Implementing Zeroconf in Linphone

Abhishek Srivastava,[*] Jae Woo Lee,[†] and Henning Schulzrinne[‡]
Computer Science Department, Columbia University

(Dated: January 12, 2011)

ABSTRACT
This report describes the motivation behind implementing Zeroconf in a open source SIP phone(Linphone) [1] and the architecture of the solution implemented. It also describes the roadblocks encountered and how they were tackled in the implementation. It concludes with a few mentions about future enhancements that may be implemented on a later date.

I. INTRODUCTION

Linphone [1] is a freely available open source SIP phone that has been ported to a number of different platforms like Linux, Windows, iPhone and more recently, even Android. It has an easily extendible architecture that allows developers to build on the existing low level libraries to easily add new features and improvements.

One such feature that has been implemented in a number of IM clients is the ability of the agent to discover other agents like itself that are present on the local network. Consider a small office or organization of people wanting to be able to communicate with each other without having to constantly configure and add friends/users into their contact lists to establish short duration conversations over voice or IM.

The Apple Bonjour protocol [3], also known as the Zeroconf protocol, provides the APIs for applications to implement service discovery and service publishing functionalities.Bonjour locates devices such as printers, other computers, and the services that those devices offer on a local network using multicast Domain Name System service records. A number of applications like ITunes, XMMS and Ekiga now implement some form of service discovery mechanisms to take advantage of the enhancement it brings to user experience.

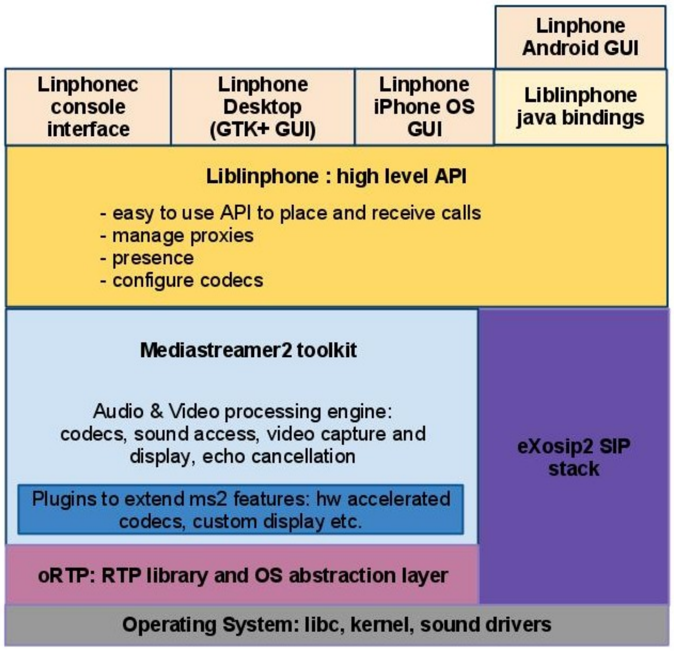
Lee et all in their IETF draft [2] proposed using Zeroconf for SIP URI discovery using DNS Service Discovery. This would allow SIP agents in a local network to discover each other without the presence of a SIP Registrar. Setting up a SIP Registrar in a small adhoc network may not be entirely feasible; and this is the perfect scenario for the solution proposed in the draft. Linphone is one such SIP agent that was chosen as an appropriate candidate for implementing the prototype of this solution.

There are alternatives to using DNS-SD for establishing adhoc SIP calls between agents in a local network. SIP Multicast sends SIP INVITEs to other agents using multicast groups with destination IP address (224.0.1.75). To be able to discover each other though, they would have to send multicast REGISTER requests and maintain databases of peers whose REGISTERs they have received. Unlike DNS-SD, newly arriving UAs will not discover other UAs until they refresh their REGISTERs, thus providing only passive discovery. There could be a significant amount of delay introduced due to this in high-churn networks like adhoc wireless networks.

II. MOTIVATION FOR CHOOSING LINPHONE

Linphone is a SIP client that provides functionalities for placing SIP voice calls when configured with an account in a SIP Registrar. It provides a minimalistic GUI but has an extensive command-line based UI. Due to its layered architecture, it has been ported to a number of platforms with ease. Figure 1 shows the architecture of the Linphone source code.

FIG. 1. Layered Architecture of Linphone Code


Linphone Android GUI
Linphonec console interface
Linphone Desktop (GTK+ GUI)
Linphone iPhone OS GUI
Liblinphone java bindings
Liblinphone : high level API
- easy to use API to place and receive calls
- manage proxies
- presence
- configure codecs
Mediastreamer2 toolkit
Audio & Video processing engine: codecs, sound access, video capture and display, echo cancellation
Plugins to extend ms2 features: hw accelerated codecs, custom display etc.
eXosip2 SIP stack
oRTP: RTP library and OS abstraction layer
Operating System: libc, kernel, sound drivers

[*] aas2234@columbia.edu
[†] jae@cs.columbia.edu
[‡] hgs@cs.columbia.edu

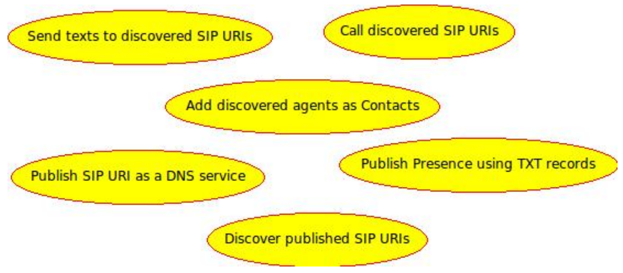
Previously, the prototype that implemented Zeroconf for SIP Communicator used Bonjour APIs and was implemented in Java. This project aimed to replicate a similar functionality in Linphone for the Linux platform. The Avahi APIs [5] are a perfect fit for an application based in C on Linux, like Linphone. It provides high-level C APIs for implementing service browsing and service publishing and also provides methods for turning these into threads that can be started and terminated asynchronously from another controlling thread.

Linphone is built over a high level library called Liblinphone that integrates all the SIP video calling features into a single easy to use API. The core functionalities of liblinphone are call initiation, termination, acceptation, management of proxy configurations and registrations, addressbook, presence, call logs and persistence of configuration data and contact lists. Currently, this library has been ported to Linux $(x86, x86\_64, ARM, blackfin)$, Windows $(XP, Vista, 7)$, MacOS X (audio only) and Google Android (audio only).Liblinphone brings together the 2 aspects of VOIP communication : media and signaling. It uses the exoSIP2 stack for SIP signaling and mediastreamer2 for voice/video streaming.

III. ARCHITECTURE OF IMPLEMENTATION

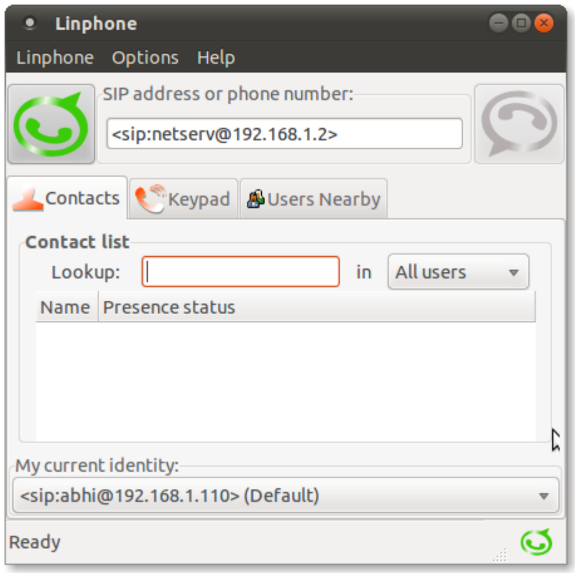
Initially, the aim of the prototype solution was to be able to allow discovery of SIP URIs on the local network. Having achieved this, the goal was then to make it possible to call these SIP URIs, send messages to them and add them as contacts. A complete list of the use cases that were envisioned are documented in Figure 2.

FIG. 2. Use Cases for Zeroconf in Linphone


Send texts to discovered SIP URIs
Call discovered SIP URIs
Add discovered agents as Contacts
Publish SIP URI as a DNS service
Publish Presence using TXT records
Discover published SIP URIs

The Avahi daemon acts on behalf of an application to advertise the service, discover new services and manage the traffic shaping of the DNS queries by caching DNS records. Applications are exposed to the avahi-common API that is built over the Avahi daemon's interfaces. The inter-process communication between the application and the daemon is done using the D-Bus messaging system [6]. As an application developer wanting to use mDNS-SD [4], one has to write the callback functions that will be called when a new service is discovered, a server state change occurs, etc. The avahi-common API is essentially event-based from the viewpoint of an application developer, although underneath it polls for Avahi daemon state changes.

FIG. 3. Linphone's Linux GUI



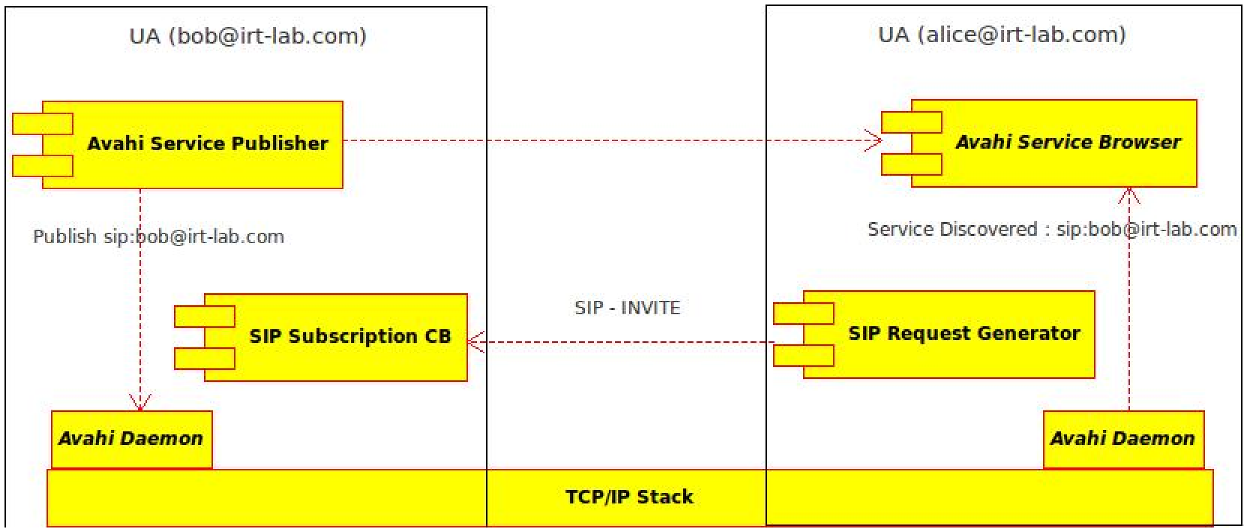
Instead of trying to integrate the solution into Linphone right from the start, the approach taken was to first create and test the Avahi service browsing and publishing modules in a standalone mode. These modules were implemented as separate threads (pthreads) and were then integrated into Liblinphone. The threads were triggered with a configuration option in the Linphone configuration menu. A separate tab (separate from the usual contacts) was added to display "zeroconf contacts" as was suggested in [2]. A simplistic component diagram in Figure 4 indicates how these modules interact when a SIP call is to be placed.

UAs discovered via mDNS-SD would need to be presented appropriately to the user in a GUI. The Linphone GUI in Linux is built with GTK widgets [7] and it was fairly simple to add a GTK tab ("Users Nearby") (Figure 3) to display the zeroconf users discovered in the .local domain. Users discovered through zeroconf would be like transient contacts that the user can interact with by calling or messaging. The GUI also provides the means to add a transient contact as a permanent contact. Doing so may be beneficial if the contact's zeroconf broadcast is switched off and the user wishes to call him/her. Offline users are not displayed in zeroconf and as opposed to Bonjour, the disappearance of a contact on the network is almost instantaneous.

IV. CHALLENGES FACED

The challenges faced while implementing this project were mainly due to inadequate documentation of the Avahi libraries, synchronizing data structures between threads and integration issues with Linphone's large code

FIG. 4. Component Diagram of Solution



base. Since the Avahi service browser and publisher were implemented as independent threads, they needed to be managed from the controlling Linphone g_main() thread. As it turns out, in the world of pthreads, one thread cannot directly terminate another thread. The suggested method is to have a shared mutex that one thread may write to and the other keeps reading from to see if it has to terminate. More recently, the Avahi community has come up with the AvahiThreadedPoll object that simplifies the integration of Avahi into multithreaded applications. The polling is run in a different thread and there are functions that can be called to start/stop this polling object.

Since the UI widgets and handlers were written in GTK, the appropriate handlers had to be integrated so as to display the zeroconf users list and their presence. There was a steep learning curve in understanding the GTK event loops, callback functions from the widgets and the GTK tags that were required for designing the UI.

Integration with a large pre-existing code base also was quite challenging. There were a number of execution flows in Linphone; some were triggered from the UI and some were triggered by externally arriving SIP INVITE requests. It was important to make sure that any changes that were made to add zeroconf did not break any of these flows. A large amount of effort went into testing all the features after integration of zeroconf with Linphone.

V. CONCLUSIONS AND FUTURE WORK

As more and more applications move towards adopting Zeroconf as a de-facto standard of discovering peers and publishing their own services, it is becoming increasingly important to have a well-documented and easy to implement API for mDNS-SD. The Avahi API designers have achieved this distinction to a large extent, and today, there are a large number of applications that use Avahi as their mDNS-SD implementation.

The patch created as a deliverable in this project has been sent to the Linphone developers so that it may be integrated with the mainstream Linphone project.For brevity, the patch has not been included in this report but the code is available as a public git repository hosted on Gitorious (gitorious.org/linphone-with-avahi-zeroconf).

Future enhancements to Zeroconf on Linphone may include parsing the TXT record for obtaining more information about a contact like presence information, location details and status messages. It is debatable whether one would want to broadcast such information in the TXT record because it would violate privacy and make it easy for miscreants to obtain sensitive details about the contact. In the context of a small office or home network, it should be reasonable to assume that such details may be broadcast without causing much harm, with the knowledge and permission of the user.

mDNS-SD may also be used to obtain configuration data for a SIP UA like SIP Proxy and Registrar servers and port numbers. The SIP Proxy servers in a particular domain may advertise themselves as PTR records and send across more detailed information in the TXT record when resolved for by a SIP UA. This would cater to a larger domain of SIP UAs that would like to call other UAs registered on different SIP Registrars. These SIP Registrars may be in the .local domain or a wide area domain; generally, one would call other SIP UAs through a Registrar only if they are on a different network domain. mDNS-SD can be done on the wide area domain to resolve such SIP Registrars.

[1] Linphone : An Open Source SIP Phone, *www.linphone.org*

[2] Lee, Schulzrinne, Kellerer, Despotovic, *SIP URI Service Discovery using DNS-SD* IETF, Network Working Group Internet-Draft

[3] Bonjour : Implementation of Zeroconf Service Discovery Protocol, *http://www.apple.com/support/bonjour/*

[4] Cheshire, Krochmal, *DNS-Based Service Discovery* IETF, Network Working Group Internet-Draft

[5] Avahi : Service Discovery API *www.avahi.org*

[6] D-Bus : Message Bus System for IPC *http://www.freedesktop.org/wiki/Software/dbus*

[7] GTK : Library for Building GUIs for X Window System *www.gtk.org*