access points, be it integrated with residential gateways, or isolated, and in order to ensure a wide deployment, the project considers a minimum set of requirements for APs, both in terms of hardware, and of software.

In ULOOP, APs must be EU compliant and support 802.11b/g, 2.4GHz. Ideally, 802.11n should be supported, even though it is not mandatory. The wireless NIC chipset must be Atheros, ideally AR7240 or AR2315. The device must also include an Ethernet interface. In terms of memory, 8Mb of flash ram and 32Mb of SDRAM are the defined requirements for ULOOP. In other words: all of the software for APs in ULOOP is being developed having in mind not only these requirements as the minima, but in fact that these requirements are the characteristics that the majority of residential APs hold. As such, it ensures that despite the fact that ULOOP software shall be addressed as a proof-of-concept, other hardware with the same characteristics will be capable of easily supporting ULOOP.

The operating system for APs considered in ULOOP is OpenWrt, due to its availability, ease of use, and flexibility, as described next. The specific version is OpenWrt Backfire 10.03.1 which is the latest stable release of the OpenWrt Operating System, built on top of a Linux Kernel.

OpenWrt

OpenWrt is a Linux distribution for embedded devices. It aims to free users from the often closed firmware provided by equipment vendors.

OpenWrt provides an SDK[6] for standard application development, and the tools for building a custom firmware image. An OpenWrt AP has the capability to integrate a wide number of software modules which can be obtained via regular mirrors. Creating packages of other applications is also possible using

[4] http://developer.android.com/sdk

[5] http://developer.android.com/tools/sdk/ndk
[6] http://wiki.openwrt.org/doc/howto/obtain.firmware.sdk

the OpenWrt SDK, or using the full source, where all the system can be build.

In ULOOP, OpenWrt is used to develop the so-called ULOOP gateway: an AP/router which has Internet access. Since the ULOOP gateway is assumed to already have a functioning internet connection due to their primary function, it is only necessary to obey the requirements of installing customized software on the devices, in this case a new firmware that contains the ULOOP software modules.

A custom firmware was chosen as the primary distribution mechanism as it is simply installed and can contain all the necessary customization and configurations, which in any other case would be a cumbersome process.

Building a Custom ULOOP Firmware

The ULOOP OpenWrt firmware is built from the official OpenWrt sources using the current stable release, as mentioned before. It incorporates the required modifications, configurations and software blocks. The firmware itself is automatically built by using a build script, specially developed for the ULOOP project. The script automatically activates the required software modules, prepares the necessary configurations and files, incorporating them into a new firmware image, using the base OpenWrt stable source. The added software includes the main ULOOP blocks.
The build script provides a user-friendly way of building a ULOOP firmware image – this is an aspect that we believe is essential for the wide deployment of ULOOP. As both ULOOP software and OpenWrt source are downloaded on runtime from the subversion repositories, newly created firmware images are always updated.

ULOOP Specific Packages

Trust Management (TM): In terms of basic requirements for the boot-up procedure, TM components, which represent the core of ULOOP, simply require the activation of several already existing modules on the access point, such as cryptographic support modules (OpenSSL[7]), besides the standard modules for trust management.

Resource Management (RM): The RM components are part of the hostpad daemon and require that the module mac80211 is installed. In hostapd, the ULOOP resource management part is a self-installable module accessible via user-space. As such, it will be easily installable.

Mobility Aspects (MA): In ULOOP, mobility aspects integrate both mobility estimation, and an entity which coordinates mobility anchor points [6]. Therefore and for the mobility coordination aspects, ULOOP considers as external technology for mobility management Proxy Mobile IPv6 [4]. From an implementation perspective, we have selected OPMIPv6[8] implementation. A major aspect to tackle in the ULOOP image for APs is that for the case of Mobility, we have to consider IPv6 support. This requires a careful integration with the additional tasks, and eventually, to consider globally IPv6 support.

The software integration requirements introduce a few additional packages that are not present on the default OpenWrt firmware. These include enabling support for debug USB/serial connections for a more comprehensive overview of what is happening on the embedded platform, which can be hard to assess using standard debug mechanisms.

Deploying the ULOOP Firmware

In order to use the ULOOP firmware image, the firmware must be deployed on the selected AP (following the hardware requirements described). The firmware deployment can be easily done by using the router's administration web interface (LuCI), where the new firmware can be uploaded, as shown in Figure 1. Once this process is complete, the ULOOP services are automatically started using the service deployment mechanisms, and also, access to the ULOOP application for smartphones is provided through the captive portal, as a part of the wireless configurations provided on the access point image. Both of these mechanisms are described next.

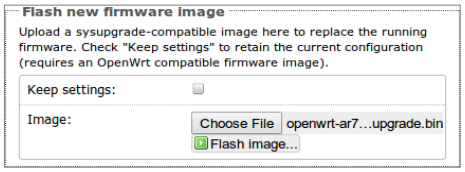

Figure 1 – Simple flash interface on the Access Point.

ULOOP Service Deployment

The different ULOOP services are run as standalone OpenWrt software packages. Therefore, a ULOOP script/daemon is responsible for launching all the necessary software components, in the correct order. Currently, a boot-up bash script launches a binary daemon, which checks for the existence of the ULOOP

[7] http://www.openssl.org

[8] http://helios.av.it.pt/projects/opmip

Comentário [r2]: Reference and not footnote

Comentário [a3R2]: It is a reference to a software website. We can change if needed.

Comentário [r1]: Luis, please confirm if anything is required

required components, launches them and provides the according log information.

Wireless Interface Configuration

An important part of the ULOOP custom OpenWrt platform concerns how the actual wireless interfaces are configured and deployed. For devices that do not have the ULOOP software, a captive portal is considered in the initial boot-up phase of ULOOP, residing such portal in the ULOOP enabled AP. End-user equipment, upon attempts to associate to an ULOOP AP, will therefore be redirected via such captive portal to download the adequate software.

Considering the Android example for end-user equipment, and to allow users to easily obtain the ULOOP Android application, a portal is provided via an open wireless network, where users can download the ULOOP Android application, in order to be able to connect to the ULOOP access points. The captive portal provides a walled-garden that only enables access to the ULOOP Android application. This Android application is accessible by end-user devices. If end-user devices wish to connect to ULOOP-enabled networks, then they must use the provided application. The captive portal was designed in order to provide simple and straightforward access to the ULOOP application, for end-user devices, which do not yet have support for ULOOP required mechanisms. Such distribution is discussed in the following sections.

Dynamic Interface Creation – Virtual Interfaces

As outlined in the ULOOP Specific Configurations, when the gateway initiates the boot up process, it will have an open wireless network, which enables users to obtain the ULOOP Android application. This wireless network is enabled by a virtual interface, created automatically when the gateway starts, that leads towards the captive portal and provides access to the ULOOP Android application.
Once users install the ULOOP application on their equipment, they are ready to negotiate a connection to the ULOOP gateway. Following the regular Wi-Fi process of attachment, the end-user equipment performs negotiation with the gateway, and the TM module in ULOOP processes such request. If authorized in terms of trust association, then the ULOOP gateway, based on the RM, takes care of call admission control and of resource allocation to accept or deny the attachment request.
Provided that a request can be accepted, resource allocation in ULOOP requires the use of virtual interfaces, one per station (end-user device), as ULOOP provides an innovative way of performing multi-user resource allocation, where the spectrum is optimized in a flexible way.
Virtual interfaces are required, as different stations will have different wireless configuration to support the notion of "Elastic Spectrum Management" in ULOOP.
The virtual interfaces creation process is achieved through a script that instantiates a virtual interface and configures a MAC address derived from the real MAC address of the device. It also provides all the necessary wireless configurations (e.g. SSID, WMM, etc) for the new interface and, at the end of the process, calls *hostapd*[9] to initiate the wireless network. *hostapd* is a daemon located in the userspace layer of the mac80211[10] architecture, which enables the access point functionality on Linux devices, by handling authorizations, associations and other aspects necessary for the correct operation of an access point. Each virtual interface created by the script operates independently of other virtual interfaces. Therefore, it is possible to disable a wireless network and delete the respective interface without interfering with ongoing communications or wireless networks that exist on the device. The same happens with the creation of a new wireless network. The number of virtual interfaces depends on resources available on the device, and is managed by the Resource Management modules in the ULOOP architecture.

Boot-Up Procedure in Android

As discussed, the main goal for deploying a captive portal in ULOOP gateways is to provide access to end-user devices to download ULOOP software. In our case, this is presented in Figure 2. When a user is connected to the open wireless network, an IP address from a different network is assigned, thus being possible to enable redirection to the captive portal from all the devices connected to that network.

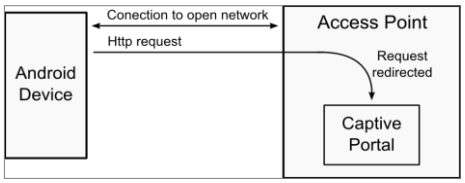

Figure 2 – HTTP redirection towards the captive portal.

When trying to connect to a remote site, the user will be redirected to the present captive portal, where it is possible to download the end-user application, in this

[9] http://hostap.epitest.fi/hostapd/
[10] http://wireless.kernel.org/en/developers/Documentation/mac80211

case the ULOOP Android application, ensuring that the user has all the components required for participating in a ULOOP-enabled network.

Android Application Overview

The ULOOP Android App implements the basic needed functionality in order to join ULOOP as a new user, to add a new node/device to ULOOP from an already existing user and to obtain Internet connectivity through a ULOOP gateway.
Once the ULOOP app is launched for the first time, it displays several "first run" screens on which the user can select how to join ULOOP with the device:

- an existing user with a validated nickname;
- an existing user with another node;
- a new user who owns no nodes in ULOOP.

In the first case, as can be seen on Figure 4 (left), the user has to provide her/his validated nickname and password in order for the Identity Validator provider (in this case ULOOP partner Level7) to verify that she/he is the owner of the validated nickname. Once this is validated, the secrets are transferred to this new device and the user can proceed to setup the dispositional trust of the new device.

In the second case, as depicted in Figure 4 (right), the user can retrieve the secrets from another node she/he owns and which is in the proximity at that time. In order to do this, the user needs to input the crypto id of her/his other node (either manually or by flashing a QR Code with the app's built in functionality) and her/his password. After this the user can proceed to setup the dispositional trust of the new device.

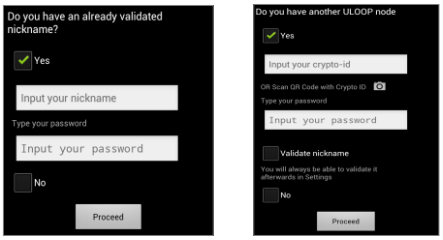

Figure 4 - User with an already validated nickname (left), or user which owns another node (right).

In a last case illustrated in Figure 5, the user enrols in ULOOP as a new user without previous owned nodes or validated nickname. The user is the prompted to choose a nickname, a password and also given the possibility to validate the chosen nickname with one of the available Identity Validator providers. After performing these steps, the user is requested to setup the dispositional trust for the device.

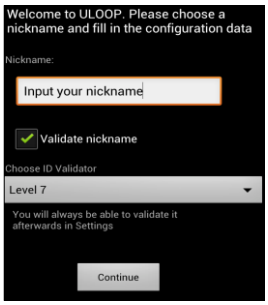

Figure 3 – New ULOOP user.

Once the user has finished configuring the ULOOP app using the first setup, which allow customizing the ULOOP experience (changing the dispositional trust or validating the nick-name if it has not been done before) through a settings menu, access basic account information such as device model, Crypto id and device id (which is the concatenation of the crypto id and the mac of the device), or use one of the main options, namely, to request Internet access through a nearby ULOOP gateway or to flash a QR code/create a QR Code from crypto id for joining/flashing purposes. Also, a foreground service is started in order to start running all the periodic activities of the requester such as evaluating cooperation, managing monetization or processing service requests/replies as described in [7].

Comentário [r4]: Crediting aspects missing and are the final part of the setup. Reference to the paper of trust management by UNIGE must be provided.

Summary

ULOOP provides many features that impact the way Wi-Fi networks are created and managed. Therefore, it becomes important that the software bootstrap problems are clearly addresses, in order to provide a simple deployment process. For the current focus platforms, OpenWrt and Android, we described how such software bootstrap can be achieved, with minimal user interaction. In this whitepaper we presented how ULOOP is tackling the problem of getting software onto devices that can later be used in the ULOOP pilots and demonstrations. This implies distributing the router image as a pre-configured firmware, ready to be installed on the device, and an android application, that is installed on mobile devices. The defined process to build and deploy the ULOOP software will continue to be improved, in order to accommodate the requirements imposed by the different software packages. The ongoing efforts are already shifting towards the interoperation between several platforms, considering the interaction of components on the access points (developed in C/C++) and on the smartphones (developed in Java).

References

[1] EU IST FP7 ULOOP – User-centric Wireless Local Loop. Gr. Nr. 257418, 2010-2013. http://uloop.eu.

[2] Sofia, Rute C. (Ed.), ULOOP D2.3: Overall Specification , *ULOOP Consortium*, September 2011

[3] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[4] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, August 2008.

[5] Rute Sofia, Paulo Mendes, "User-provided Networks: Consumer as Provider", IEEE Communication Magazine, Feature Topic on Consumer Communications and Networking - Gaming and Entertainment, Vol 46, # 12, pp. 86-91, December, 2008.

[6] Qing Zhou (Ed.), Mobility Coordination Aspects in ULOOP , *ULOOP Consortium*, October 2012

[7] Carlos Ballester (Ed.), Trust Management in ULOOP , *ULOOP Consortium*, October 2012

The research leading to these results has received funding from the EU IST Seventh Framework Programme (FP7/2007-2013) under grant agreement n 25741, project ULOOP (User-centric Wireless Local Loop), participants: Alcatel-Lucent Bell Labs, (FR), COFAC/University Lusófona (PT), Huawei Technologies Duesseldorf GmbH (DE), ARIA S.p.A (IT), Caixa Mágica Software (PT), FON Wireless Ltd (UK), Technische Universität Berlin (DE), University of Kent (UK), Université de Genève (CH), Level7 srlu (IT), University of Urbino (IT).

For further information please visit http://www.uloop.eu/, join the **FP7 ULOOP Project** group on LinkedIn, follow **@uloopproject** and look for **#uloopproject** on Twitter.