PePiT: Opportunistic Dissemination of Large Contents on Android Mobile Devices

Matteo Sammarco*, Nadjet Belblidia*, Yoann Lopez◇, Marcelo Dias de Amorim*, Luis Henrique M. K. Costa†, and Jérémie Leguay◇

* CNRS and UPMC Sorbonne Universités ◇ Thales Communications & Security † Universidade Federal do Rio de Janeiro

1. CONTEXT AND MOTIVATION

Enabling content sharing among mobile users is a promising application for opportunistic networks. Clearly, collocated people are likely to share mutual interests. In this context, disseminating contents through opportunistic communications could be more efficient than passing through central servers.

We implemented PACS (Prevalence-Aware Content Spreading), a popularity-based strategy to select pieces of contents to be exchanged between neighbors solely based on localized information [1]. Through their successive contacts, devices keep track of the dissemination level of the pieces throughout the network and use this information to transfer less prevalent pieces first.

In this paper, we present PePiT, an Android application based on PACS. PePiT enables the dissemination of pictures between collocated Android devices in an ad hoc mode. We show both the architecture and the deployment requirement of PePiT on Android devices. We also briefly describe the demonstration scenario.

2. PACS: PREVALENCE-AWARE CONTENT SPREADING

The goals of PACS are to achieve fast content dissemination while keeping the overhead low and making better use of contact opportunities [1]. To this end, nodes must have a clue on the dissemination progress of each piece of contents, so that they can appropriately prioritize their transmissions.

Let $\mathbf{N} = \{n_0, n_1, \ldots, n_{N-1}\}$ be the set of N nodes in the network. Nodes are mobile, but we do not assume any a priori knowledge of mobility patterns. To illustrate the workings of the algorithm, we assume that all nodes in the network are interested in the single content initially available at a single node called the *data source*. To generalize to any number of data sources and contents, we simply apply the algorithm to a randomly selected content.

The data source chops the content into K pieces of equal size. Pieces are sequentially identified as $c_0, c_1, \ldots, c_{K-1}$.

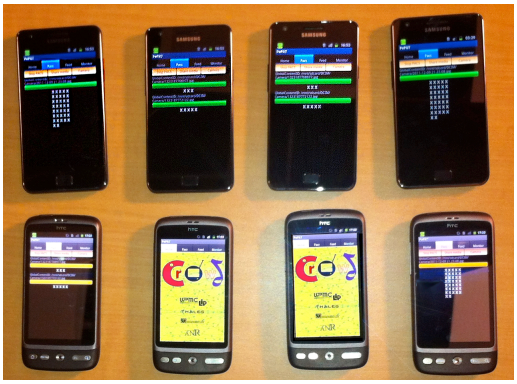

Figure 1: Mobile devices running PePiT.

Nodes use their contact opportunities to get pieces (i.e., we assume that there is no infrastructure to help the dissemination process). Nodes can get pieces from the data source and from any other node in the network having it. Each node n_i stores locally an *availability bitmap vector* $\mathbf{a}_{n_i} = \{a_0, \ldots, a_{K-1}\}$, where $a_k = 1$ if the node has piece c_k, and $a_k = 0$ otherwise. In addition to the availability vector, node n_i also keeps a prevalence vector $\mathbf{p}_{n_i} = \{p_0, p_1, \ldots, p_{K-1}\}$, which gives a local view of the prevalent pieces in the network. Initially, all nodes have an empty prevalence vector.

When nodes n_i and n_j meet, they exchange their availability vectors $\mathbf{a}_{n_i}$ and $\mathbf{a}_{n_j}$. Node n_i (resp. n_j) computes $\mathbf{a}_{n_i} \wedge (\neg \mathbf{a}_{n_j})$ (resp. $\mathbf{a}_{n_j} \wedge (\neg \mathbf{a}_{n_i})$), which gives the candidate pieces to be transferred ($\wedge$ stands for the "AND" operator and $\neg$ is "NOT'). They also update their prevalence vectors respectively as: $\mathbf{p}_{n_i} \leftarrow \mathbf{p}_{n_i} + \mathbf{a}_{n_j}$, and $\mathbf{p}_{n_j} \leftarrow \mathbf{p}_{n_j} + \mathbf{a}_{n_i}$.

Among the candidate pieces to be transferred, nodes select the one with the lowest prevalence. In case of tie, a piece is chosen in a uniformly distributed random way. Let $c_{i \to j}$ be the piece sent by n_i to n_j and $c_{j \to i}$ be the piece sent by n_j to n_i. After one round of exchanges, nodes update their availability vectors as: $\mathbf{a}_{n_i} \leftarrow \mathbf{a}_{n_i} \vee \mathbf{i}_{c_{j \to i}}$, and $\mathbf{a}_{n_j} \leftarrow \mathbf{a}_{n_j} \vee \mathbf{i}_{c_{i \to j}}$. where $\mathbf{i}_{c_{i \to j}}$ and $\mathbf{i}_{c_{j \to i}}$ are K element vectors with all positions set to 0 except the position relative to the piece just received, which is set to 1 ($\vee$ stands for the "OR" operator).

3. PEPIT: DESIGN AND IMPLEMENTATION

In this section, we present all the steps to bring PACS from theory to practice with the PePiT Android application.

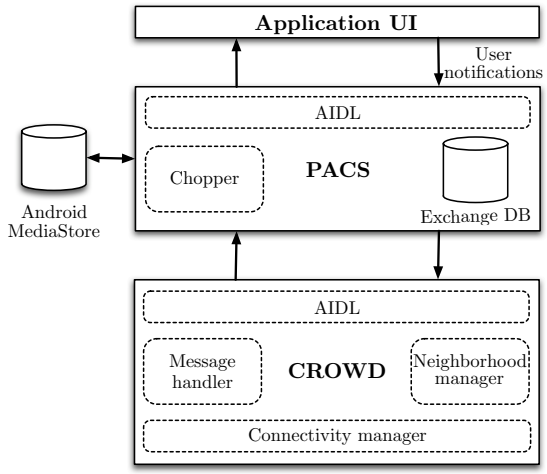

Figure 2: Extensible architecture of PePiT.

3.1 Architecture

A mobile application, called PePiT, has been developed for Android operative systems using JAVA. The application is mainly composed by two modules (Fig. 2), called PACS and CROWD[1], implemented as Android remote services. In this way, they can be called by other applications making the system extensible and able to exchange photos, videos, files, or any other contents. We decoupled the two modules so that the lower level functionalities of CROWD could be used by different higher protocols.

Application User Interface. The application user interface provides the possibility to exchange a photo either by picking it out from the gallery or by directly shooting it with the phone camera.

PACS Component. The *chopper* sub-module cuts the photo into K pieces filling the *availability bitmap vector* and creating the *prevalence vector*. The PACS service instanciates the PACS protocol and runs in the background. As such, it selects and sends content pieces using the CROWD service, as well as tracking every received piece into an internal exchange database.

CROWD Component. CROWD service is composed of three sub-modules. The *connectivity manager* sub-module creates (or connects to) an ad hoc network, and offers an interface for sending and receiving packets through UDP or TCP, in unicast or broadcast. The *neighborhood manager* broadcasts UDP beacons every T seconds in order to announce its presence to the neighbors. It also keeps state of the device neighborhood.

3.2 Deployment on Android mobile device

PePiT runs on mobile phones equipped with Android system with a minimum API level equal to 8 (Android Froyo, version 2.2.x). It has been successfully installed on a virtual machine (VM) running android-x86 [2].

As the Android system does not provide an API to manage IEEE 802.11 ad hoc communications, it is foremost required to follow a procedure, called *rooting*, in order to gain administrative access rights on the phones. Taking advantage of the Android NDK (Native Development Kit) tools, we also compiled *Linux wireless tools* for ARM and wrapped them into PePiT. In this way it is possible to connect the smartphone to an ad hoc network just like in a Linux environment. As a future work, we could use a method like Wifi-Opp [3] or the new Wifi Direct [4] to make the application runnable also on unrooted stock smartphones.

[1]CROWD is the name of the project that supports this work (http://anr-crowd.lip6.fr).

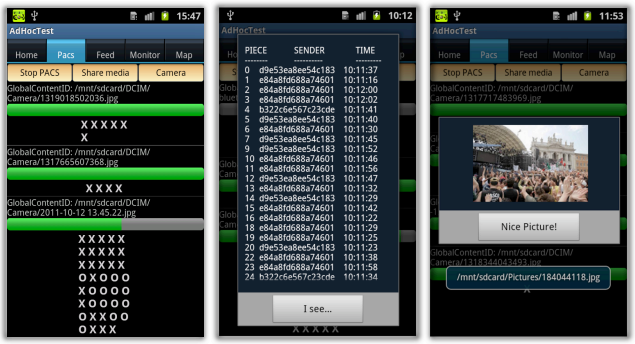

Figure 3: PePiT. From left to right: exchanges in progress, received pieces from different peers historical, preview of the received picture.

4. DEMONSTRATION

For the demonstration, we provide 4 HTC Desire and 8 Samsung Galaxy-S-II, equipped with Android 2.3.3, as well as a laptop with a virtual machine running android-x86 with Android 2.2.1. While the android-x86 screen is projected, the phones can be distributed to some people in the public.

Once PePiT is started, the mobile phones connect to the *CrowdAdHoc* ad hoc network and autoconfigure an IPv4 address. They are then ready to exchange pictures to other phones in their vicinity.

For every content exchanged, a progress bar is displayed (Fig. 3) with the detail of the missing and received pieces. When the download is complete, the image is stored in the phone gallery and a preview is showed by clicking on the item.

Acknowledgment

This work is partially supported by the ANR Crowd project under contract ANR-08-VERS-006.

5. REFERENCES

[1] N. Belblidia, M. Dias de Amorim, L. H. M. K. Costa, J. Leguay, and V. Conan, "PACS: Chopping and shuffling large contents for faster opportunistic dissemination," in *WONS*, Bardonecchia, Italy, 2011.

[2] [Online]. Available: http://www.android-x86.org

[3] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre, "Wifi-opp: ad-hoc-less opportunistic networking," in *ACM CHANTS*, Las Vegas, Nevada, USA, 2011, pp. 37–42.

[4] [Online]. Available: http://www.wi-fi.org/Wi-Fi_Direct.php