Opportunistic Content-Centric Networking: The Conference Case Demo

Sylvia Kouyoumdjieva, Emre A. Yavuz, Ólafur Helgason, Ljubica Pajevic, and Gunnar Karlsson
Laboratory for Communication Networks
School of Electrical Engineering, KTH
100 44 Stockholm, Sweden
{stkou, emreya, olafurr, ljubica, gk}@kth.se

***Abstract*—We present a demonstration scenario to evaluate a middleware architecture that we designed and implemented for distributing content over mobile ad hoc networks. The peer-to-peer networking architecture allows content dissemination between mobile devices without relying on any infrastructure support. Content is exchanged opportunistically when nodes are in proximity. We developed a mobile application utilizing the services provided by the implemented middleware. The application facilitates opportunistic content distribution in both one-to-one and one-to-many dissemination modes.**

I. Introduction

Wireless networks have traditionally been established either between fixed infrastructures or to provide connectivity for mobile users via fixed infrastructures. Recently, cooperative communication between mobile users has also been incorporated in the form of wireless ad hoc networks. The metamorphosis of mobile devices into smartphones has made content distribution popular among mobile users. Content is distributed by downloading when such a device is either docked to a computer with internet access or linked to fixed infrastructure such as cellular networks or Wi-Fi access points. The former mode is enabled by the massive storage available, yet distribution is limited to content present at the time of docking. Frequent downloading opportunities can be obtained with access via public Wi-Fi networks when devices are on the move but coverage is limited. Cellular networks provide good coverage and continuous access to content, yet if downloading of large-size content, such as multimedia files, becomes highly prevalent, its adverse impact on cell load along with high pricing and net neutrality will be the primary concerns.

We have designed and implemented a middleware architecture that allows any mobile application that utilizes its services, to distribute content opportunistically: mobile devices can either utilize connections to fixed infrastructure, such as public access points in range, or distribute content to other devices in proximity. Hence, the availability of the content can be extended beyond the reach of the limited-range fixed infrastructure or the motivation can simply be to avoid high prices and net propensity that might be experienced on a cellular network. Traditionally, network architectures focus on addressing *nodes* and forwarding packets between such nodes. The middleware aims at disseminating *contents* with a mechanism that relies on opportunistic content forwarding while abstaining from any routing substrate. As a result, sophisticated multi-hop communication protocols are not needed since the system design does not assume a traditional network layer with point-to-point unicast routing. Scaling comes naturally as popular content is likely to be available on many nodes in the system. The architecture is inspired by podcasting and BitTorrent, yet our operating scenario is radically different than what is experienced on the wired network. It has to cope with sporadic contacts, none or limited end-to-end connectivity and contact duration times. The rest of the paper is organized as follows. In section II we give an overview of the designed architecture. The description for the implementation of the architecture is given in section III and the demonstration scenario to evaluate the middleware is presented in section IV. We conclude in section V.

II. System Architecture

A general instantiation of the system may consist of three domains, as shown in Fig.1. The architecture adopts and extends the content structure of the Atom Syndication Format [1] to organize the content. This format has primarily been used for publishing web-feeds and podcasts on the net. Our system [2] imposes a hierarchical structure on contents by organizing them into *feeds* which function like containers for *entries* that contain the actual data objects of interest. Each feed can have multiple entries published at different times by different entities. Each entry can optionally have a range of other elements including zero or more *enclosures* which can be a single file attachment of types such as audio, video, or text. We provide a programming interface, inspired by the Java Message Service (JMS) publish/subscribe API [3], that any mobile application can use to access the services of our middleware. The interface implicitly defines the content structure for applications and allows them to publish/subscribe to content feeds. A synchronization manager processes contents shared by the client applications and solicits new contents on behalf of them. A discovery module finds which neighbors are running the service, decides which is feasible to associate with and sends a notification to the system.

The sharing of content is based on a request-reply protocol by which a node solicits entries for one or more feeds from a peer (a peer node can either be a mobile device or a gateway to the internet). Hence, there is no flooding of content

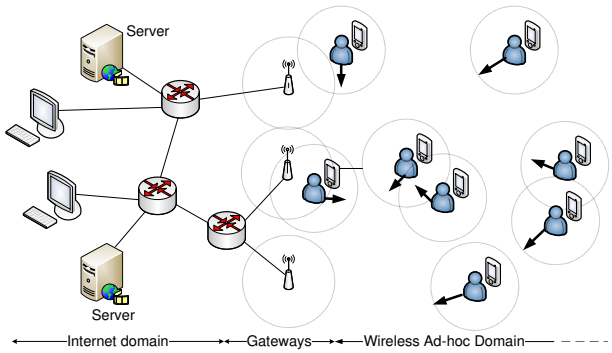
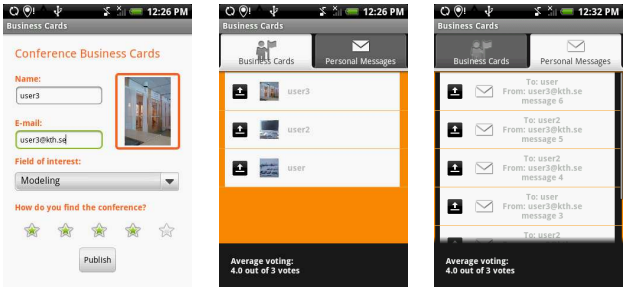


Fig. 1. The system composed of servers, wireless gateways and mobile devices



(a) (b) (c)

Fig. 2. (a) Creating a new user profile (b) Previewing received user profiles (c) Previewing received personal messages

in the broadcasting area. Protocol messages consists of the following types: `hello`, `request`, `reply`, `reject` and `close`. When a node discovers a new peer, it first sends a `request` message to the peer to initiate a unilateral session for downloading. The request contains either a query for a particular feed entry or for meta-data to discover content availability. The peer sends a `reply` message, establishing the session and replying to the query. Ungraceful session termination (e.g. when nodes move out of range) is handled by a soft-state timer; if there is no activity from the peer for a certain time, the session is closed and any allocated resources are freed up.

III. Implementation

The designed architecture has been implemented in Java for the Android OS and tested on the HTC Hero mobile devices. The implementation is based on the 802.11 ad-hoc mode which is currently not supported by the Android Java libraries but by both the driver and the hardware interface on HTC Hero devices. Hence, it requires the devices to run in privileged user mode (i.e. rooted mode) so that the interface can be reconfigured to run in ad-hoc mode. The middleware is implemented as an Android *service* which runs in the background and uploads/downloads content to/from other nodes that it discovers. Client applications can *bind* to the service and communicate with it by means of remote procedure calls through the publish/subscribe interface that it exposes. The discovery module is implemented as two threads. One thread periodically broadcasts `hello` messages on a well-known UDP port and a listener thread waits for incoming `hello` messages from other nodes. The implementation supports multiple active upload and download sessions at a time.

IV. Demo Scenario

To validate our middleware, we demonstrate a scenario where users are equipped with HTC Hero mobile devices that exchange user-generated content on the fly. In the description below we assume that demonstration scenario takes place at a conference venue. However, any other mingling event can as well be considered. First, we let users create profiles on devices with some basic information about themselves such as a self-photo, taken with the phone camera, a name, research interests and a username. Moreover we prompt them to rate the conference venue during this set up. The display for creating a new user profile in shown in Fig. 2(a). Pressing the *publish* button triggers three events: *(1)* the new profile is published to the *conference* feed to be disseminated to users carrying devices that run the same application, *(2)* user's vote on the venue is published to the *voting* feed to be shared similarly, and *(3)* user's device subscribes itself to *conference* and *voting* feeds along with a third one labelled with the *username* given in the profile. Users who are in proximity will then receive the published profile along with the vote on separate feeds and disseminate them to others. Fig. 2(b) illustrates how a list of received user profiles might look like. At the bottom of the display, voting statistics based on the collected results is also given. The results may differ from user to user due to the various numbers of contacts a user might have until then.

In addition to the one-to-many dissemination mode described above, each user can also send a personal message to any other user who shared a profile with a *username*. In Fig. 2(c), list of such sent/received messages are shown. As opposed to user profile and vote exchange, personal message exchange is achieved on one-to-one dissemination mode. Thus, when a personal message is sent, it is published on a feed labelled by the recipient's username. Since a user's device subscribes automatically to its username upon profile creation, the personal message will only be received by the intended user.

V. Conclusion

We present a demonstration scenario to evaluate our middleware, designed and implemented for opportunistic content-centric networking. The content dissemination in our system is purely interest-driven however content caching and forwarding is one of our primary directions for future work.

References

[1] M. Nottingham and R. Sayre, "The Atom Syndication Format," RFC 4287, Dec. 2005.
[2] Ólafur Helgason, E. Yavuz, S. K. L. Pajevic, and G. Karlsson, "A mobile peer-to-peer system for opportunistic content-centric networking," in *Proc. ACM SIGCOMM MobiHeld workshop*, 2010.
[3] "Java message service (jms)," http://java.sun.com/products/jms/.