Distributed simulation platform based on Android VMs

Radu Paşea
Faculty of Automatic Control and Computers
University POLITEHNICA of Bucharest
Email: radu.pasea@gmail.com

Abstract—**The HYCCUPS framework relies on opportunistic interactions to create a sort of ad-hoc mobile cloud which is used to migrate computational tasks between devices with the goal of improving performance and reducing energy consumption. Because the fact that a large number of people and devices need to be coordinated in order to test various utilization scenarios for HYCCUPS, testing the framework is very difficult.**

To this end, a distributed simulation platform was proposed which would offer HYCCUPS an easily configurable real-time simulation environment. In this paper, the project is continued with the proposals for a networking setup able to simulate opportunistic interactions and Android device simulators based on Android x86 virtual machines.

I. Introduction

This paper is the continuation of [1] which presents the design of a real time simulation platform for the HYCCUPS framework. The HYCCUPS project [2] proposes an opportunistic cloud with the goal of extending the battery life of individual devices. This cloud is formed through opportunistic interactions (such as sharing a wireless access point [3]) of task sharing Android devices. Tasks can be offloaded to other devices, with a preference for ones that are not impacted by energy consumption (such as devices plugged to an electrical outlet).

In order to test and evaluate HYCCUPS as accurately as possible, a real time distributed simulation platform has been proposed in [1] which would simulate real world opportunistic and application usage scenarios based on recorded traces or on event models. The main challenges for the success of the project are the implementation of an Android device simulator and a mechanism for simulating opportunistic networks without modifying the Android system or HYCCUPS itself. These two challenges, however, are linked because the opportunistic network mechanism must use the tools provided by the Android simulator.

In this paper, the practical approach to overcome these two challenges will be presented. In section II the modeling of the opportunistic simulation will tackled, by first describing the networking layer of HYCCUPS and then presenting a tested solution for simulating opportunistic interactions in it, and section III will describe the challenge of simulating Android devices and present the pursued solutions with their drawbacks and its current status. Finally, the conclusion and planned future work will be presented in section IV.

II. Opportunistic Network Simulation

The opportunistic network modeling component represents the core of the simulation platform, opportunistic interactions being the fundamental concept on which HYCCUPS is based. The goal is to perform these interactions at the platform layer without modifying the higher layers in any way, thus providing an environment as close to real world scenarios as possible. The communication framework on which HYCUPPS relies is AllJoyn, an open source project which enables applications to "communicate over various transport layers, such as Wi-Fi, power line or Ethernet, regardless of manufacturer or operating system and without the need for Internet access."[4]

It is intuitive that we start the section by detailing how AllJoyn works and then we go on to show how to simulate virtual LANs using it.

A. AllJoyn

The AllJoyn framework is designed to run on the local network and it was built to be transport-agnostic, currently supporting both Ethernet and Wi-Fi, the mediums which are of interest for our project. The framework provides and internal AllJoyn router to the apps which is built at the OSI Application layer and provides the AllJoyn enabled apps a means for discovering and communicating with each other. The option for not using an internal router is provided, so some devices can use a lightweight client which can use an AllJoyn router present on another device in the LAN, but this is of no interest for us because HYCCUPS comes bundled with a full implementation. Also, AllJoyn defines both service providers and consumers, but HYCCUPS clients are identical so this differentiation will not be used in the paper.

AllJoyn establishes networks between sets of peer devices which can use different transport mediums. A high level view of one such network is presented in figure 1. The system allows devices to announce and discover each other in the network using a mechanism based on IP multicast messages. The devices register themselves on the 224.0.0.113 multicast group, where they can send and receive special announcement and discovery messages. This is the foundation on which the opportunistic network simulator will be built, because routing multicast messages can be configured in Linux and, using this, a local network can be partitioned into a number of smaller networks without inter multicast communication.

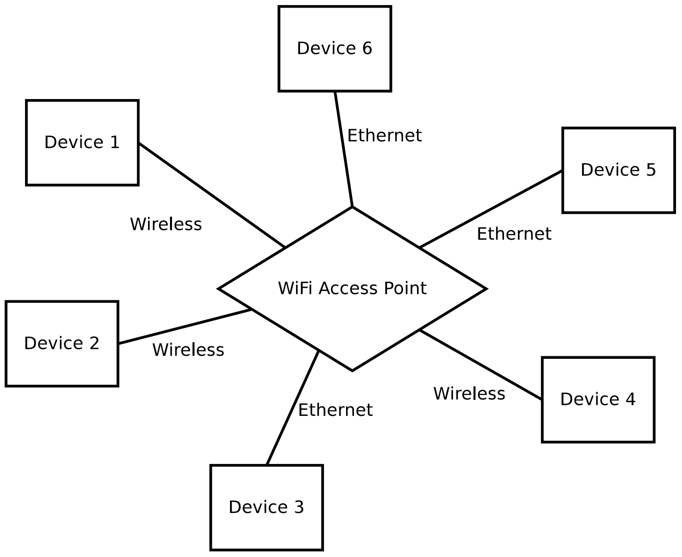

Fig. 1. An AllJoyn standalone network which bridges multiple transport mediums.

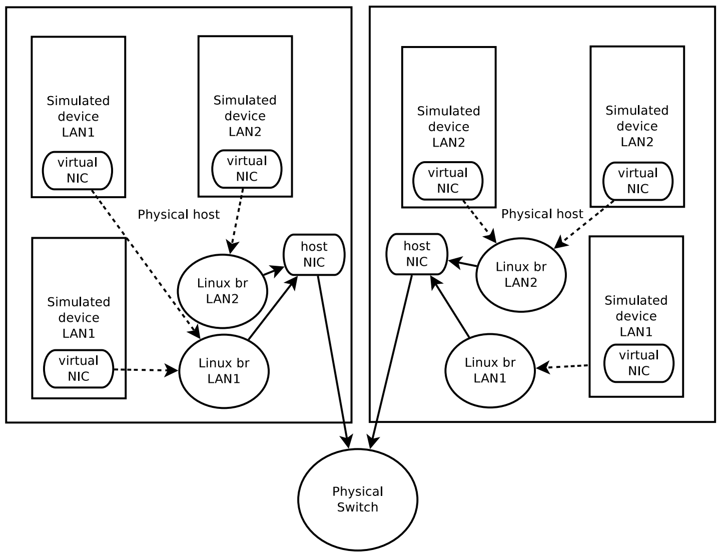

Fig. 2. The networking model for the simulation platform. Two physical hosts containing multiple simulated devices which are connected to two different sub networks.

B. Simulating Opportunistic Networks

The objective for the simulation platform is to seamlessly provide an opportunistic network model by only manipulating the platform's networking properties and not modifying HYC-CUPS or any other application. The proposed solution is based on the fact that AllJoyn is built on top of IP multicast, which can be configured at OS level. The first step is to simulate separate LANs.

The simulation platform is assumed to reside on computers connected in a network with an address space reserved for the interfaces of the various simulated devices. This address space can be then partitioned into multiple subspaces, each of the simulated LANs being allocated one such subspace. The simulated interfaces must provide a network interface which can be manipulated by simulator. These interfaces will be given addresses corresponding to the LANs to which the devices have to be connected at a specific time during the simulation. The computer which hosts the device has a Linux bridge associated with the LAN and to which all the hosted devices connected to that LAN are attached.

Of course, bridges corresponding to the same LAN can be found on multiple physical hosts and its the platform's job to set up routes between them and to ensure that the simulated devices can communicate through multicast messages. The platform must correctly manage and assign IP addresses by taking account of the physical machines, the bridges and the simulated devices addresses. This means that during the simulation there have to be no DHCP servers operating in the network. The network setup illustrated in figure 2. Note that the physical hosts must route message to and from the Linux bridges.

Using this setup, the HYCCUPS tracer was able to detect an interaction between a simulated device and a real device present in the same physical network (by bridging the simulated device's network interface with the actual physical interface of the host). After the interaction, the simulated device was assigned in real time to a sub network and no more interactions were detected. This experiment suggests that the networking setup is valid and that it can simulate opportunistic interactions.

III. Device Simulation

With the networking model in place, the next challenge is to simulate Android devices while achieving a number of conditions. First of all, a device simulator should be able to run Android or a close enough system so that HYCCUPS functions properly and can do its job. Second, the device should offer a virtual network interface through which all its network traffic is directed and which can be controlled from the host by attaching it to different bridges and assigning it different IP addresses at runtime. And finally, the host should have a way through which to run commands in the simulated device for managing its internal network configuration and for reacting to simulation events such as application launches. Two potential solutions were tested: LXC containers running Android emulators and Android x86 virtual machines. Both approaches will be described in this section.

A. LXC and Android Emulators

LXC containers offer ligthweight operating system virtualization, allowing each container to appear to be running its own operating system, but the containers actually share the same kernel. Isolation is achieved through Linux mechanisms such as cgroups and namespaces, ensuring that each container has its own filesystem and network stack. [6]

LXC containers are an appealing option for simulating android devices because they are ligthweight and supported by a wide range of Linux distributions. Containers can easily be controlled from the host and an API is provided for both C and Python, allowing for easy scripting and automatization. Regarding networking, multiple connection modes are offered, including a bridged mode which exports the container's virtual interface in the host, which can then be bridged with a host interface. We can see that conditions two and three are achieved by LXC containers. It only remains for a way of running an Android environment inside of a container to be found.

Although it is theoretically possible to directly run Android inside an LXC container, all the Android's required modules and libraries must be installed on the host (since the container shares the host's kernel, Android must be able to run on the host) and the configuration is tedious and not easily portable. A more convenient solution is to run an Android emulator inside the container.

The Android emulator is provided to help test Android applications without an actual Android device. It works by emulating through software ARM instructions for the host device, reason for which it offers very low performance, and it runs a modified, sandboxed version of Android, making enabling AllJoyn on it very hard if not impossible. Because of this, the option of using the combination of LXC containers with Android emulators has proven to be unfeasible to achieve the goals of the simulation platform.

B. Android x86 VMs

A project aimed at bringing Android to x86 netbooks [7] offers the possibility of running Android virtual machines on the simulation platform, thus offering great portability and good performance, since no emulation is performed. Because the Android x86 project offers install images, Android can be brought to any Virtual Machine Manager that can run Linux machines. In [8], Oracles's Virtualbox is shown to slightly outperform VMware in network performance, so the former was chosen for the simulation platform.

Oracle's VirtualBox is an open source virtualization package which achieves virtualization by scanning the guest OS code at its first run and replacing sensible instructions with traps to the VMM [8]. More features, such as shared storage and a guest control framework, are added by installing the GueastAdditions package inside the guest (an example of paravirtualization). VirtualBox also offers a bridged networking mode which behaves similarly to the LXC one, allowing the simulation platform to control the network configuration of the guest.

The successful scenario of running HYCCUPS in a simulated device and detecting an opportunistic interaction with a physical Android device was performed using a VirtualBox virtual machine and the networking setup described in section II. There is, however, a setback, because the GuestAdditions package isn't supported on Android so VirtualBox itself doesn't provide any way of controlling the guest from the host, so another way has to be found (for example, through the Android Debug Bridge). In spite of this, the Android x86 Virtual Machine solution shows a lot of promise.

IV. Conclusion and Future Work

In this paper, parts of the distributed simulation platform project from [1] are presented, specifically the proposals of a networking setup to simulate opportunistic interactions and a mechanism of simulating Android devices running HYCCUPS. The proposed network setup was tested by partitioning a larger LAN into two smaller networks, using two containers and moving them at runtime between the two networks, thus verifying that the networks are correctly detected at container level.

For testing the Android device simulator, a combination of LXC containers and Android emulators was first attempted to be used, but the limitations of the Android emulator's network capabilities proved the solution to be unfeasible. A second solution consisting in an Android x86 virtual machine running on the VirtualBox virtual machine manager was tested on the networking setup and an opportunistic interaction was observed between the virtual machine and a physical device, both running HYCCUPS and residing in the same LAN. After moving the virtual machine in a sub network, the two devices stopped interacting suggesting the experiment was successful and that the proposed networking setup can be used to simulate opportunistic networks.

A limitation which has to be overcome was found in VirtualBox which stops the host from running commands inside the guest, namely the fact that the GuestAdditions package isn't compatible with the Android x86 virtual machine. This means that commands have to be run directly in the virtual machine's shell, thus disabling the capabilities for automatization and scripting needed by the simulation platform. A workaround for this limitation is the first challenge that needs to be tackled during the next development period.

After a solution will have been found, all of the described building blocks have to be put together, tuned and automated in order to enable the simulating platform to function on a single host. Of course, the next and final step is to develop the distributed system needed to expand the simulation platform to multiple hosts and deploy it.

References

[1] R. Paşea, *Distributed simulation platform based on Android emulators*, University POLITEHNICA of Bucharest, 2014.

[2] Radu-Corneliu Marin and Ciprian Dobre, *Reaching for the clouds: contextually enhancing smartphones for energy efciency*, Proceedings of the 2nd ACM workshop on High performance mobile opportunistic systems, pages 31 to 38, 2013.

[3] Luciana Pelusi, Andrea Passarella, and Marco Cont, *Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks*, IEEE Communications Magazine, pages 134 to 141, 2006.

[4] https://allseenalliance.org, *Alljoyn - proximity based peer-to-peer technology*, Accessed: 18.01.2015

[5] https://linuxcontainers.org, *LXC - Linux containers*, Accessed: 21.01.2015

[6] Mathijs Jeroen Scheepers, *Virtualization and Containerization of Application Infrastructure: A Comparison*, University of Twente, 2014

[7] http://www.android-x86.org/, *Android-x86 Project - Run Android on Your PC (Android-x86 - Porting Android to x86)*, Accessed: 21.01.2015

[8] Damodaran, Deepak K., et al., *Performance Evaluation of VMware and VirtualBox.*, International Proceedings of Computer Science and Information Technology 29, 2012.