SCAMPI Application Platform

Teemu Kärkkäinen
Aalto University
teemuk@netlab.tkk.fi

Mikko Pitkänen
Aalto Univeristy
mikko.pitkanen@aalto.fi

Paul Houghton
Futurice
paul.houghton@futurice.com

Jörg Ott
Aalto University
jo@netlab.tkk.fi

ABSTRACT

In this paper we demonstrate an application platform architecture and implementation that allows developers to easily target opportunistic networks. The platform includes an opportunistic router, HTML5 application development framework, and an opportunistic application market for distributing applications. We demonstrate the platform and multiple HTML5 applications – including chat, music and social networking applications – running on Android devices.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Algorithms, Design, Experimentation

Keywords

Delay-tolerant Networking; DTN; Opportunistic Networking; HTML5

1. INTRODUCTION

The development and deployment of opportunistic networking applications on real consumer devices has been slow. This is in large part due to the lack of suitable frameworks and software stacks on popular consumer mobile device platforms. Consumer mobile devices lack this support because they are in the majority of cases designed to serve as clients seeking to access services in some infrastructure network (typically the Internet), and most networking technologies and software stacks are aimed at implementing this goal. This leaves support for opportunistic peer-to-peer communication to be non-existent or at most an afterthought, even when the underlying hardware is capable of much more.

Most mobile platforms that have come to the market in the past decade have exposed their own native software development frameworks, tools, APIs and even programming languages (e.g., Symbian, iOS, Android, webOS). Application portability between these platforms has been very poor due to the lack of standardization. However, in the recent years mobile HTML5 [3] application development has been gaining in popularity as it provides a standardized set of APIs that work on all platforms with modern web browser support. Therefore, creating a set of HTML5 APIs and frameworks for opportunistic communications would allow application developers to easily take an advantage of the new communication opportunities provided by opportunistic networking in platform independent manner.

After the applications have been developed, they must be distributed. Different mobile platforms have their own distribution mechanisms for native applications. Some allow the user to directly install downloaded software onto their devices, while some provide centrally controlled application stores. However, all of the distribution mechanisms are primarily designed for distributing native applications while providing little support for HTML5 applications, leading to developers having to wrap their HTML5 applications in native applications. The SCAMPI Application Platform solves this problem by allowing direct distribution of HTML5 applications over the opportunistic network itself. These applications contain only the HTML5 elements (HTML, CSS, JavaScript) and no native code for any platform. The HTML5 applications can be published by and run on any device that supports the SCAMPI platform.

In this paper we describe our solution for an opportunistic application development platform with support for both native application development and HTML5. The platform includes an opportunistic router based on the DTNRG specifications (Section 2), a framework for developing pure HTML5 applications that take advantage of the router (Section 3), and an opportunistic application distribution system (Section 4). Finally, we will demonstrate three opportunistic applications (chat, music, and social networking) written in HTML5 for our platform (Section 5).

2. SCAMPI ROUTER

The fundamental enabler for opportunistic communications is a store-carry-forward router. The router maintains caches of messages, attempts to discover peers and open contacts with them, and routes messages over multiple hops. The SCAMPI Platform includes an opportunistic router based on the DTNRG architecture and protocols [5][1][7], and our own extensions (bottom in Figure 1).

The router uses various discovery mechanisms, including IP multicast/broadcast beaconing, TCP unicast discovery and subnet scanning for known ports. The combination of different discovery mechanisms increases the likelihood of successful discovery in the face of varying policies on, for example, blocking of multicast/broadcast traffic, as well as allowing the use of known infrastructure nodes

CHANTS'12, August 22, 2012, Istanbul, Turkey.

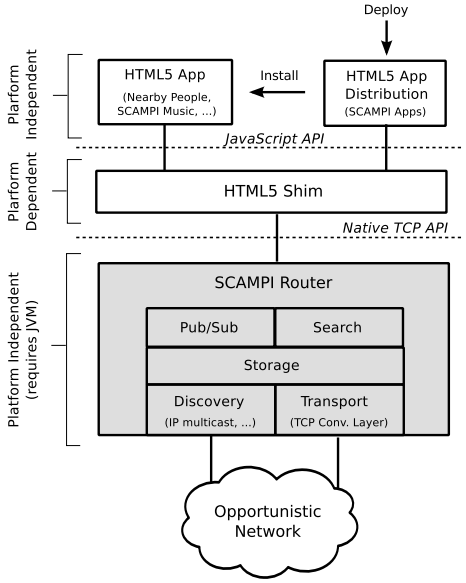

Figure 1: SCAMPI Application Platform architecture.

when possible (unicast discovery). After peers have been discovered, links must be opened between them. The SCAMPI Router uses TCP Convergence Layer (TCPCL) for all message transmissions between peers in IP-based networks. Adding additional convergence layers is supported for non-TCP/IP-based networks (e.g., Bluetooth) or for transports other than TCP (e.g., UDP, LEDBAT [6]). Each SCAMPI Router instance has a cryptographic identity (currently RSA-based), which allows messages to be signed and verified, as well as encrypted. This also enables external identities, such as Twitter accounts, to be cryptographically bound to the node identities if the external identity provider supports mechanisms such as OAuth [2].

The router provides a native Application Programming Interface (API) over TCP that supports the following operations: *publish/subscribe* for messages, automatic *framing* for structured messages, *searching* for content-based on message metadata, and *peer discovery* of nearby nodes. The TCP API can be used by native applications written in any language by implementing the client side of the SCAMPI API protocol (currently only Java client implementation is provided).

The platform enables application developers to work with entities called *SCAMPIMessages*. These are self-contained, semantically meaningful application layer objects that map to Bundle Protocol bundles at lower layers. SCAMPIMessages are not opaque binary blobs unlike the payload of most Bundle Protocols implementations (e.g., DTN2, IBR), but instead have a map structure where arbitrary string keys map to binary buffers, strings, numbers or file pointers. The framing and serialization of this map into a bundle is handled by the router. This means that the application developer does not need to define the framing format or (de)serialize the message data, and also allows all the SCAMPI Routers to understand the structure of the messages. Understanding of the message structure can be exploited to develop mechanisms that operate on parts of the message, such as versioning or automatic merging, although these are not currently supported by the platform. Furthermore, each SCAMPIMessage can be tagged with namespaced metadata that describes the content, for example, if the content of the message includes a music file, the artist, title, genre, etc. can be exposed.

These messages can then be published to *services* identified by opaque strings. Any node subscribed to the service will receive copies of the published messages. Since the messages can be tagged with metadata, search queries can be executed against this metadata by any SCAMPI Router without the need to understand the content semantics. It is also possible to limit the spreading of messages geographically, allowing floating content [4] style distribution.

Implementation of the router is in plain JavaSE, allowing it to be run on any platform with a Java Virtual Machine, including popular platforms such as MacOS X, Windows and GNU/Linux. Further, we have created an Android application that runs the router as a persistent background process on Android devices and allows other applications to use the services provided by the router through the native TCP API.

3. HTML5 APPLICATION FRAMEWORK

While the router provides a TCP-based API that can be used by native applications, we have also created a framework for developing applications in HTML5 (top of Figure 1). Unlike native applications that are tied to a single platform, HTML5 applications can be written once and run on any platform supported by SCAMPI.

The SCAMPI HTML5 support is realized as a thin shim whose responsibility is to translate messages between the SCAMPI router and the HTML5 application framework. In particular, the shim provides a light-weight native container that can load and run the platform independent HTML5 application packages. The shim further exposes a JavaScript API that can be used by the HTML5 applications to communicate with the underlying SCAMPI Router. In particular, the JavaScript API contains functions to publish/subscribe SCAMPIMessages and to register callbacks for peer discovery events. The current implementation of the shim is written as platform dependent Android code, but porting it to other platforms such as iPhone is straightforward (although this would also require a native iOS implementation of the SCAMPI Router since iOS does not include a Java runtime).

The HTML5 application packages contain the entire logic to implement the applications in fully platform independent form. The application logic in our demonstration applications is implemented by using JavaScript model-view-controller (MVC) frameworks, which are a popular way of structuring mobile and web applications. One of the benefits of the MVC architecture is that the data model can be separated from the view. The messages from the SCAMPI router update the data model, and the programming frameworks, for example, currently used backbone.js [1], are responsible for rendering the application view on the user's device. User actions then update data model and are propagated to the SCAMPI router through the shim. This allows HTML5 developers to use the same frameworks and libraries they would normally use while the SCAMPI JavaScript APIs provide additional capabilities that the developer can take advantage of for opportunistic networking.

Using the HTML5 framework allows the same application code to be used for managing both the model and the rendering of the view independent of which device is used for executing the application. This is possible because the modern smartphones provide rich and uniform support for the HTML rendering and JavaScript.

4. APPLICATION DISTRIBUTION

SCAMPI application developed in HTML5 is packaged into a self-contained set of HTML, JavaScript, and CSS files. These files

[1] http://documentcloud.github.com/backbone/

contain the UI views on the users device (HTML and CSS) as well as data models and control for the application logic (JavaScript libraries and custom JavaScript).

A key benefit of the approach is that the applications contain only code that does not need to be compiled for any specific execution platform, but requires only support for functionality provided by any modern web browser.

These packaged applications are then sent as SCAMPIMessages similarly to all other messages. This allows all SCAMPI mechanisms to be applied to the applications themselves, including metadata-based search and the security features such as message signing to verify the originator of the application.

The distribution system itself is created as a SCAMPI HTML5 application, which publishes, receives, validates and installs bundled HTML5 applications.

Distributing the applications in an opportunistic manner in self contained messages avoids the need to publish applications through centralized authority that controls the market place. Together with easy-to-develop web programming approach this makes the SCAMPI design a compelling way to share information when sporadic interests, for example festivals or street protestations, take place.

5. APPLICATIONS

We have developed several HTML5 applications to demonstrate the proposed approach. These include a *chat* application, SCAMPI Tunes *music sharing* application, and a NearbyPeople *social networking* application.

The simplest of our demonstrator applications, chat, simply allows users to post and read messages in a shared chat room. The messages posted by the participants are distributed opportunistically, and flood to all reachable devices in the vicinity without the need for a centralized chat room server (e.g., Internet Relay Chat).

Instead of only distributing short text messages, the platform can also distribute larger content items. We demonstrate this with the SCAMPI Tunes music sharing application that allows users to publish MP3s from the their devices so that other users around them can find and listen to them. The published music files are picked up and displayed by the surrounding clients, but they are not permanently stored. This means that the available music changes dynamically depending on the environment (and the music taste of surrounding people).

Finally, our most complex application is an opportunistic social networking application, NearbyPeople. This application allows users to input their profiles, including status information, contact information, interests and a profile picture. Surrounding nodes can view the profiles of nearby people, find profiles with matching interests, send messages between users and schedule ad-hoc social events. This allows popular social networking interactions without relying on infrastructure, without uploading personal data to a centralized repository, and without spreading personal information beyond the nearby physical environment.

All of the demonstrated applications are written in HTML5 and distributed through the SCAMPI HTML5 app distribution mechanism.

6. CONCLUSION

In this paper we have described the architecture and implementation of and application development platform that enables developers to easily take advantage of opportunistic networking, including a number of HTML5 example applications that run on top of the framework.

In the future, more functionality can be added to the underlying router, such as shared content authoring (e.g., forums and discussion boards) and exposed to the applications through the platform independent HTML5 APIs. All while also enabling direct application deployment and distribution through the opportunistic network itself.

We believe approach provides application developers a low-effort means to take advantage of opportunistic networking, potentially leading to wider scale adoption of these techniques in real, production applications.

Acknowledgements

This work received funding from the Academy of Finland in the RESMAN project (grant no. 134363) and from the European Community's Seventh Framework Programme under grant agreement no. 258414 (SCAMPI).

7. REFERENCES

[1] Mike Demmer and Jörg Ott. Delay Tolerant Networking TCP Convergence Layer Protocol. Internet Draft draft-irtf-dtnrg-tcp-clayer-01.txt, Work in Progress, February 2007.

[2] E. Hammer-Lahav (Ed.). The OAuth 1.0 Protocol. RFC 5849, April 2010.

[3] Ian Hickson (Ed.). HTML5 - A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, March 2012.

[4] Esa Hyytiä, Jorma Virtamo, Pasi Lassila, Jussi Kangasharju, and Jörg Ott. When does content float? characterizing availability of anchored information in opportunistic content sharing. In *IEEE Infocom*, Shanghai, China, April 2011.

[5] Keith Scott and Scott Burleigh. Bundle Protocol Specification. RFC 5050, November 2007.

[6] Stanislav Shalunov, Greg Hazel, Janardhan Iyengar, and Mirja Kuehlewind and. Low Extra Delay Background Transport (LEDBAT). Internet Draft draft-ietf-ledbat-congestion-09.txt, Work in progress, October 2011.

[7] Susan Symington. Delay-Tolerant Networking Metadata Extension Block. Internet Draft draft-irtf-dtnrg-bundle-metadata-block-00, Work in progress, September 2008.