Voice Messaging for Mobile Delay-Tolerant Networks

Md. Tarikul Islam
Aalto University Comnet
mtislam@cc.hut.fi

Anssi Turkulainen
Aalto University Comnet
anssi.turkulainen@aalto.fi

Jörg Ott
Aalto University Comnet
jo@comnet.tkk.fi

***Abstract*—Mobile ad-hoc networks, especially when they are sparse, are not well suited for ad-hoc voice communication when using end-to-end real-time audio streams: as paths grow in length, stability and forwarding performance suffer, and sometimes paths may not exist at all. In this paper, we suggest using asynchronous voice messaging built upon delay-tolerant networking concepts with some degree of hop-by-hop reliability to enable communication even in sparse environments and ensure intelligible speech. We evaluate the resulting walkie-talkie-style service in different synthetic and mobility-trace-driven settings and report on our experience from interoperable implementations on various mobile device platforms.**

I. Introduction

Voice telephony is one key means of human interaction, today complemented by other close-to-instant forms such as chat and text messaging. Depending on the situation and the information to be conveyed, speaking and listening may be less distracting or more efficient than typing and reading. However, synchronous voice conversations require users to be available at the same time and can be more intrusive than, e.g., messaging. This also applies to Push-to-talk (PTT), a half duplex voice service that provides individual and group communication in a walkie-talkie fashion. PTT is available in mobile phones, with a single button to toggle between voice transmission and reception. One version of PTT, called Push-to-talk over cellular (PoC) [1], is based on 2.5G or 3G packet-switched networks and thus requires cellular network infrastructure to function (in contrast to amateur radio walkie-talkies). Voice mail allows effectively circumventing both restrictions above, at the cost of reduced interactivity.

The three above types of mobile voice communication impose essentially no restrictions on the locations of the communicating users, but all rely on infrastructure networks. These need to be utilized even if the users are near each other. This may prevent communication if 1) the cellular access networks are congested, 2) the network is not accessible because users are not authorized (e.g., due to missing roaming agreements, restrictive tariffs, of empty prepaid calling cards) or due to lack of network coverage, or 3) the cost is deemed prohibitive for a (roaming) user.

Instead of relying on cellular infrastructure, it may be desirable to bypass such networks for local conversations, as has been widely discussed for data communication in mobile *delay-tolerant*, *pocket-switched*, or *opportunistic* networks (e.g., [2], [3], [4]), to circumvent the above limitations. As ad-hoc networks formed by cooperating mobile users may be sparse so that packet-based synchronous voice communication (e.g., [5]) may often not be feasible, we turn to asynchronous voice messaging only: such a voice conversation resembles two-way-alternate walkie-talkie-style communication if the nodes are well connected and messages can be delivered virtually instantaneously, but degrades to asynchronous voice messaging as delivery delays within mobile networks grow.

Several studies have explored the feasibility of voice messaging as a primary means for interpersonal communication [6], [7], [8] that emerged in parallel to our own earlier work [9]. In this paper, we extend our past experimental work in two ways: 1) We provide a comprehensive system and protocol design and implementation that also addresses interoperability aspects known from traditional telephony. 2) We quantify the performance of asynchronous voice interactions in opportunistic mobile environments for different (urban) scenarios to understand the feasibility for different types of interactions.

II. Interactive Voice messaging

We envision an application running on mobile devices that offers asynchronous and interactive voice messaging in scenarios where voice interaction is a more convenient way for communication than using text communication (e.g., because of ease to use [7] and requiring less attention); and where infrastructure networks are not necessarily available.

We opt for asynchronous message-based communication because this allows us to address two major issues of mobile ad-hoc environments: 1) Network partitions and unstable or non-existing end-to-end paths due to sparse node distributions and mobility make packet-based end-to-end communication unlikely to succeed [10], [11]. 2) Performance degradation observed for wireless multihop communication may impede packet-based voice even if an end-to-end path exists [12].

We overcome both issues by applying delay-tolerant networking [13] concepts based upon asynchronous store-carry-and-forward messaging: voice statements are recorded locally and packed into DTN messages to be forwarded as a whole. We assume an underlying communication substrate following the DTNRG architecture [14] and the bundle protocol [15] that offers best-effort delivery of virtually arbitrarily sized messages (*bundles*) to one or more destinations. Routing and forwarding is based upon single-copy or multi-copy routing protocols. DTN forwarding does not require instant end-to-end paths and tolerates disconnections. It performs error control on each hop so that messages are delivered completely and error-free if they reach the destination. We thus accept an increase in delay to improve reachability and voice quality.

We consider target scenarios that can be generalized to three classes as depicted in figure 1 where device A sends voice messages destined for device B. In the simplest case (a), device A and B are within radio range of each other (directly or indirectly via, e.g., WLAN infrastructure). In a more complex case, mobile devices form larger ad-hoc networks and one or more nodes forward the voice messages opportunistically towards the recipient: in a connected (b) or (partly) disconnected (c) network. In the latter case, senders and forwarders may store the messages in persistent memory until the forwarding opportunities become available, e.g., when the mobile device C moves physically towards the destination.

We assume that, in some scenarios, similar to voice mail, voice communication does not always need to be highly interactive and that delays are permissible. The delay D obviously varies with the connectivity between the involved peers as depicted in figure 1 and the message size S (the latter of which is turn a function of the audio encoding). The lower bound on delay is implied from the length of the statement or talkspurt T_s since this needs to be fully recorded first. If two nodes are in direct contact with each other (a), the additional delay is minimal and only depends on the net channel capacity C (in bit/s): $D = T_s + S/C$. In a connected multihop network with n hops (b) with a mean capacity C per hop, the store-and-forward nature of the network yields a delay of $D = T_s + n \times S/C$ in case of an otherwise empty network; queuing delays due to other messages add further to this. In a disconnected network (c), the time that passes between a node receiving a message and meeting the next hop to forward it to needs to be factored in; such inter-contact times depend on the node density and mobility characteristics.

For voice messaging, we expect mouth-to-ear delays in the order of seconds—compared to some hundred milliseconds or less for real-time interactive voice—when sender and receiver are close to each other, increasing with topological and generally with geographic distance. As message delivery delays grow (to minutes or even hours), interactive voice messaging converges to traditional voice mail-style communication, but bypassing the infrastructure networks. Latency limits interactivity and thus the applicability of voice messaging. Communication may further be constrained to limited geographic distance between sender and receiver(s) as the probability that a message is delivered at all is expected to decrease with network size and distance.

We expect voice messaging to provide a suitable communication means among peers in geographical proximity. This yields different potential real-world usage scenarios: In sports events, concerts, festivals, and similar events groups of people may seek to coordinate or find each other; when cycling, rollerblading, or hiking (especially in large events) groups of people may stay in contact even if they are somewhat apart; inside convention centers or large hotel facilities, people may interact for appointments without depending on cellular networks (that may be costly or whose coverage may be limited, e.g., in underground facilities), but possibly leveraging local WLAN infrastructure; and in remote or disaster areas, voice messaging may offer the only option for communication.

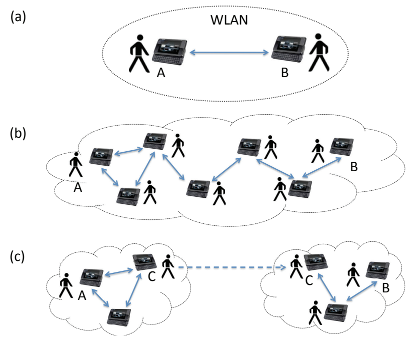


Fig. 1. Different delivery modes for voice messaging

III. DT-Talkie Design

A. System Operation

The general processing steps of the DT-Talkie are depicted in figure 2. If user A wants to send a voice message to user B in a one-to-one communication scenario, she manually starts and stops recording the message, e.g., using a button. The voice message is captured from the audio source (e.g., the microphone) of user A and encoded using at least one codec.

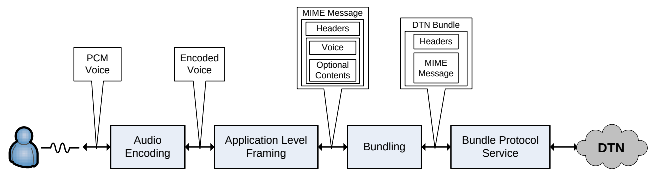


Fig. 2. General processing steps of the DT-Talkie

Message Encapsulation: After encoding, all pieces of a voice message are aggregated into a DTN bundle. In the simplest case, the message just comprises one audio segment; in more complex ones, multiple audio segments (e.g., talkspurts identified by voice activity detection). In the latter case, the relative timing between individual message segments must be preserved so that a header with timing information is added per segment as is a sequence number to detect losses.

To allow adding these headers, including multiple message segments, and sending auxiliary information along with voice, we use MIME as an extensible, recursive encapsulation scheme. MIME can also provide security mechanisms for message secrecy, authentication, and integrity protection (S/MIME). We define several headers: *X-Bundle-Destination* to distinguish one-to-one and group conversations; *X-Bundle-TS* and *X-Bundle-SeqNo* to include a timestamp and a per-sender message sequence number; and *X-Bundle-Type* to indicate if voice messages are sent in full-length or as fragments.

We define two auxiliary pieces of content to be included in the MIME message: a digital business card (vCard)[1] to

[1]http://www.icm.org/pdi/vcard-21.txt

provide information about the sender (display name) and an image—either a user's profile picture or instant snapshot from the device's camera—to offer further context.

Bundle Delivery: Finally, the MIME message is encapsulated into a bundle and sent using the Bundle Protocol services. When user B receives the bundle, the MIME message is decapsulated and its contents extracted, the headers being used to ensure ordering and proper rendering. The DT-Talkie decodes the voice message and starts playback to the audio sink (e.g., speaker) of user B. Any optional auxiliary contents is displayed (e.g., the originator name and the image are shown in the GUI). It might happen that user B does not support the necessary codec(s) to decode and playback received voice messages, in which case an error message is returned to A to negotiate codecs. We will return to this in section III-C.

Bundle Addressing: All the endpoints in the DTN domain are identified by a URI-like endpoint identifier (EID) [15] for which we use the *dtn:* scheme. Every node has a unique singleton EID but can register for any number of multicast EIDs. An EID takes the form: *dtn://node-id/application-id*. In DT-Talkie, <*host*>*.dtn* is used as the *node-id* and *dttalkie* is used as *application-id* (e.g., *dtn://nokia-n900.dtn/dttalkie*). For mobile phones equipped with a SIM card, a suitable default address would be the corresponding E.164 number as <*host*>.

Group Communication uses the same concepts as one-to-one communication, the main difference being that voice messages are destined to a multicast EID. The structure of a multicast EID is defined as *dtn://*<*group-name*>*.dtn/dttalkie*. If users want to receive voice messages from a particular group, they register with the corresponding multicast EID.

Since group conversations may have multiple senders, we apply a simple variant of *causal ordering* [16]: The sender includes the bundle identifiers[2] of the last k messages received before this message in the present message, text-encoded as a comma-separated list in an optional *X-Preceding-Messages* header. This allows the receiver to wait with playback until the messages triggering the last one have arrived. Optionally, especially if messages are small, the sender may include the entire previous message(s) so that the context of a statement is always provided and no extra waiting delay is added. The previous messages are included using the MIME multipart mechanisms (*multipart/related*) and are each labeled using the *X-Bundle-Source* header to denote the original sender. Further studies are required to determine how many message identifiers or messages should be included. Using a dynamic timeout accounting for past delivery delays inside the group and accordingly limiting the age of the messages to be considered could be a suitable means to determine which *recent* messages are included.

Voice Sessions: DT-Talkie implicitly sets up and manages "conversations" between users. A node is *idle* upon startup and when it has not received a voice message for some time. When a message comes in (from user A or belonging to group G) user B is alerted. If B plays the message, a session is implicitly set up: messages from B will by default be directed to A (or G) and incoming messages from A (or for G) will be played back without further user intervention as long as they arrive within a time window of the previous one sent or received in this session. If a message arrives from a different user C while B is in session with A, the message is queued and the user alerted; user controls are provided to toggle between sessions.

B. Voice Message Fragmentation

Sending large voice messages in a single bundle might not be feasible in some scenarios. For example, contact durations in opportunistic DTN environments may be too short to successfully transmit a large message. This suggests fragmenting a large message into smaller pieces to enable communication over short-lived links. In addition, when users are well connected to each other (e.g., low end-to-end delay) during an ongoing DT-Talkie session, delivering voice messages as fragments could help improving session interactivity.

Since bundle layer fragmentation may negatively impact message delivery [17], we apply fragmentation at the application layer following the concept of *application layer framing* [18]. Basically, each voice message contains a sequence of talkspurts (sentences) and silence periods. DT-Talkie separates the talkspurts from the silence periods, considers the talkspurts as segments, and maps each segment to an individual fragment. The fragments are then sent as different bundles. This approach may significantly reduce latency and thus increase the interactivity of a voice session.

We encapsulate voice fragments in individual MIME messages. Each MIME message carries two further headers: *X-Frag-No* and *X-Last-Frag*. *X-Frag-No* counts the fragment number within the message (starting from 0 for each message), *X-Last-Frag* indicates if a fragment is the last one of a particular message.[3] Including this metadata in a message allows the recipient to play back the voice fragments in the correct order and wait for all (or most) fragments to arrive.

For now, we keep fragmentation static: voice activity detection is used to identify talk spurts and fragments are generated and handed to the bundle layer for transmission as soon as they are recorded. It is up to the bundle layer to buffer messages when no (suitable) next hop is available and send out fragments in bursts as soon as an opportunity arises, knowing that this incurs additional per-message overhead (as we evaluate below). Future work will investigate dynamic adaptation of fragmentation to the observed network connectivity.

C. Codec Interoperability Issues

We expect endpoints participating in a DT-Talkie sessions to be heterogeneous and so their respectively supported codecs are likely to differ.[4] In regular interactive voice communication as well as in PoC, codecs are negotiate during session setup; this takes at least one extra RTT. As one-way delays in DTN scenarios may be significant, we have to avoid such

[2]A bundle is identified by its originating node's singleton EID, the DTN timestamp, the payload length, and the fragment offset.

[3]The sender knows this as the user presses a button to start/end a message.

[4]E.g., codecs requiring licensing may not be available on some platforms.

extra round-trips and yet provide some basic facilities for interoperable codec selection. We suggest using a combination of three simple mechanisms:

Baseline Encoding: We choose 8-bit G.711 PCM encoding at 8 kHz sampling rate as a baseline as this is most likely to be supported by most nodes since there is virtually no computational overhead for encoding or decoding besides a table lookup for A- or μ-law encoding. We recognize that PCM is about one order of magnitude more expensive in bandwidth consumption than other VoIP codecs. Therefore, we consider additional means for interoperability discussed below.

Multiple Encodings: A node A sending to another B for the first time has no knowledge about the codecs supported by B. For this first transmission, A could encode a voice message separately using all of its supported codecs and send them in one compound message. Recursive multipart MIME encoding (using *multipart/alternative*) allows for this. However, A may support many codecs and using all of them may not be a good idea because the resulting large message may experience poorer delivery performance. Hence, we need to use a minimal number of codecs so that we achieve an acceptable probability of interoperability.

Assuming two devices A and B choose from a set $C = C_1, ..., C_n$ of codecs, $n = |C|$; A and B each support k codecs, $C_A, C_B \subset C$; and they pick their codecs independently. We are interested in the probability $p_n(k)$ that $C_A \cap C_B \neq \emptyset$. With n codecs, there are a $N_s = \binom{n}{k}$ different sets and $N_d = \binom{n-k}{k}$ disjoint sets, yielding $p_n(k) = 1 - \frac{N_d}{N_s}$. For example, for $n = 20$,[5] $p_{20}(4) = 0.62$, $p_{20}(5) = 0.81$, $p_{20}(6) = 0.92$, $p_{20}(7) = 0.978$, $p_{20}(8) = 0.996$, and $p_{20}(9) > 0.999$. We find for $k > \sqrt{n}$, $p_n(k)$ quickly approaches 1 with larger k and verified that $p_n(\lceil\sqrt{n}\rceil) > 0.67$ for up to $n = 100$. This effect is even more pronounced with a non-uniform distribution (as one would expect for codecs).

To illustrate the impact of the number of codecs on the delivery performance, we conduct some simulations using the simple random waypoint (RWP) mobility model and epidemic routing (for the other parameters, see section IV). The results are plotted in figure 3. The message size grows with the number of codecs.[6] For up to 3 codecs, the impact on message delivery rate and delay is small: 5% drop in delivery rate and 10 min increase in delay. For 4–6 codecs, the impact becomes more noticeable. Even though the chances for success increase with non-uniform distributions for 3 codecs or less, a mechanism is needed to ensure interoperability *after* the first message exchange.

Implicit negotiation: We introduce an additional MIME extension header, borrowed from HTTP, *Supported:*, to indicate the (audio) codecs supported by an endpoint using a comma-separated list of audio MIME types. A DT-Talkie sender A includes its supported codecs (in the order of preference) in each message sent to a peer. The peers store the list and

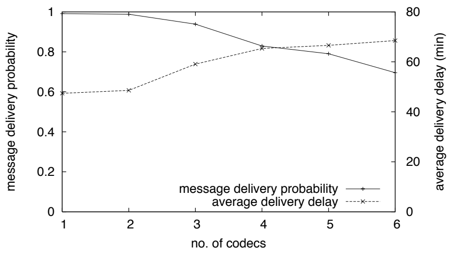


Fig. 3. Message delivery probability and average delay

remember it for future messages for the same and future conversations.[7] An error message is sent if a node B has received only encodings that it does not support; this message includes a *Supported* header so that the originator learns about B's capabilities. This mechanism is also usable in group conversations: nodes then pick the minimal number of codecs so that all recipients can decode the message.

In the resulting operation, users can agree upon a common (set of) codec(s) in the first message exchange. If a baseline codec C_1 supported by many platforms exists, the voice messages can be encoded in the baseline format C_1 and sent to B along with A's supported codecs. B may use a common codec from A's list for its reply in which it includes its own capabilities, using C_1 if no other common ones are available.

In the absence of a common baseline, assuming that node A supports codecs C_1, C_2, ... C_k, A chooses, e.g., $j = 3$ codecs out of these to encode a voice message m for the first interaction with B (likely including the most widespread codecs) and generates a compound message $\{C_1(m), C_2(m), C_3(m)\}$. In addition, the node stores m locally. After receiving the compound message, node B checks if the message includes an encoding it supports and, if so, plays back the message. Otherwise B returns an error message with its supported codecs; A can then use the previously stored message and resend it in an encoding understood by B. In this worst case, both nodes will require once an additional round-trip, assuming that they have any codec in common.

D. Protocol Overhead

Using MIME encoding provides flexibility but also introduces overhead as does the bundle protocol. While we could be more efficient by combining fields or shortening header names, we leave these optimizations for further study. At this point, we are only interested in a rough overhead assessment.

Per MIME message, we use six message-level headers, two per segment, and two per fragment. One additional per-source or per-message header is added when using group communication. We assume based upon our implementation: 32-byte DTN EIDs and 24-byte MIME separators, some 12 bytes per audio MIME type (e.g., *audio/G729* plus separator), 10-byte timestamps, and 2-byte fragment and sequence numbers, for

[5]Various sources, e.g., http://www.voip-info.org/wiki/view/Codecs list some 15–20 codecs used in VoIP systems, so 20 appears reasonably conservative.

[6]We choose the commonly used VoIP codecs including G.723.1, G.726, G.728, G.729, GSM, iLBC to model the message sizes.

[7]It may be advisable to include a device identifier so that the same user may be associated with different capability sets depending on the device she is using at a given point in time; we leave this for further study.

two-party sessions. This yields an overhead of some 400 bytes per-message (indicating support for 10 codecs), another 80 bytes per segment, and (optionally) 32 bytes per fragment, including all CRLF header and body separators.

Format	Len	#	Payload	Headers	MIME	Total
Statement	5 s	1	5 kB	401 B	8.0%	31%
Statement	15 s	1	15 kB	401 B	2.7%	14%
Segments	5 s	2	4 kB	480 B	12.0%	40%
Segments	15 s	5	11 kB	717 B	6.5%	20%
Fragments	5 s	2	4 kB	866 B	21.7%	68%
Fragments	15 s	5	11 kB	2165 B	19.7%	61%

TABLE I
PROTOCOL OVERHEAD FOR G.729-ENCODED VOICE

Table I summarizes the overhead for two different statement lengths (5 and 15 s) assuming a single encoding (G.729 at 8 kbit/s) when sent as a continuous *statement* and as a sequence of talkspurts (#: 2 or 5), either in *segments* within a single message or *fragments* spread across multiple messages. For the latter cases, we assume talkspurt length of 2–3 s and silence periods of 1 s (leading to less audio data). On top, the bundle protocol adds some 100 bytes headers per message and, when assuming TCP/IP, the TCP convergence layers adds minimal header overhead per bundle, but causes an extra exchange for setup and teardown as does TCP, and finally TCP and IP headers (including typical options) need to be considered. The table shows the approximate overhead for MIME and in total, assuming one bundle being exchanged per TCP and convergence layer connection (worst case).

We see that the overhead can vary a lot and become quite significant for short statements and when using voice activity detection and possibly fragmentation. For comparison, an interactive VoIP call using G.729 would, using a packetization of 20–100 ms generate 20–100 bytes payload plus 40 bytes headers (RTP/UDP/IP) per packet, yielding 40–200% overhead. Hence, the overhead appears acceptable. Overall, the above suggests that fragmentation be avoided unless nodes are close by and communication capacity is sufficient so that they can benefit from better interactivity.

IV. EVALUATION

To evaluate DT-Talkie, we use the Opportunistic Network Environment (ONE) simulator [19]. We run simulations using different mobility patterns: three synthetic models—Random Waypoint (RWP), the Manhattan-grid-style Helsinki City Scenario (HCS) [19], and the Working Day Movement model (WDM) [20] approximating some aspects of daily routines in urban areas—and a real-world mobility trace (TaxiTrace) that tracks cabs in San Francisco [21].

We choose two classes of multi-copy routing: epidemic [22] that performs flooding without any limitation on message replication, and binary Spray-and-Wait (SW) [23] that limits the number of copies to a fixed maximum (we use 16 copies). The mobile devices have up to 100 MB free buffer space for storing and forwarding messages. Communication takes place using bidirectional links at 2 Mbit/s. We conduct 10 simulation runs for each combination of parameters and report on the mean results; we calculated 95% confidence intervals, but those would be barely visible in the plots.

Traffic Generator: We implement a traffic generator for ONE assuming a G.729 codec to produce DT-Talkie style voice messages. They can be full-length voice messages (*message* mode) or fragments thereof (*fragmentation* mode). Talkspurt and silence periods are generated so that their durations follow a Pareto distribution as suggested in [24]. We choose voice message, fragment and silence durations in the range of 5–15 s, 2–3 s and 1–2 s respectively, that are drawn from the Pareto distribution. Each node generates a new voice message every 5 simulation minutes.

Destination Node Selection: Generally, the destinations are chosen randomly from anywhere in the simulation area. In addition, we are interested in the DT-Talkie performance if the two nodes are within some shorter distance—assuming that users will utilize voice messaging when they assume their peers to be "within reach". We define the *geographical distance* d for a particular node (e.g., 200 m) and then limit the choice of a random target to other nodes within d.

Codec diversity: We carry out most of our simulations assuming a common baseline codec. In addition, one simulation series investigates the performance impact when facing heterogeneous nodes. In this setup, each node randomly chooses support for $k = 6$ out of $n = 20$ codecs[8] uniformly distributed. For each voice message, a node picks a random destination from within $d = 200m$ and chooses to encode the message in $j = 1, ..., 6$ different codecs. Nodes do not learn codecs supported by other nodes from previous interactions which gives us a worst case assessment. We simulate codec diversity only for the *message* mode.

Performance Metrics: We consider mainly two performance metrics to assess the DT-Talkie operation in the *message* and *fragmentation* modes: the *delivery probability* p is measured as the number of unique messages received divided by the total number of messages sent; *delivery delay* is calculated as the interval between message generation at the source and its reception at the destination. A message comprising n fragments is considered delivered as soon as $n - 1$ fragments are received. Both metrics are reported for unidirectional voice messages.

Finally, we analyze the performance of sessions comprising multiple voice messages exchanged in a conversation between two peers. We define the *session completion rate* calculated as a number of sessions completed (i.e., all messages sent in this session were received) over the number of sessions created and the *session completion time* as the time to complete a session. These metrics are studied only in the *message* mode.

[8]AMR (7.4kbps), BroadVoice Codec (16kbps), CELP (4.8kbps), GIPS Family (13.3kbps), GSM (13kbps), iLBC (15kbps), G.711 (64kbps), G.722 (48kbps), G.722.1 (24kbps), G.722.1C (32kbps), G.722.2 (6kbps), G.723.1 (5.3kbps), G.726 (16kbps), G.726 (24kbps), G.726 (32, 40kbps), G.728 (16kbps), G.729 (8kbps), LPC10 (2.5kbps), and Speex (2.2kbps).

A. Simulation Scenarios

We use mostly sparse scenarios approximating different types of users and activities during a day for 6 hours simulation time, with 2 hours warmup time for the node buffers to reach steady state, and 2 hours cooldown time to allow for the delivery of already sent messages. The communication range of each mobile node in the simulation environment is 10 m (Bluetooth), except for the taxi trace using 100 m (WLAN).

RWP: We use RWP with 100 nodes moving as pedestrians in an area of $1{\times}1$ km (sparse) and $100{\times}100$ m (dense), approximating some outdoor and indoor convention space, respectively. The nodes move at random speeds of 0.5–1.5 m/s with pause times of 0–120 s, both uniformly distributed.

HCS: We simulate 126 mobile nodes moving as eager tourists in downtown Helsinki: 80 by foot, 40 by car, and 6 by tram. Each node moves with respectively realistic speed along the shortest paths between different points of interest (POIs) and random locations. The nodes are divided into four different groups, each with different POIs and probabilities to choose a next POI or a random place.

WDM: We model 543 persons in Helsinki following their daily sleep, work, and leisure routines, shown to approximate real-world contact characteristics. The scenario is based on section 5 in [20], with the number of nodes reduced from 1029 to 543 by shrinking all the group sizes about evenly so that the basic contact characteristics remain the same.

TaxiTrace: We finally choose a real-world scenario with detailed position information: a GPS-based mobility trace of the taxi cabs in San Francisco. The main data set contains GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay Area. For the purpose of our studies, we pick 317 cabs tracked over a period of 6 hours.

B. Simulation Observations

Our simulations are divided into six groups: delivery probability and mean delivery delay are analyzed in the first four, session completion rate and the mean session completion time in the fifth group, all for sparse scenarios. The sixth scenario investigates all metrics for the dense scenario.

Group 1: We select destination nodes randomly from anywhere in the simulation area and set the hop-count limit to 10. The results are plotted in figure 4 as a function of the time-to-live (*ttl*). We see that delay tolerance pays off but delays of 60 min or more to achieve $p > 0.5$ may limit the use of voice messaging. WDM (not shown) yields $p < 0.05$ as most nodes do not leave their offices for most of the day.

Across all *ttl* values, Epidemic and SW routing protocols perform best in the HCS scenario (featuring best connectivity) for both delivery probability and delay; expectedly, performance improves with connectivity and SW routing performs better for both metrics than Epidemic in most scenarios due to lower overhead. The only exception is the high mobility taxi trace for messaging (especially for short *ttl*): the larger radio ranges and motion patterns seem to support quick and broad spreading which helps delivery of full messages. Fragments, in contrast, may easily get dispersed into different directions as nodes move faster making recovery of $n-1$ of them difficult.

Full-length messages have better chances of delivery and experience lower delays than fragmented ones across all mobility scenarios and routing protocols. This holds consistently across the first three groups and is in line with [17].

Group 2: Destination nodes are chosen as above, *ttl* is fixed at 120 min, and the hop count is varied (see figure 5). When using binary SW with 16 copies, forwarding is de-facto limited to 4 hops. While SW behaves as expected, a higher hop-count limit improves Epidemic routing performance for messages across all scenarios: the load is low enough and contacts are sufficiently short so that broader flooding helps delivery. For fragments, the gain in p is less pronounced and delays barely improve. Overall, a hop-count limit seems advisable: with SW, 4 hops yield reasonable performance and, for Epidemic, the marginal gain (in p) starts diminishing above 6 hops.

Group 3: Figure 6 shows the results of varying the maximum distance d between source and sink. While RWP shows virtually no changes (presumably due to the truly random movement), the other (more structured) mobility models exhibit a slight dependency on d, albeit less pronounced than one might expect. We observed the most significant impact for WDM, with p(50m)=0.5 and p(200m)=0.3: in both cases, destinations were more frequently picked from with the same or nearby offices, suggesting that localized communication is well workable—which we will explore further in group 6.

Group 4: Figure 7 summarizes the impact of heterogeneous nodes supporting each $k = 6$ different codecs and choosing a subset of j codecs to send messages. Messages are sent with *ttl*=120 min and a hop-count limit of 10. In this figure, the delivery delay indicates the mean time passed from the initial message generation to the reception of a codec understood by the recipient; this includes possibly returning an error message and subsequent retransmission with a suitable codec. The figure clearly shows that the delivery rate increases with the number of codecs included and that the mean delay decreases. For scenarios with random short contacts as observed in RWP, the gain diminishes (for $j \geq 4$) as the message size becomes a limiting factor for successful forwarding during a contact.

Since voice messaging may succeed or fail at different stages, we further investigate the success rate for the first message, the returned error messages, and the success rate for the retransmitted messages. Note that all three messages may get lost. Table II summarizes our findings, showing for different j across all scenarios the fraction of voice messages successfully received after the first transmission (1) and after retransmission (2) as well as the fraction of messages that lead to returning errors (E).

As expected, with increasing number of codecs sent in the first message, the value of retransmissions diminishes and, for more than five codecs, nothing is gained anymore in our scenarios as either error messages or retransmissions are lost. We can also see that often less than half of the error messages (overall some 20–70%, except for $j = 6$ when this virtually only occurs for nodes supporting disjoint codec sets) lead to

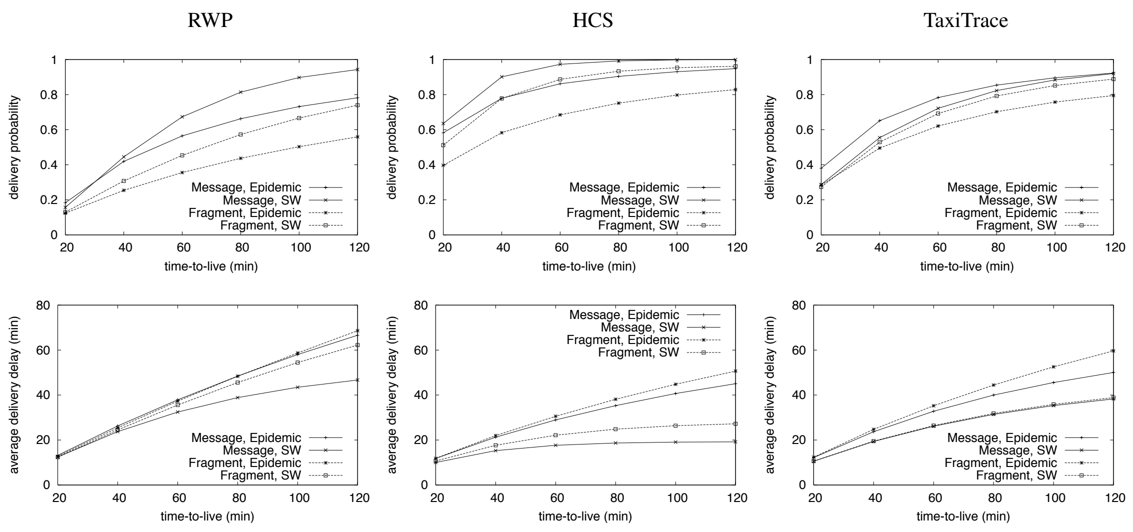


Fig. 4. Delivery probability and average delivery delay as a function of time-to-live (hop-count limit=10)

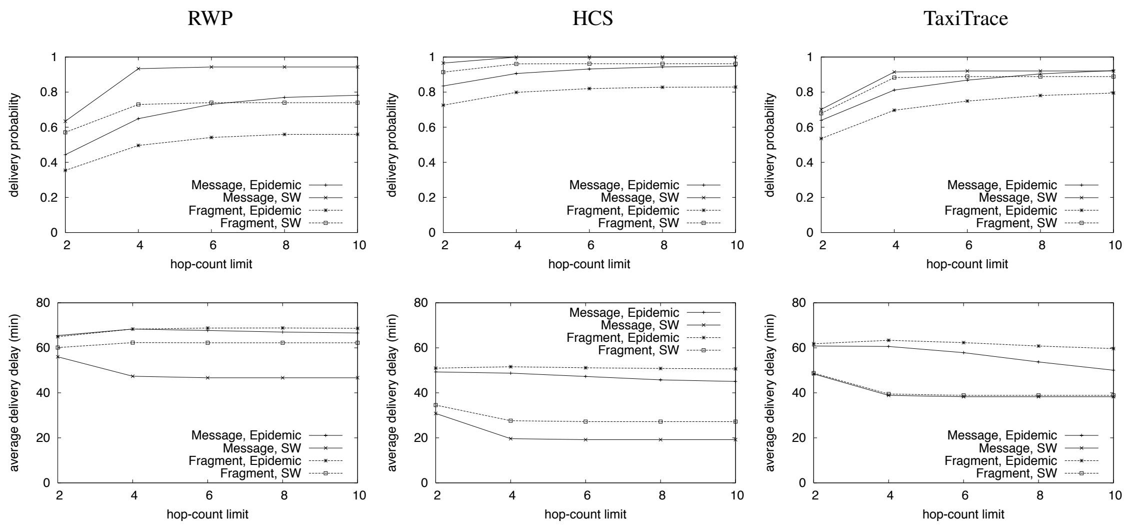


Fig. 5. Delivery probability and average delivery delay as a function of hop-count limit (*ttl*=120 min)

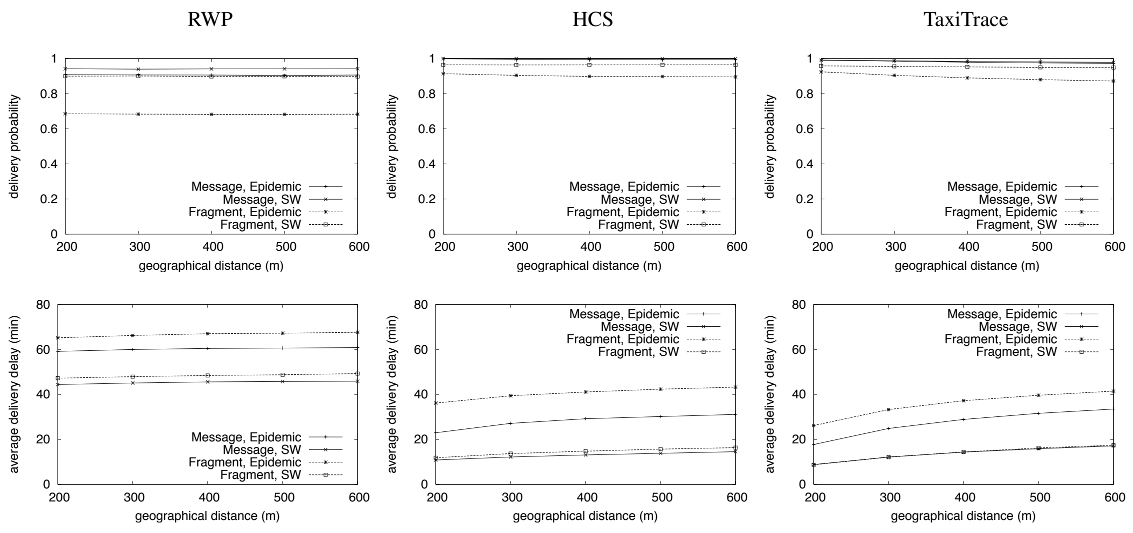


Fig. 6. Delivery probability and average delivery delay as a function of d (hop-count limit=10, ttl=120 min)

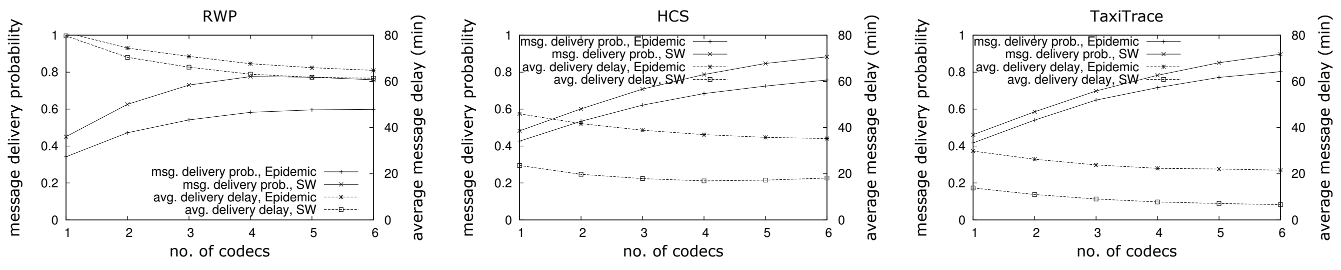


Fig. 7. Message delivery rate and delay as a function of the number of codecs included (hop-count limit=10, *ttl* = 120 min, $d = 200$ m)

success after retransmission. Moreover, in most cases, the joint success rate of the first and second transmission with j codecs is less than or about equal to the success rate of the first transmission only with $j + 1$ codecs. RWP with short random contacts using SW is borderline for $j \geq 3$. This suggests that a reasonable strategy would be to include a modest number of codecs ($j = 5$ in our scenarios) and revert to retransmissions only as last resort—which seems intuitive anyway given the potential delays. Given that nodes would store capabilities of their peers after the first message exchange, also scenarios with short contact durations could be served well.

Comparing these results to figure 6, we find that heterogeneity clearly has an impact on the success rate of voice message exchanges, but also that the proposed approach is capable of mitigating the impact to a large extent across all scenarios.

No. of codecs		1	2	3	4	5	6
RWP, Epidemic	1	20.2	36.4	46.8	54.3	58.0	60.0
	2	14.0	10.8	7.3	4.0	1.7	0.0
	E	47.4	33.2	22.6	14.2	8.7	5.2
RWP, SW	1	25.7	46.9	62.3	71.6	75.1	75.9
	2	19.4	15.8	10.7	6.0	2.1	0.0
	E	60.3	43.2	28.9	18.6	11.0	6.4
HCS, Epidemic	1	17.7	34.4	49.2	60.7	69.2	75.7
	2	24.9	19.1	13.0	7.7	3.2	0.0
	E	41.7	31.8	22.9	15.9	10.5	6.4
HCS, SW	1	19.6	37.5	54.6	69.0	80.4	88.3
	2	28.6	22.7	16.2	9.7	4.3	0.0
	E	45.1	34.9	25.5	18.0	11.9	7.6
TaxiTrace, Epidemic	1	16.4	34.8	51.9	63.9	73.8	80.3
	2	25.3	19.3	13.0	7.7	3.3	0.0
	E	40.9	31.9	23.0	16.5	10.3	6.3
TaxiTrace, SW	1	17.6	36.3	54.3	68.9	81.0	90.0
	2	28.6	22.3	15.5	9.4	4.2	0.0
	E	41.9	33.0	24.5	17.5	11.5	7.2

TABLE II
SUCCESS RATES (%) OF VOICE MESSAGES AFTER THE FIRST (1) AND SECOND (2) TRANSMISSION AND FRACTION (%) OF MESSAGES FOR WHICH CODEC MISMATCH ERRORS (E) WERE GENERATED.

Group 5: In figure 8, we compare session completion rates and average times as a function of the number of interactions between a pair of close-by nodes. We find that p decreases steadily with a growing number of interactions for most mobility scenarios, the structured ones (HCS and TaxiTrace) performing better than RWP and the TaxiTrace with higher mobility better than the less dynamic HCS. Epidemic routing is significantly inferior in all cases. Despite the short distances, only short conversations (4–5 messages) have a chance of completing with $p > 0.8$, and a mean session completion time of some 50 min does not yield much interactivity.

Group 6: The above findings motivate investigating a dense scenario such as an exhibition hall, with many people moving around in somewhat randomly, e.g., in a break, for which we provide a crude first approximation by RWP. Figure 9 confirms that acceptable performance is achievable for sufficiently co-located nodes: About 99% of the messages are delivered via 3 hops maximum for *ttl*=10 min using both (Epidemic and SW); SW yields shorter delivery delays ($\sim$1.5 min) than Epidemic ($\sim$3.4 min). We use *ttl*=10 min and hop-count limit 3 for simulating sessions with multiple interactions. SW achieves a session completion rate of more than 95% across all numbers of interactions, while Epidemic (due to more overhead) shows decreasing performance (from 92% for 2 interactions to 70% for 6). For SW, the sessions are reasonably interactive with four message exchanges completing within 5 min whereas Epidemic takes about three times as long.

Overall, we find voice messaging workable under different conditions if the nodes are sufficiently close to one another relative to a scenario's node mobility and density. Especially for reasonably reliable sessions involving multiple interactions, users should be within a confined space with sufficient message carriers around—in line with our use cases. Our findings suggest using Spray-and-Wait routing with a limited total number of messages over epidemic routing with hop-count limit. We expect that, when targeting short delivery delays, more sophisticated history-based routing protocols may not provide much extra gain; but this remains for further study.

V. IMPLEMENTATION

We have implemented DT-Talkie for the Maemo and Symbian mobile software platforms. For both, we use platform-specific Bundle Protocol and TCP convergence layer implementations on top of IP for inter-device communication. As mobile devices have different screen sizes and input methods even if they run on a same mobile OS platform, the DT-Talkie code is split into the application logic (engine) and UI components so that the implementations are portable beneath the UI layer. Only the latter needs to be adapted to different-sized touchscreen and non-touchscreen devices. Figure 10 depicts the component model of implementation architecture.

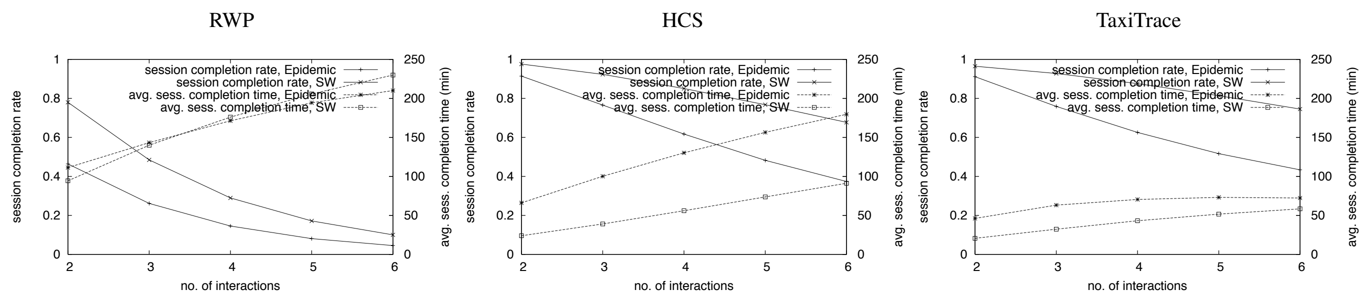


Fig. 8. Session completion performance as a function of # interactions (hop-count limit=10, *ttl* = 120 min, *d*=200 m)

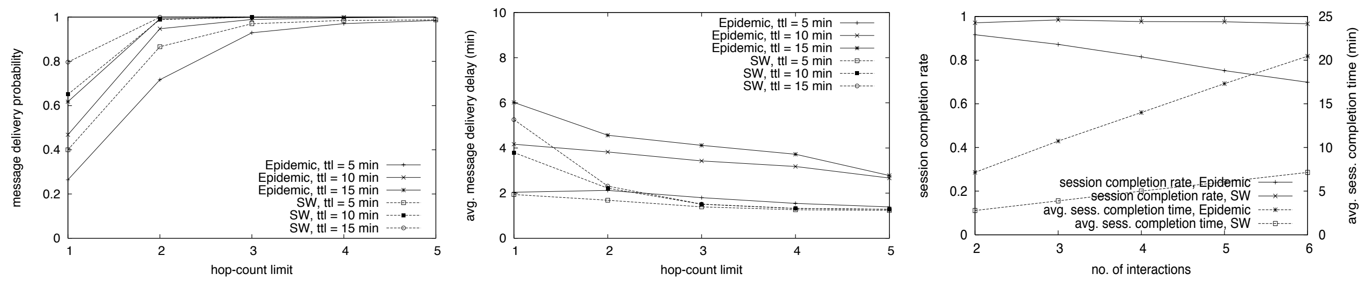


Fig. 9. RWP scenario (area size: 100 m × 100 m)

A. *Maemo implementation*

On the Maemo platform, we use the DTN2 reference implementation for Bundle Protocol services, the GTK+ framework for implementing user interface components, the GStreamer framework for audio recording and playback, and the GMIME framework for creating and parsing MIME messages.[9] The target devices of the Maemo implementation are Nokia N810 and N900 Internet tablets (with touchscreen). Figure 11 shows a screen shot of the UI. As Linux-based implementation, DT-Talkie is also portable to Linux PCs, Openmoko, MacOS X.

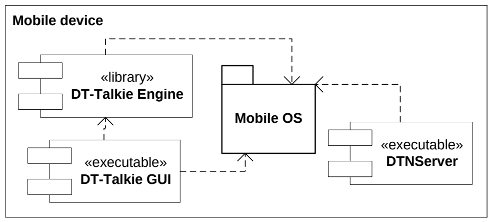
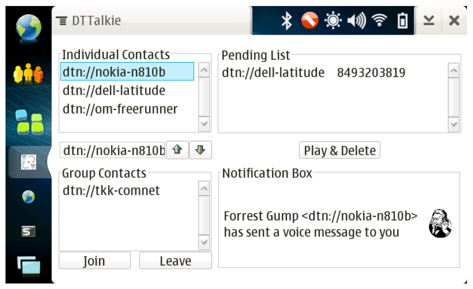


Fig. 10. Architecture of DT-Talkie

B. *Symbian implementation*

We have developed a native Symbian C++ bundle implementation (DTNS60) of RFC5050 [15] and the TCP convergence layer. To provide a robust system architecture, DTNS60 has been implemented using the client-server framework offered by Symbian OS. It conforms to the microkernel architecture and enables multitasking of applications, i.e., multiple delay-tolerant applications can run in parallel (and along with other applications) on the device using the same daemon process as asynchronous service provider.

DT-Talkie runs as one application, originally designed for the S60-based Nokia N95 and E90 mobile phones (no touchscreen). The user interface of the Symbian-specific DT-Talkie uses the Avkon GUI framework [25]. We use active objects for concurrent processing and event-driven programming for energy-efficient operations. We incorporated the multimedia framework[10] for audio recording and playback, but wrote the MIME functionality ourselves.



Forrest Gump <dtn://nokia-n810b> has sent a voice message to you

Fig. 11. UI of DT-Talkie for Maemo

C. *Experimental Validation*

As experimental validation of the voice messaging for mobile DTNs, we use Maemo and Symbian implementations to deliver voice messages between heterogeneous devices (E90 and N95 smartphones, and N810 and N900 Internet

[9]http://www.dtnrg.org/Code, http://www.gtk.org. http://www.gstreamer.net, and http://spruce.sourceforge.net/gmime, respectively.

[10]http://forum.nokia.com

tablets), optionally with a Linux laptop (running only DTN2) as a bundle forwarder. This validation targets heterogeneous devices instead of complex routing setups. To simulate DTN behavior at the link layer, we create artificial disruptions by walking with a device out of the communication range of the other or turning off one device for some time.

We use static or epidemic routing due to present limitations of DTN2, and pre-configured PCM, MP3 and G.729 audio codecs for encoding voice messages. Bluetooth or WLAN are used as link layer technologies; devices running DTN2 use the Bluetooth and Bonjour node discovery mechanisms. The clocks of the devices need to be synchronized in the order of minutes (which is easy to achieve using operator time or occasional NTP) and use a message timeout of one hour. This requirement is an artifact of the Bundle Protocol's use of time.

The DT-Talkie operation follows the steps described in section 3.1. Voice messages are sent by choosing an individual or group contact in the UI (figure 11) and using the touchscreen or hardware buttons to start and stop recording, after which the message is sent automatically. The receiver sees the sender of an incoming message and, after listening, may reply using the session mode or choose a different party to talk to.

We experimented with two party and group communication scenarios using all four device types as senders and receivers, demonstrated on various occasions (e.g., [9]). While communication between peers was mostly direct, we also carried out simple multihop experiments with a single forwarder. We also validated the implementations in more complex multihop settings by connecting them to the ONE simulator's convergence layer interface as well as to our internal DTN testbed.

VI. Related work

Since DT-Talkie offers a PTT-style communication service, we mainly discuss the previous works related to PTT.

Wu et al. [26] have described a design and implementation of an OMA-specified PoC client for mobile users in cellular network or WLAN. Parthasarathy [27] presents a prototype implementation of a Push-to-talk server in the Internet. Kim et al. [28] have provided an OMA-compliant PoC solution for packet-switched networks accessed via GPRS/UMTS or WLAN technology. Raktale [29] proposes and evaluates a 3GPP architecture for an efficient implementation of the PoC services in a 3G packet-switched network. Cruz et al. [30] describes a PTT over IMS solution designed with a Talk Burst Control Protocol based on SIP messages for call session control, and test their solution with a high bandwidth LAN and a CDMA2000 wireless network. Blum et al. [31] presents a concept of extending PTT to Push-to-MultiMedia (PTM) to allow other media types (e.g., video) and integrating this PTT/PTM functionality in community-based services using the IMS architecture. Hsu et al. [32] design a context-aware PTT service with the combination of the PTT features and context-aware service. O'Regan et al. [33] implement and evaluate a SIP-based PTT service for a 3G network using the 3GPP UMTS Release 5/6 IMS specification.

Ronnholm [34] presents an outline for a push-to-talk system over Bluetooth, which is independent of the cellular networks. Lin et al. [35] have proposed a peer-to-peer PTT service over distributed operator-independent network environments that does not rely on the functionality provided by the underlying mobile networks. Gan et al. [36] propose a distributed PTT system for the Intelligent Transportation Systems environment. Chang et al. [37] have designed and implemented the PTT mechanism in an ad-hoc VoIP network, in which the PTT server and the user agent combined with the pseudo SIP server provide the PTT service without the support of the standalone SIP server. Hafslund et al. [38] have implemented and tested a solution for PTT voice group communication in mobile ad-hoc networks, which reuses the optimized flooding techniques from the OLSR protocol (relying on a connected network).

Furthermore, DTN-based asynchronous voice communication has also been subject to recent research. Honicky et al. [6] propose the idea of using a mobile phone primarily as a voice message device focusing on asynchronous communication, and they outline the potential benefits of switching to an asynchronous model, even with infrastructure. Heimerl et al. [7] extend the previous idea [6] by developing a prototype cellphone system with voice messaging and explore its value for Ugandan users via trial deployments. Scholl et al. [8] present issues related to the development of store-and-forward VoIP based rural Telemedicine networks, and advocate to build such networks on the basis of DTN.

VII. Conclusion

In this paper, we have presented asynchronous voice messaging as a mechanism for interaction in opportunistic networks between heterogeneous devices and its cross-platform implementation *DT-Talkie*. We have seen that message-based communication following the DTN paradigms can offer a feasible communication platform as long as the usage scenarios and users are sufficiently delay-tolerant or the user populations are dense enough. The latter is likely to be the case for many of the scenarios introduced in section II; also, connectivity may be assisted by WLAN access points inside a given facility and even across a city [39] which we will explore further. In any case, using voice messaging, while introducing extra delay, decouples sender and receivers, eliminates the need for a (stable) end-to-end path, and improves voice fidelity in the presence of packet losses by means of hop-by-hop reliability.

Our quantitative findings indicate that it seems preferable not to fragment messages unless the nodes are reasonably well connected to each other and reducing latency brings benefits in interactivity. It is advisable to include a small number of codecs in a message to maximize the delivery rate and minimize delay. In our simulation scenarios, we find that Binary Spray-and-Wait with a fixed message count (e.g., 16) provides a suitable and simple routing scheme for DT-Talkie when distances between nodes are limited and messages can be short-lived. Our future work includes investigating a broader range of scenarios (including more realistic indoor models) and more diverse conversation settings (including groups).

User requirements and acceptance probably constitute the biggest issues. While our research prototypes provide an initial user interface design, usability and user expectation studies will be needed to determine how to offer the functionality to users to gain acceptance. At least, a seamless integration of the DT-Talkie interface with the address book, call history, and call control functions of a mobile phone will be required to offer a coherent presentation to the user and minimize usage efforts. Providing hints when to choose which mode of operation may be another interesting step. Besides improving the user interface integration, we are working on DT-Talkie implementations for the Android and iPhone platforms.

Acknowledgments

The authors would like to thank Philip Ginzboorg for contributing his insights on the codec matching analysis.

This work was supported by TEKES as part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT) and by Teknologiateollisuus ry within the REDI project.

References

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

[2] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot, “Haggle: A Networking Architecture Designed Around Mobile Users,” in *Proceedings of the IFIP WONS*, 2006.

[3] Omar Mukhtar and Jörg Ott, “Backup and Bypass: Introducing DTN-based Ad-hoc Networking to Mobile Phones,” in *Proceedings of the ACM REALMAN*, 2006.

[4] Avri Doria, Maria Uden, and Durga Prasad Pandey, “Providing connectivity to the Saami nomadic community,” in *Proc. 2nd Int’l Conference on Open Collaborative Design for Sustainable Development*, 2002.

[5] Simone Leggio, *A Decentralized Session Management Framework for Heterogeneous Ad-Hoc and Fixed Networks*, Ph.D. thesis, University of Helsinki, Finland, 2007.

[6] R.J. Honicky, Omar Bakr, Michael Demmer, and Eric Brewer, “A message oriented phone system for low cost connectivity,” in *Proc. 6th Workshop on Hot Topics in Networks*, 2007.

[7] Kurtis Heimerl, RJ Honicky, Eric Brewer, and Tapan Parikh, “Message Phone: A User Study and Analysis of Asynchronous Messaging in Rural Uganda,” in *Proc. ACM Workshop on Networked Systems for Developing Regions*, 2009.

[8] Jeremiah Scholl, Lambros Lambrinos, and Anders Lindgren, “Rural telemedicine networks using store-and-forward Voice-over-IP,” *Stud Health Technol. Inform.*, vol. 150, pp. 448–452, 2009.

[9] Md. Tarikul Islam, Anssi Turkulainen, Teemu Kärkkäinen, Mikko Pitkänen, and Jörg Ott, “Practical Voice Communications in Challenged Networks,” in *Proceedings of the ExtremeCom workshop*, 2009.

[10] Jörg Ott, Dirk Kutscher, and Christoph Dwertmann, “Integrating DTN and MANET Routing,” in *Proceedings of the ACM SIGCOMM Workshop on Challenged Networks*, 2006.

[11] John Whitbeck and Vania Conan, “HYMAD: Hybrid DTN-MANET Routing for Dense and Highly Dynamic Wireless Networks,” in *Proceedings of the 3rd IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications*, 2009.

[12] Marina Petrova, Lili Wu, Matthias Wellens, and Petri Mähnen, “Hop of No Return: Practical Limitations of Wireless Multi-Hop Networking,” in *Proceedings of the ACM REALMAN*, 2005.

[13] Kevin Fall, “A Delay-Tolerant Network Architecture for Challenged Internets,” in *Proceedings of the ACM SIGCOMM*, 2003.

[14] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H.Weiss, “Delay-Tolerant Network Architecture,” RFC 4838, 2007.

[15] Keith Scott and Scott Burleigh, “Bundle Protocol Specification,” RFC 5050, November 2007.

[16] Kenneth P. Birman and Thomas A. Joseph, “Reliable communication in the presence of failures,” *ACM Transactions on Computer Systems*, vol. 5, no. 1, pp. 47–76, February 1987.

[17] Mikko Juhani Pitkänen, Ari Keränen, and Jörg Ott, “Message Fragmentation in Opportunistic DTNs,” in *Proc. 2nd WoWMoM Workshop on Autonomic and Opportunistic Communications*, 2008.

[18] David D. Clark and David L. Tennenhouse, “Architectural Considerations for a new Generation of Protocols,” in *Proceedings of the ACM SIGCOMM*, 1990.

[19] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen, “The ONE Simulator for DTN Protocol Evaluation,” in *Proc. 2nd International Conference on Simulation Tools and Techniques*. 2009, ICST.

[20] Frans Ekman, Ari Keranen, Jouni Karvo, and Jörg Ott, “Working Day Movement Model,” in *Proc. 1st SIGMOBILE Workshop on Mobility Models for Networking Research*, 2008.

[21] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser, “CRAWDAD data set epfl/mobility (v. 2009-02-24),” Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, Feb. 2009.

[22] A. Vahdat and D. Becker, “Epidemic routing for partially connected ad hoc networks,” Technical Report CS-200006, Duke University, April 2000.

[23] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra, “Spray and wait: an efficient routing scheme for intermittently connected mobile networks,” in *Proceeding of the ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.

[24] Trang Dinh Dang, Balazs Sonkoly, and Sandor Molnar, “Fractal Analysis and Modeling of VoIP Traffic,” in *Proceedings of Networks*, 2004.

[25] Leigh Edwards, Richard Barker, and Staff, *Developing Series 60 Applications: A Guide for Symbian OS C++ Developers (Nokia Mobile Developer Series)*, Addison-Wesley Professional, March 2004.

[26] 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*, vol. 7, no. 5, pp. 539–552, 2007.

[27] A. Parthasarathy, “Push to talk over cellular (PoC) server,” in *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, 2005.

[28] P. Kim, A. Balazs, E. van den Brock, G. Kieselinann, and W. Bohm, “IMS-based push-to-talk over GPRS/UMTS,” in *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2005.

[29] S.K. Raktale, “3PoC: an architecture for enabling push to talk services in 3GPP networks,” in *Proc. IEEE International Conference on Personal Wireless Communications*, 2005.

[30] Rui Santos Cruz, Mario Serafim Nunes, Guido Varatojo, and Luis Reis, “Push-to-Talk in IMS Mobile Environment.,” in *ICNS*, Jaime Lloret Mauri, Vicente Casares Giner, Rafael Tomas, Tomeu Serra, and Oana Dini, Eds. 2009, pp. 389–395, IEEE Computer Society.

[31] N. Blum and T. Magedanz, “PTT + IMS = PTM - towards community/presence-based IMS multimedia services,” in *Proc. 7th IEEE International Symp. on Multimedia*, 2005.

[32] Jenq-Muh Hsu, Wei-Bin Lain, and Jui-Chih Liang, “A Context-Aware Push-to-Talk Service,” in *Proc. 2nd International Conference on Multimedia and Ubiquitous Engineering*, 2008.

[33] Eoin O’Regan and Dirk Pesch, “Performance Estimation of a SIP based Push-to-Talk Service for 3G Networks,” in *Proceedings of the 5th European Wireless Conference*, 2004.

[34] V. Ronnholm, “Push-to-Talk over Bluetooth,” in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, 2006.

[35] Jiun-Ren Lin, Ai-Chun Pang, and Yung-Chi Wang, “iPTT: peer-to-peer push-to-talk for VoIP,” *Wireless Communications and Mobile Computing*, vol. 8, no. 10, pp. 1331–1343, 2008.

[36] Chai-Hien Gan and Yi-Bing Lin, “Push-to-Talk Service for Intelligent Transportation Systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 391–399, 2007.

[37] L.-H. Chang, C.-H. Sung, H.-C. Chu, and J.-J. Liaw, “Design and implementation of the push-to-talk service in ad hoc VoIP network,” *IET Communications*, vol. 3, no. 5, pp. 740–751, 2009.

[38] A. Hafslund, Toan Tuan Hoang, and O. Kure, “Push-to-talk applications in mobile ad hoc networks,” in *Proceedings of the 61st IEEE Vehicular Technology Conference*, 2005.

[39] Mikko Pitkänen, Teemu Kärkkäinen, and Jörg Ott, “Opportunistic Web Access via WLAN Hotspots,” in *Proceedings of IEEE PerCom*, March 2010.