Practical Voice Communications in Challenged Networks

Md. Tarikul Islam[1], Anssi Turkulainen[1], Teemu Kärkkäinen[1], Mikko Pitkänen[2], and Jörg Ott[1]
[1]Helsinki University of Technology, Department of Communications and Networking
[2]Helsinki Institute of Physics, Technology Programme

***Abstract*— Today's Push-to-talk (PTT) services are infrastructure based and require stable end-to-end paths for successful communication. However, users with PTT enabled mobile devices may roam in challenged environments where infrastructure is not available. In such cases the traditional PTT services may exhibit poor performance or even fail completely. Delay-Tolerant Networking (DTN) is a research area which addresses the communication requirements specific to challenged networks. In this paper, we apply the DTN concept of asynchronous message forwarding to transmit voice messages. We present an implementation of the DT-Talkie for heterogeneous clients such as Maemo, Symbian and Android based personal handheld devices. We also demonstrate the use of a DTN throwbox with large storage capacity to improve message forwarding.**

I. Introduction

Push-to-talk (PTT) is a half-duplex voice service which provides individual and group communications in "walkie-talkie" fashion. PTT services have become increasingly popular due to proliferation of personal handheld devices (e.g., PDA, Smartphone) around the world. PTT is commonly employed in mobile phones, which use a single button to switch between voice transmission mode and voice reception mode. One popular implementation of PTT is called Push-to-talk over cellular (PoC), which is based on 2.5G or 3G packet-switched networks. The Open Mobile Alliance (OMA) has developed an open standard for PoC [2].

PoC service has been suggested for operator independent wireless networks (e.g., WLAN, Bluetooth). Lin-Yi Wu et al. [7] have proposed a client architecture for the PoC service based on the OMA specification and implemented it in the WLAN environment. The service relies on infrastructure for its operation. However, the PoC enabled mobile devices may be used in environments where infrastructure is not available.

Rönnholm [5] presents an outline for a push-to-talk system over Bluetooth, which allows users to communicate in peer-to-peer fashion. The system requires sufficient node density to establish end-to-end path and eventually fails to communicate in sparse mobile ad-hoc environments. All of the PoC solutions discussed above depend on the traditional Internet protocols which require stable end-to-end paths for successful protocol operations.

Delay-Tolerant Networking (DTN) attempts to overcome the problems associated with frequent disconnection, intermittent connectivity, long or variable delays, and asymmetric data rates between the source and the destination. DTN allows communication in the sparse mobile ad-hoc and other extreme environments without the requirement of end-to-end path at any point in time. The Internet Research Task Force (IRTF) Delay Tolerant Networking Research Group (DTNRG) has been developing the DTN architecture [3] and the Bundle Protocol (BP) [6] in order to enable communications in challenged environments. BP provides common overlay service to interoperate among the wide range of network types, while the convergence layers (CL) perform mapping between the bundle protocol and network-specific lower layers. The DTN architecture, BP and the TCP convergence layer [4] serve as a basis for our demonstration.

In this paper we describe a system called DT-Talkie which enables both one-to-one and one-to-many communications over infrastructure-less and challenged environments in the walkie-talkie fashion. Complete voice messages are captured at the source and encapsulated in bundles. The voice bundles are then sent over DTN where they are transmitted using the store-carry-forward mechanism. The voice message is played back when it reaches the destination node. In addition, DT-Talkie includes an application level framing mechanism that allows other information, such as images, to be sent along with the voice messages.

We demonstrate an implementation of DT-Talkie for a diverse set of mobile user devices. The DT-Talkie is implemented for Maemo based Nokia Internet Tablet. We have ported the DT-Talkie application to OpenMoko based smartphone, Linux and Mac. Furthermore, we are developing cross-platform support for DT-Talkie with Symbian and Android based smartphones. We also present a mobile throwbox with message repository as an auxiliary DTN node, which is used to relay voice messages between mobile nodes.

The rest of this paper is structured as follows: Section II describes the DT-Talkie architecture. The DT-Talkie implementation is presented in Section III, followed by a description of the demonstration setup in Section IV.

II. System Architecture

The general processing steps of DT-Talkie application are shown in Figure 1. We describe the architecture with the aid of a one-to-one communication scenario. Suppose user A intends to send a voice message to user B and both have DTN communication capabilities. First, DT-Talkie captures voice from the audio source (e.g., microphone) of user A and encodes the voice using an audio coding technique. DT-Talkie also includes an image along with the voice message. The image can be either user A's profile picture or instant snapshot from the device's camera.

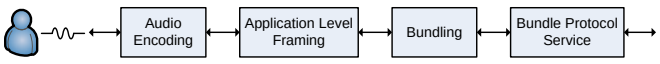

Fig. 1. DT-Talkie General Processing Steps

Second, application layer framing mechanism is used to create a combined message of the voice and image data. We use MIME multipart encapsulation for this purpose. `X-Bundle-Destination` header field is added to distinguish messages sent in the context of one-to-one and group conversations. The voice message is encoded as a body part of type `audio/G729` and image as another body part of type `image/jpeg`. Finally, the MIME message is encapsulated in a bundle and handed to the DTN layer for delivery.

When the bundle is received by user B, DT-Talkie decapsulates the bundle payload. Then the MIME message is parsed to get the encoded audio data and the image. Finally, DT-Talkie decodes the audio data and starts playback in the audio sink (e.g., speaker). Appropriate action is taken with the image (e.g., shown in the GUI).

All the nodes in the DTN communication are identified by a URI endpoint identifiers (EID). Every node has a unique singleton EID but can also register to any number of multicast EIDs. The general structure of an EID is: <scheme name>:<Scheme-Specific Part, or "SSP">. BP defines "dtn:" EID scheme with an arbitrary SSP. When an application wishes to receive bundles destined to a particular EID, it registers the corresponding EID with its local DTN node.

One-to-many, or group communications work the same way as one-to-one communications, but the messages are destined to a multicast EID. Any nodes wishing to receive messages to that group can register with the corresponding EID.

In extremely sparse scenarios direct contacts between nodes can be so infrequent that even store-carry-forward type message delivery is not feasible. However, even if the nodes do not come to direct contact with each other, they may pass the same location. We have developed a stationary throwbox that can be placed in such location. A throwbox is a BP agent but does not have any DT-Talkie specific functionality. This allows it to pick up and store messages from passing nodes and forward them to other nodes.

III. Implementation

We use DTN2 [1] as the Bundle Protocol agent in our implementation of DT-Talkie in Linxux platform. DTN2 is the reference implementation of the Bundle Protocol developed by DTNRG. DTN2 supports TCP, UDP, Bluetooth, Ethernet and Sneakernet convergence layers. It also includes support for several link types such as always-on links, on-demand links, opportunistic and scheduled links. We are currently developing inter-communication between DTN2, Symbian and Android BP implementations.

In DTN2, an EID takes the canonical form: "dtn://<node identifier>/<application tag>", where router identifier is a DNS-style hostname string and application tag is any string of URI-valid characters. In our implementation, "<host>.dtn" is used as node identifier and "dttalkie" is used as application tag (e.g. dtn://Nokia-N810.dtn/dttalkie) to identify the endpoints.

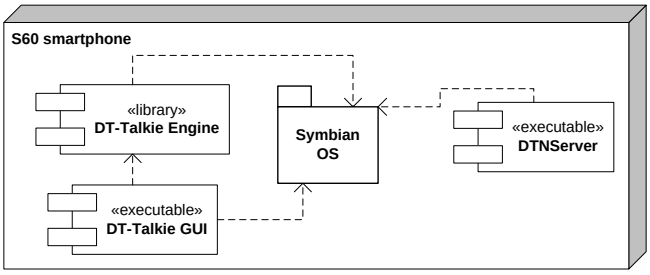

Fig. 2. DT-Talkie Application Architecture for Symbian

In the GUI module of DT-Talkie we use platform-specific GUI framework, such as GTK+ and Hildon framework in Maemo and GUI framework in Symbian, to draw the main window and other widgets. We use also platform-specific audio framework to capture and playback voice messages. MP3 codec is used to encode the captured voice from audio source and decode to playback audio in the sink. BP service API is used for sending, receiving and forwarding bundles over DTN architecture. We use DTN2 API for Linux based DT-Talkie and platform-specific DTN API for Symbian and Android based DT-Talkie to use the bundle protocol services.

The application architecture of DT-Talkie in Symbian is shown in Figure 2. The Maemo implementation follows a similar pattern but DTN2 is used instead of using a custom made DTNServer. The Android implementation is otherwise similar but does not use an external DTNServer, instead using an embedded TCP CL.

IV. Demonstration Setup

In the demonstration setup, we use a throwbox, three Nokia Internet Tablets, one laptop and two smartphones. The latest release of DTN2 (version 2.6.0, released in July 2008) is ported to Linux based Nokia Internet Tablets. We also use our own DTN implementation for Symbian and Android based smartphones. We use epidemic routing to spread voice messages. For persistent storage service, we use local file system to store the bundles.

In Figure 3, DT-Talkie and dtnd (DTN2 routing daemon) are running in the devices of User A, B, C, D and E. Only dtnd is running in the forwarder node Q and the throwbox R, which is a fixed message repository with enough storage capacity. We create disruptions to get the DTN experience through disconnecting the link between the node R and User B. The disconnection can be created artificially or by moving outside local wireless communication range.

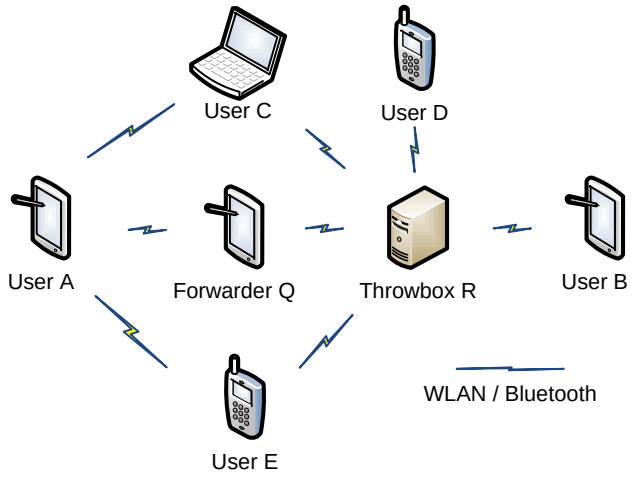

Fig. 3. Demonstration Setup

When User A sends a voice message destined to User B, the message is queued in the node R. After a while, we enable the connection between the node R and User B. The node R then forwards the message to User B as soon as the link becomes available between itself and User B. In the same setup, we also demonstrate the scenario of group communication through sending voice message from User A to User B, C and D.

References

[1] DTN2. http://www.dtnrg.org/wiki/Code, May 2006.

[2] Open Mobile Alliance. Push to talk over Cellular (PoC) Architecture. OMA-AD-PoC-V1 0-20060609-A, June 2006.

[3] Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, and Howard Weiss. Delay-tolerant networking architecture. RFC 4838, April 2007.

[4] Mike Demmer and Jörg Ott. TCP Convergence Layer. Internet Draft draft-demmer-irtf-tcp-clayer-02.txt, November 2008.

[5] Valter Rönnholm. Push-to-Talk over Bluetooth. Hawaii International Conference on System Sciences, January 2006.

[6] Keith L. Scott and Scott C. Burleigh. Bundle Protocol Specification. RFC5050, November 2007.

[7] Lin-Yi Wu, Meng-Hsun Tsai, Yi-Bing Lin, and Jen-Shun Yang. A Client-Side Design and Implementation for Push to Talk over Cellular Service. Wireless Communications and Mobile Computing, June 2007.