Data Broadcasting using Mobile FM Radio: Design, Realization and Application

Hang Yu[1], Ahmad Rahmati[1], Ardalan Amiri Sani[1], Lin Zhong[1], Jehan Wickramasuriya[2], Venu Vasudevan[2]

[1]Department of Electrical & Computer Engineering
Rice University, Houston, Texas, USA
{hang.yu, rahmati, ardalan, lzhong}@rice.edu

[2]Betaworks, Applied Research Center
Motorola Mobility, Libertyville, Illinois, USA
{jehan, venu}@motorola.com

ABSTRACT
In this work, we offer a novel system, *MicroStation* (*μStation)* that allows ubiquitous data broadcasting applications using the FM radio on mobile devices such as smartphones. μStation includes two key modules to enable data broadcasting based on existing mobile FM radio hardware. *Channel Selector* assigns different FM channels to neighboring μStation broadcasters to avoid collision and guides μStation listeners to find their broadcasting of interest. *Data Codec* realizes bit-level communication between mobile devices through existing FM radio hardware. We describe an implementation of μStation on the Nokia N900 smartphone, and provide low-level APIs and services to support application development. We also demonstrate two representative applications: *Facebook-FM* and *Sync-Flash*. These applications demonstrate the capability of μStation to readily enable a new class of ubiquitous data broadcasting applications on mobile devices.

Author Keywords
FM radio, data broadcasting, mobile devices.

ACM Classification Keywords
C2.0 Computer-Communication Networks: Data communications

General Terms
Design, Experimentation, Measurement.

INTRODUCTION

Peer-to-peer (P2P) and location-based mobile applications have long been an important focus of the ubiquitous computing community. A few recent indicative applications include *Color* [1] which allows a user to share photos with others in their vicinity, and *Cisco StadiumVision* [2] which delivers location-based content and service for sports venues. The proliferation of such applications requires power-efficient and overhead-free mobile broadcasting technologies.

In this work, we exploit the FM radio that is increasingly available on mobile devices to provide a *data broadcasting* system for P2P and location-based applications. We are motivated by a recent and important hardware trend on mobile devices: due to its continuously reduced cost and integration with other wireless technologies (e.g., Wi-Fi and Bluetooth) on a single chipset, the FM radio is becoming increasingly available on mobile devices such as smartphones, tablets and media players. Testimony to this trend includes the hardware requirements of the Windows Phone 7 Platform, which embraces the FM radio [3]. As another example, the Apple iPhone and iPod Touch already incorporate the FM receiver and transmitter hardware, and the software to enable them is reportedly under development [4]. The mobile FM receiver and transmitter are intended to allow users to listen to the broadcasted programs, and stream music to short-range home and automobile stereos, respectively. However, we go beyond such intended uses of the mobile FM radio and enable data broadcasting applications based on existing hardware.

We reveal two challenges toward enabling practical and deployable data broadcasting applications. First, simultaneous broadcasters must not collide. Therefore, they should coordinate with each other to use the available FM channels while remaining quickly identifiable by interested listeners. Unlike radios in white space, FM radio has a much shorter range (~5 meters) and multiple orthogonal channels so that the solution requires special treatment toward the hidden-node problem and channel allocation. Second, the solution must not modify the FM radio hardware, but has to rely on the device's audio interface to realize data broadcasting. Therefore, binary data must be properly converted into audio, and the broadcaster and listener must be symbol-synchronized using software.

We address these challenges with μStation, a software solution that operates without modification to device

UbiComp'11, September 17–21, 2011, Beijing, China.

hardware, operating system, and the FM radio driver. µStation has two key modules: *Channel Selector* and *Data Codec*. First, Channel Selector enables automatic and efficient FM channel allocation, so that nearby broadcasters use orthogonal channels with high probability. Meanwhile, it ensures listeners to find the channel with broadcasting of interest timely, by letting broadcasters periodically announce their presence. Second, Data Codec performs conversion between binary data and audio to compatibly work with the FM radio hardware. More importantly, it effectively achieves symbol synchronization in software by leveraging different sampling frequencies in the FM transmitter digital-to-analog converter (DAC) and FM receiver analog-to-digital converter (ADC).

We have implemented µStation on the Nokia N900 smartphone and provided both low-level APIs and services to support a large set of mobile applications. We have also developed two representative applications using µStation to demonstrate its effectiveness. First, *Facebook-FM* broadcasts a user's Facebook ID so that nearby users can load the corresponding Facebook profile to facilitate social interaction. Second, *Sync-Flash* realizes a synchronized flashing pattern using many smartphone screens from users physically close to each other, e.g., the audience of a concert or a sports game.

As the first publicly reported system that aims to enable ubiquitous data broadcasting applications using existing mobile FM radio hardware, µStation has offered the FM radio, which possibly obtains little attention before, significant potential to facilitate the interaction between physically adjacent mobile devices. We hope our initial endeavor with µStation can enlighten more innovative ways to leverage the FM radio as an important hardware feature on current and emerging mobile devices.

BACKGROUND OF MOBILE FM RADIO

We next provide background of the mobile FM radio.

Spectrum Usage

The spectrum usage of FM radio falls into the UHF band and spans from 87.5 MHz to 107.9 MHz in the U.S. In this case, the spectrum is equally divided into 103 orthogonal channels, each of which is 200 KHz apart from its neighbors. A FM channel can be identified by its center frequency, or by a unique number (e.g., Channel #1 for the channel at 87.5 MHz). In each FM channel, FM radio uses 15 KHz and 30 KHz bandwidth for broadcasting monaural and stereo audios respectively. The remaining bandwidth is used for sideband applications such as the Radio Broadcast Data System (RBDS otherwise known as RDS in Europe) [5], and the currently obsolete Microsoft DirectBand [6].

Both licensed and unlicensed FM broadcasting use the same set of channels. Licensed broadcasting usually comes from radio stations owned by commercial organizations or educational institutions, with a fixed coverage and schedule.

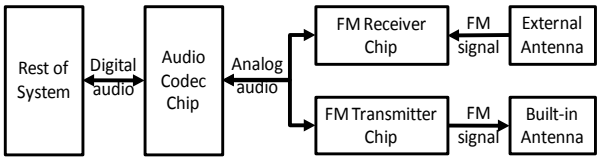

Figure 1: FM radio hardware architecture on mobile devices

As a result, given time and location, licensed broadcastings are often highly predictable, as indicated by [7].

Regulations for Unlicensed Broadcasting

Most countries allow unlicensed FM broadcasting. The US FCC regulates the output power and mandates that the field strength of any unlicensed FM emissions must not exceed 250 microvolt/meter at 3 meters, leading to a maximum output power of approximately 15 nW. The EU regulation is similarly enacted but the maximum allowed output power is 50 nW. Australia allows output power as high as 10 µW, but only if the unlicensed transmission does not interfere with present licensed broadcastings. We note that a few countries such as China do not yet allow unlicensed FM broadcasting; mobile devices used in these countries usually have the FM transmitter disabled. Due to our implementation and experimental constraints, we work under the US FCC regulations.

We define available FM channels in the following way: *a FM channel is available if the channel RSSI under licensed broadcasting is below certain threshold* (e.g., -110dBm in our implementation). We note that a FM channel can be available even with present licensed broadcastings, as long as the licensed broadcasting is sufficiently weak. As a result, checking the availability of a FM channel can be simply done by measuring the channel RSSI, which is supported by mobile FM radio. Our RSSI based definition ensures the quality of unlicensed broadcastings, yet there can be alternative ways to define available FM channels such as using the human perceivable quality of licensed broadcasting. Nonetheless, µStation is agnostic of the definition of channel availability, as we will show later.

Hardware Architecture

We show the hardware architecture of mobile FM radio in Figure 1. We note that the FM transmitter and receiver may be in the same or separate integrated circuits, e.g., the Nokia N900 uses the Silicon Lab Si4713 for the transmitter and Broadcom BCM2048 for the receiver, while the iPhone 4 utilizes the Broadcom BCM4329 including both the transmitter and receiver. Furthermore, some FM radio chipsets have an internal DAC for converting digital audio into analog audio, and an ADC for the opposite conversion. Other chipsets only provide an analog audio interface; an audio codec chip provides the necessary ADC and DAC, and effectively bridges the FM radio hardware with the rest of the system through a digital audio interface. µStation is independent on these implementation issues by interfacing with the digital audio interface provided by either the FM

Figure 2: The headphone, earphone and the winding case used in our experiments

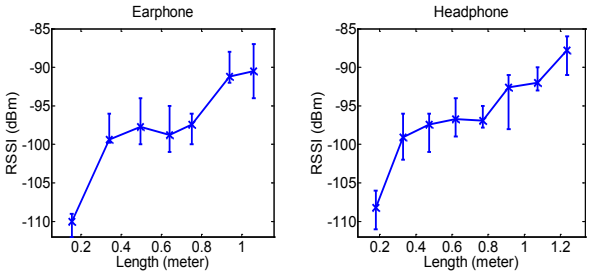

Figure 3: RSSI with different cable lengths of the earphone and headphone

radio chipsets or the audio codec. For generality, we use the term *FM radio hardware* to refer to the FM transmitter, FM receiver and audio codec if existing.

Current mobile FM transmitters and receivers employ different antennas. To achieve small antenna loss, the antenna length should be at least a tenth of the carrier wavelength, i.e., about 30 cm for FM radio. Noticeably, this is too big for mobile devices such as smartphone. As a result, the transmitter simply uses a small antenna integrated in the chip, which is sufficient to provide the low permissible output power. The receiver, on the other hand, usually relies on an earphone or headphone as the external antenna, to maintain small antenna loss. But we note that recent FM receiver realizations such as [8] can work with a receive antenna that is built inside the mobile device.

Digital Audio

Digital audio has two important properties, the sampling frequency f_s and the quantization resolution m. The audio can be either monaural or stereo. Stereo audio uses twice the bandwidth as a monaural audio does, i.e., 30 KHz and 15 KHz in each FM channel respectively.

The DAC and ADC in the FM radio hardware can work with any sampling frequency below their maximum, e.g., 96 KHz in the N900. For FM transmission, if the sampling frequency of the digital audio is lower than the maximum, the DAC will perform up-sampling to the input digital audio while maintaining the bandwidth of the output analog audio. For FM reception, the digital audio output always has the maximal sampling frequency but one can further perform down-sampling using software. µStation exploits these properties of the DAC and ADC to address the symbol synchronization challenge.

Other Data Broadcasting Standards

FM radio has a set of benefits compared to other current and emerging wireless technologies supporting broadcasting, such as Wi-Fi, Bluetooth, ZigBee and ANT [9]. First, FM radio is inherently connection-free, unlike all the other technologies that require device pairing. This feature of FM radio prevents extra overhead in connection establishment and maintenance, and thus makes FM radio perfectly fit mobile applications that are short-lived and impromptu. Second, FM radio is power-efficient compared to other technologies especially Wi-Fi, due to its much lower output power under unlicensed regulation. The power efficiency allows an always-on treatment of the FM radio on battery-constrained mobile devices. Third, due to the short range and low frequency nature of mobile FM radio, it can provide an accurate indication of physical proximity with minimum influence from environments. The combination of all the features makes FM radio an outstanding choice for one-way data broadcasting as the focus of this work. For example, to broadcast one's socializing profile as one walks around, device pairing not only is unnecessary but also degrades the throughput significantly; meanwhile broadcasting in Wi-Fi's range would allow a much smaller number of simultaneous broadcastings due to interference.

We note that Bluetooth can be alternatively used for broadcasting without device paring, by presenting the message as the device ID in the Bluetooth beacon that is periodically broadcasted. However, this non-standard use of Bluetooth has two limitations. First, each message has a limited length, as the ID is limited to 128 bits. Second, the achievable data rate is limited, e.g., around 1 Kbps for Bluetooth vs. up to 20 Kbps for µStation. The second limitation is made worse when there are multiple Bluetooth broadcasting devices since they have to share the medium for broadcasting beacons.

CHARACTERIZATION OF MOBILE FM RADIO

The mobile FM radio has several distinctive features compared to conventional FM radio from licensed stations, such as limited output power and the use of external antenna for receiving. Since it is relatively new to mobile devices, to the best of our knowledge, there is no published work about its characteristics. Therefore, we next experimentally evaluate the properties of the mobile FM receiver and transmitter with real-life settings, based on the FM radio chipsets in the Nokia N900.

FM Reception

As mentioned earlier, mobile FM receivers employ an external antenna such as a headphone or an earphone. Our measurements show that the choice of the headphone or earphone, and its cable length (within a certain range) has little impact on the quality of FM reception.

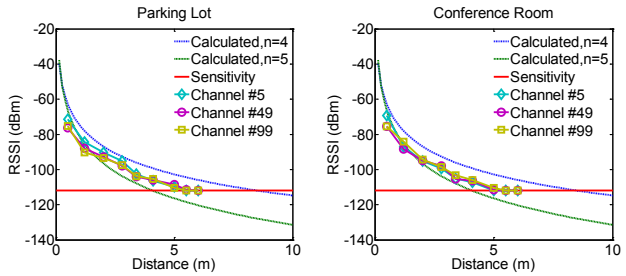

Figure 4: Broadcast range of mobile FM radio

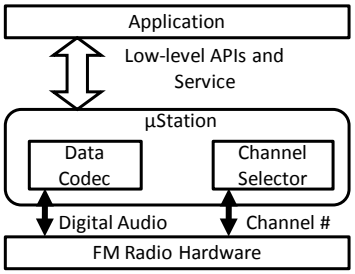

Figure 5: Overview of µStation

Headphones and earphones use a soft cable, which may advertently or inadvertently become coiled. Therefore, we examine the impact of cable length on FM reception quality with real-life settings. To quantify the impact, we measure the received RSSI at the FM receiver first with a 1.9 meters Lenovo P550 headphone, and then with a 1.6 meters Nokia N900 earphone. We use a winding case to adjust the cable length in a controlled manner, as shown in Figure 2. To ensure a fair comparison, we keep other settings fixed, including the output power (15 nW) and the distance between the transmitter and receiver (2.5 meters).

As shown in Figure 3, the RSSI is highly correlated with the length of the headphone/earphone cable. We further make three important observations. First, the RSSI drops with a smaller cable length due to higher antenna loss, and there is a significant drop when the length falls below 20 cm. Second, the two tested antennas exhibit similar characteristics. Last and most importantly, there exists a certain range of the cable length within which the RSSI is approximately constant, e.g., 0.4-0.8 meter. Noticeably, such range is often encountered in real-life situations, e.g., when a user is using a smartphone while wearing an earphone/headphone. Based on these observations, we conclude that the mobile FM reception quality will be relatively independent on the receive antenna. Therefore, for the experiments in the remainder of the paper, we use a 0.5 meter earphone as the receive antenna.

FM Transmission

The broadcast range of the mobile FM radio is important to applications. Our measurements show that the broadcast range is approximately five to six meters under regulation, regardless of the environment or channel number.

We chose two environments to evaluate the broadcast range. The first one is an empty campus parking lot with a line-of-sight (LOS) path and the second one is a conference room with people inside sitting around a table without any LOS path. These two environments have little and rich multipath effect respectively. We performed experiments for three available FM channels at the center and far ends of the spectrum: channel #5, #49, and #99. To see the broadcast range, we measure the RSSI with different distances between the transmitter and receiver. We also compare our measurements with that under a theoretical path loss model $P_r=P_t\cdot[\lambda/(4\pi d)]^n$ where P_t and P_r are the transmit and received power, λ the wavelength, n the power decay factor of the environment, and d the distance.

Our measurement results are summarized in Figure 4. We can see that the broadcast range is limited to about 5-6 meters. The environment and channel number have little impact on the range, due to the good penetration property of FM radio and the closeness of FM channels in the spectrum. The results also indicate a power decay factor between 4 and 5 for the path loss of FM radio.

AN OVERVIEW OF µSTATION

µStation leverages existing FM radio hardware to allow mobile devices with data broadcasting and reception, as the *µStation broadcaster* and *µStation listener* respectively. We assume that each mobile device is, at any given time, either a broadcaster or a listener. Application can choose and quickly switch between these two roles at will. Figure 5 provides an overview of µStation: for both broadcasters and listeners, µStation interacts with the FM radio hardware and presents an interface to applications with low-level APIs and services. There are two key modules in µStation. Channel Selector uses one control channel and multiple data channels so that with a high probability, no two nearby broadcasters will use the same data channel. Furthermore, Channel Selector guarantees listeners to find their broadcasting of interest in a timely fashion, even when devices are mobile. Data Codec converts binary data into an audio stream for transmission and vice versa for reception, in order to compatibly work with the FM radio hardware. Data Codec meanwhile achieves symbol-synchronization between the broadcaster and listener, solely in software.

µSTATION CHANNEL SELECTOR

µStation can be considered as a special incarnation of the extensively researched cognitive radio systems in which collision between coexistent users has to be resolved [10]. µStation Channel Selector enables multiple unlicensed and equally important broadcasters to efficiently share multiple orthogonal FM channels. While apparently the most efficient way to avoid collision is to let the broadcasters use orthogonal channels, achieving it is nontrivial.

µStation Channel Selector is based on one *control channel* and multiple *data channels*, all of which are available FM channels. First, all µStation broadcasters share the control channel. On the control channel, a broadcaster either

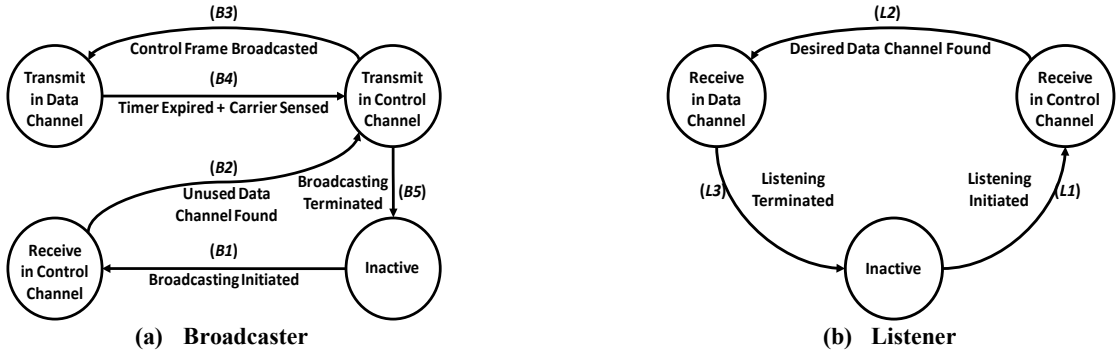

Figure 6: State transition diagrams of µStation Channel Selector

periodically announces which data channel it is using, or finds an unused data channel by listening to the announcement from other broadcasters. Second, only one broadcaster can use each data channel to broadcast application data. Any listener that is interested in the application on a data channel can receive the broadcasting based on the periodical announcement on the control channel.

The control channel enables quick discovery of the data channel usage by broadcasters and their applications. This is important since each device can only listen to one channel at any time. The control channel allows broadcasters to indicate their presence on a single channel to all potential listeners. However, in order to efficiently leverage the spectrum resource of available FM channels, application data broadcasting should utilize individual data channels. Without the control channel, it would take listeners much longer time to scan all the channels to find the right one.

Channel Transition

We illustrate the operation of Channel Selector using state transition diagrams shown in Figure 6. There are four possible states for a broadcaster and three for a listener. We next elaborate the state transitions.

Control Channel

When an application starts, the broadcaster or listener goes from inactive to listening to the control channel (*B1* and *L1*). This is important for two reasons. First, new broadcasters need to monitor the control channel in order to identify an unused data channel for their broadcasting (*B2*). Second, listeners similarly need to monitor the control channel to find the desired data channel (*L2*). Once a broadcaster has identified its data channel, it periodically sends a control frame containing its data channel number, its device ID, and its application ID. This enables listeners to find their data channels of interest by receiving the control frame.

Since multiple broadcasters use the same control channel, we adopt a *Carrier Sense Limited Access* (*CSLA*) mechanism for broadcasters to share the control channel. CSLA avoids collision between broadcasters with carrier sensing, and meanwhile limits the channel usage by each broadcaster to a certain percentage. We note that CSLA is intrinsically similar to p-persistent carrier sense multiple access (CSMA) [11]. We choose not to use 1-persistent CSMA [11] because the notorious hidden-node problem of 1-persistent CSMA may happen frequently due to the short range nature of the mobile FM radio. In accordance to CSLA, a broadcaster announces its presence on the control channel periodically if and only if (*i*) the control channel is free (carrier sensing by checking the RSSI), and (*ii*) it has not broadcasted on the control channel for a certain duration (two seconds in our design).

Data Channel

At any time, a data channel can be occupied by only one broadcaster. The broadcaster can continuously broadcast on its data channel, but must periodically switch to the control channel to announce its presence, as discussed previously (*B3* and *B4*). We leave the decision on how often the broadcaster should utilize the control channel to individual applications, as long as their usage follows the CSLA rule. When the application ends, the broadcaster releases its data channel by stopping broadcasting control frames on the control channel and going back into inactive (*B5*). A listener finds a desired broadcast by listening to the control channel. The listener then tunes to the desired data channel for receiving (*L2*), until the application is terminated (*L3*).

Collision Performance

We next provide an analysis of the collision performance of Channel Selector. That is, we analyze the probability for a listener to successfully receive its broadcasting of interest, with the presence of hidden broadcasters. For simplicity, we assume N broadcasters are in the range of a listener, and K of them are hidden from each other (i.e., the hidden broadcasters cannot hear each other and therefore may collide). We also assume the control frame on the control channel lasts for t seconds, and each broadcaster is allowed to broadcast on the control channel once every T seconds. In this case, when the control channel utilization is small, we can approximate the collision probability on the control channel as:

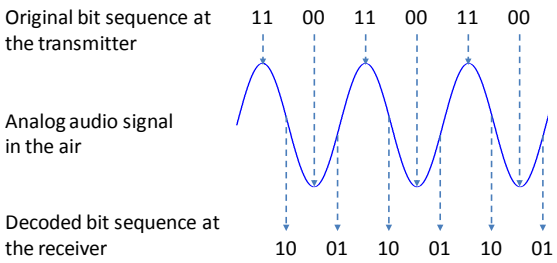

Figure 7: An example of the original and decoded bits without symbol synchronization

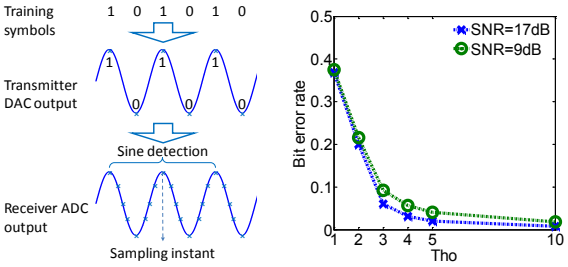

Figure 8: (Left) Symbol synchronization by µStation Data Codec (Right) Bit error rate over different ρ

$$P_{Control} \cong \frac{2Kt}{T}.$$

When a collision occurs on the control channel, the listener cannot decode the control frame from all of the affected broadcasters.

For the data channels, the hidden node problem appears in a different fashion. Since the hidden broadcasters cannot hear each other, there is a chance that more than one broadcaster is using the same data channel. As a result, the number of unused channels for the K hidden broadcasters is M'=M-(N-K), where M is the total number of data channels. In this case, the probability that more than one broadcaster is using the same data channel is

$$P_{Data} = 1 - \frac{(M')!}{(M'-K)!} \cdot \frac{1}{(M')^K}.$$

From the perspective of listeners, the probability of successfully receiving its desired broadcasting is

$$P = 1 - \frac{K}{N}(P_{Control} + (1 - P_{Control}) \cdot P_{Data}).$$

Assuming the following parameters that are typical in real-life scenarios and adopted in our design

t=0.05, T=2, N=10, K=3, M=20,

we can calculate P=90%. Therefore, Channel Selector ensures successful reception by listeners with a high chance.

Channel Convergence

For Channel Selector to operate correctly, we must ensure that all µStation devices in a particular geographical area converge on the same control channel and have identical lists of data channels. The convergence is easily achievable if all µStation devices have the same set of available FM channels. In this case, the FM channel with the smallest channel number can be treated as the control channel and others as data channels.

In order to achieve channel convergence, we leverage an important property of FM channels. That is, the FM channel availability is time and location-dependent and more importantly, highly predictable, since licensed radio stations often follow a fixed broadcasting schedule. As a result, one can simply construct a mapping from time and location to available FM channels offline, and store it in a remote server which can be accessed in real time. We highlight that the location dependency of FM channel availability is coarse-grained due to the large coverage of licensed broadcastings; a device will usually see the same list of available FM channels even when it moves over certain kilometers. Therefore, localization based on cell towers is more than sufficiently accurate, and device can even locally store the mapping with minimal cost. Additionally, the device only needs to store the mapping with locations where it is used.

µSTATION DATA CODEC

Data Codec leverages the existing FM radio hardware to realize bit-level communication between two µStation devices. Here let us consider a single pair of transmitter and receiver. We use the name "transmitter" and "receiver" to generally refer to µStation devices that are transmitting and receiving respectively, but note that a broadcaster needs to both transmit and receive, while a listener only needs to receive. Data Codec interfaces with both the application and the FM radio hardware, by applying conversion between digital data and audio. While the conversion may sound straightforward, Data Codec must properly cope with *symbol (clock) synchronization*, a particular challenge to µStation. By addressing the symbol synchronization challenge with proper design, Data Codec achieves an effective bit rate of up to 10 Kbps for monaural audio.

Bits-Audio Conversion

We use pulse amplitude modulation (PAM) to convert binary bits into audio. PAM is natural and straightforward for our application: one only needs to specify the sampling frequency f_s and quantization resolution m of the audio. Then, m bits are jointly represented as one audio sample with 2^m possible amplitudes, and the audio is streamed to the FM radio hardware with a sampling frequency of f_s. Converting an audio stream to binary bits requires the same procedure in reverse, but encounters an additional challenge, symbol (clock) synchronization, which we will tackle in the next subsection.

Choosing f_s should follow the Nyquist sampling theorem, i.e., f_s can be no greater than twice the available analog audio bandwidth, or f_s≤30 KHz. Similar to other modulation techniques in digital communication, a larger m will render

higher bit error rates, thereby lower reliability. Based on our experimental results, under typical distance between the transmitter and receiver, such as 2.5 meters with a RSSI of -100 dBm, m=2 can offer a good reliability, i.e., a bit error rate of less than 5%.

Symbol Synchronization

Current mobile FM radio is not intended for digital communication; therefore it lacks the dedicated hardware for symbol synchronization, which is critical to PAM. For a receiver to correctly recover the broadcasted bits, the receiver ADC must be symbol-synchronized to the transmitter DAC. That is, the receiver must perform analog-to-digital conversion of the received audio at the right instant. Figure 7 shows an example when the receiver ADC and transmitter DAC are not symbol-synchronized. In the example, a bit sequence of "110011001100" is transmitted with m=2, i.e., each symbol representing two bits. Due to up-sampling, a sine signal will be the analog output of the transmitter DAC. Because the receiver ADC is not symbol-synchronized, it samples the received analog signal with an offset, decoding bits incorrectly.

To achieve symbol synchronization without hardware support, we explore the properties of the DAC and ADC in the FM radio hardware. Similar to the oversampling technique in conventional digital communication but in the software domain, we let the receiver ADC adopt a much higher sampling frequency than that of the transmitter DAC. As a result, the receiver can analyze the approximated analog audio. To achieve symbol synchronization, a transmitter sends a pre-determined sequence of training symbols with alternative symbol "0" and symbol "1" with a sampling frequency of $f_{s,t}$. Due to up-sampling to the input digital signal by the DAC, a sine signal with frequency $0.5f_{s,t}$ will be the analog output and transmitted through FM. On the other hand, the receiver expects a sine signal with its peak corresponding to the symbol "1" in the training symbol sequence. Importantly, the peak also indicates the correct sampling time. The receiver employs a sampling frequency $f_{s,r}$ much higher than $f_{s,t}$ so that the received analog signal can be approximately recovered. Figure 8 (Left) illustrates the symbol synchronization procedure: after getting the samples, the receiver seeks to detect the training symbols which exhibit a sine pattern, and then simply searches for the peak sample and treats it as the right sampling instant.

Noticeably, the ratio of $f_{s,r}$ over $f_{s,t}$, or $\rho=f_{s,r}/f_{s,t}$ determines the symbol synchronization accuracy. In our design, we adopt ρ=5, which is identified experimentally. That is, we measure the bit error rate performance of the broadcasting under different ρ, shown in Figure 8 (Right). We repeat the measurements with two different channel SNRs: 17 dB and 9 dB. One can clearly see that a larger ρ can reduce the bit error rate and the reduction is not linear, i.e., after a certain threshold, e.g., ρ=5, the reduction is trivial. This is because the biggest offset from the right sampling instant is $T_s/2\rho$

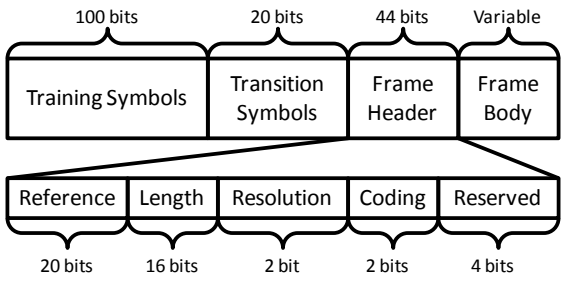

Figure 9: Framing in µStation Data Codec

where T_s is the signal period. As a result, the benefit of increasing ρ decreases with larger ρ, and the BER is mainly affected by channel SNR.

Given ρ=5, the concrete values of $f_{s,r}$ and $f_{s,t}$ in our design are $f_{s,r}$=50 KHz and $f_{s,t}$=10 KHz respectively. While $f_{s,r}$ can be as high as the maximum sampling rate of the receiver ADC, i.e., 96 KHz in N900, we found that it is not only unnecessary, but also hardly achievable. This is due to the insufficient computing capacity of mobile devices for real-time processing, especially when the device is under a heavy workload. Our experimental experience reveals that a maximal $f_{s,r}$ of around 50 KHz can be achieved in N900 without lagging the decoded data from the received audio. It also indicates a $f_{s,t}$ of 10 KHz which is below the maximal sampling frequency allowed by the FM channel.

Framing

Similar to conventional digital communication, to ease bits delivery, Data Codec needs framing to organize binary bits. We next present the frame design. As shown in Figure 9, each Data Codec frame contains four parts: *training symbols*, *transition symbols*, *frame header* and *frame body*.

Training symbols are utilized for two purposes: one is to perform symbol synchronization as we explained previously, and the other is to provide reference amplitude of the symbols so that the rest of the frame can be correctly translated with different quantization resolutions.

Transition symbols partition the training symbols and frame header, in order to avoid the confusion of detecting the starting bit of frame.

Frame header includes various control information for the receiver to decode the bits. It is always translated with m=1 to maximize reliability.

Frame body carries actual data bits given by the application and can have various length.

As shown, the non-data parts of a frame contain 164 bits altogether, while the frame body is allowed to have arbitrary length from 1 to 2^{16}=65536 bits, depending on the application data. For connectionless broadcasting applications, the broadcaster is often continuously and repeatedly streaming data, and therefore the frame body usually has a much longer duration than the other parts do,

meaning that the actual application throughput can approach the raw bit rate of µStation.

Reliability Enhancement

µStation assumes one-way broadcasting. While an occasional frame loss and moderate BER might be acceptable for some applications, e.g., location-based services, they can be fatal for applications that have a tight accuracy requirement for data delivery, e.g., synchronization between devices. While the responsibility of maintaining reliability can be left to applications, Data Codec does employ coding to the raw binary bits to assist the application for reliability enhancement. We use linear coding with a rate of 1/2, which is simple and effective.

Bit Rate Performance

Given all the above techniques adopted by Data Codec and the parameters in our design, we can approximately calculate the effective bit rate as

$$R_{bit} = f_{s,t} \times m \times r_{coding},$$

which is 10 Kbps for monaural audio with $f_{s,t}$=10 KHz, m=2 and r_{coding}=1/2. For stereo audio, the achievable bit rate can be up to 20 Kbps.

IMPLEMENTATION

Our design of µStation compatibly works with existing mobile FM radio hardware, and it is largely device independent from an implementation perspective. To offer application developers both flexibility and ease of use, µStation provides both low-level APIs and services. In this section, we first explain the APIs and services, and then present our smartphone-based implementation of µStation.

µStation Low-level API

We encapsulate the basic functionality of µStation as a regular python library, which can be imported by applications and used on any mobile devices through the provided low-level APIs. The core APIs include:

channel_getlist(): list all available FM channels. µStation will gather location information, try to query the remote or local server, and obtain the list of available channels.

channel_find(): find an unused data channel for broadcasting. µStation will treat the device as a broadcaster, tune to the control channel to get a list of unused data channels, and randomly pick one among them.

channel_announce(): announce the application in the control channel. µStation will switch to the control channel and perform carrier sensing. If the medium is free and the broadcaster has not used the control channel in the last T seconds, µStation will broadcast a control frame with its data channel number, device ID and application ID.

channel_bond(app_ID, dev_ID): find the data channel occupied by app_ID and dev_ID. µStation will treat the device as a listener, and keep monitoring the control channel until finding the desired control frame. Then it will tune to the corresponding data channel.

broadcast(bi_data, ch_data): broadcast binary data bi_data in the data channel ch_data. µStation will generate a frame as appropriate, form it as audio, and stream the audio to the FM transmitter for broadcasting.

receive(bi_data[], ch_data): retrieve all the frames in the data channel ch_data and store the decoded binary data in bi_data[]. µStation will keep recording audio from the FM receiver and extract the data within all the frames.

µStation Service

µStation services are built on top of the low-level APIs to further ease application development. Here we show an example service, *inter-device synchronization*, which allows multiple devices to synchronize their system clocks with high accuracy (<0.1 seconds).

µStation leverages the short range property of FM radio to accomplish inter-device synchronization. That is, a "beacon" device as the broadcaster periodically broadcasts reference time signals through µStation. Seeing and then decoding the broadcasted reference time signal, surrounding devices as listeners can adjust their system clocks accordingly. This provides a high-level of accuracy, since the signal latency due to RF propagation is virtually non-existent at the FM radio range, and our measurements show that each device has a small and more importantly, constant signal detection latency. Our experiments show that the accuracy of inter-device synchronization can be less than 0.1 seconds while devices are in range.

Smartphone-based Implementation

To demonstrate the feasibility of µStation, we choose the Nokia N900 smartphone for implementation. The N900 adopts Maemo [12], which is open-source and allows rapid customization and application development. Nonetheless, µStation does not require any hardware, OS or FM radio driver modification.

The N900 implements both the FM transmitter and receiver as Video4Linux radio devices and controls them over the hardware bus I^2C. The FM transmitter integrates a DAC within the chip thereby directly interacts with the system; the FM receiver does not have an ADC and relies on the audio codec for interaction. Hardware configurations such as switching on and off the FM radio or changing the FM channel can be achieved through the standard Video4Linux APIs. The FM radio drivers also expose some information in a virtual file system called *sysfs*, allowing simple configurations such as setting the output power and measuring RSSI. The audio codec and FM transmitter in N900 do not enable very low-level interfacing, e.g., the user cannot directly send or get a digital audio as a binary stream. Nonetheless, µStation indirectly interacts with them using the following methods: for transmission, µStation plays the audio stream as a media which will be automatically

captured by the FM transmitter; for reception, µStation records the audio stream from the audio codec with proper sampling frequency, using the *PulseAudio* utility [13] available in Maemo.

Our smartphone-based implementation of µStation is limited by the FM radio hardware and its software interface. As a result, much of the computation of µStation has to be realized using the powerful Application Processor, which inevitably limits the power efficiency of µStation. However, given proper firmware support, µStation can be implemented in the microcontroller within the FM chipset to significantly improve its power efficiency.

APPLICATION

µStation supports a large set of data broadcasting applications on mobile devices, although the broadcast range is limited (5 meters) and the data rate is moderate (20 Kbps). For example, a user can broadcast its profile during a social gathering; a device can broadcast its sensing data to peer devices to support collaborative sensing services; a user can broadcast a product or service advertisement as he or she moves around and encounters other people. Next we present two example applications based on µStation: *Facebook-FM* and *Sync-Flash*.

Facebook-FM

Facebook-FM is a socializing application that allows automatic and rapid sharing of Facebook profiles among a group of users in a small area. To enable the sharing, a device as a broadcaster uses µStation to broadcast the Facebook ID of the user. All the other in-range devices as listeners can simultaneously receive the broadcasting. Each device can switch between broadcasters and listeners so that each user's Facebook ID is shared among others. This application can be particularly useful in public occasions such as a party or a conference to initiate interaction between attendees.

We have implemented Facebook-FM solely based on the µStation low-level APIs, as shown in Figure 10. For broadcasters, the application first calls *channel_getlist()* and *channel_find()* to reserve a data channel, and then repeatedly calls *channel_announce()* to indicate its presence and *broadcast()* to broadcast the user's Facebook ID in its data channel. For listeners, the application first calls *channel_getlist()* and *channel_bond()* to find the data channel used by Facebook-FM. Then it calls *receive()* to retrieve the message for each data channel that is associated with Facebook-FM. The homepage associated with the Facebook ID is further shown in the browser.

Sync-Flash

Another innovative application we create using µStation is called *Sync-Flash*, which coordinates multiple mobile devices to synchronously "flash" using their regular LED flash or screen. It can be used in concerts or sports games where the audiences/fans would like to present certain flashing pattern using their smartphones. Another usage of Sync-Flash can be to provide multiple "slave" flashes for photography, in order to offer additional light or reduce shadows.

Broadcaster	Listener
[ch_control, ch_data_list] = *channel_getlist()* ch_data = *channel_find()* bi_data = *enc*("Facebook ID") while (true): *channel_announce()* *broadcast*(bi_data, ch_data) if (app terminated): break	[ch_control, ch_data_list] = *channel_getlist()* ch_data[] = *channel_bond* ("Facebook-FM", N/A) while (true): for ch in ch_data[]: *receive*(bi_data[], ch) if (app terminated): break

Figure 10: Realization of Facebook-FM using the low-level APIs of µStation

We demonstrate Sync-Flash using the concert example shown in Figure 11. Assuming there is a large population of audience and they are uniformly distributed, a flashing pattern can be realized by letting a subset of the audiences, e.g., those in the middle in Figure 11, turn on the screen of their smartphones. Clearly, all the intended audiences should be triggered at the same time and their flashings have to be in the same fashion, e.g., always on or flash every one second, to achieve certain intended pattern. Different fashions can be indicated by different pattern ID.

µStation realizes Sync-Flash using both the low-level APIs and the inter-device synchronization service. The realization involves three steps, as shown in Figure 12. First, a few audiences need to become the initiators, either voluntarily in real time by registration through Internet, or in advance by pre-assignment. The initiators are responsible for triggering the flash and broadcasting the intended flashing pattern. They also act as "beacon" devices for inter-device synchronization. The initiators can be remotely synchronized. Second, when a flashing pattern is desired, the intended initiators, depending on which subset of the audience should flash, broadcast a triggering message to their neighbors. The message has two uses: serving as the reference time signal for inter-device synchronization, and containing a pattern ID which the surrounding audience identify and flash as appropriate. Finally, seeing the triggering message from initiators, all involved audiences simultaneously flash in the same fashion, collectively offering a desired flashing pattern. The pattern stops when the initiator broadcasts a termination message.

RELATED WORK

To the best of our knowledge, µStation is the first work that leverages the mobile FM radio to enable data broadcasting applications. Other uses of mobile FM radio beyond its audio transmission and reception capability are exclusively limited to localization based applications, such as [14].

There exist several systems for radio stations to broadcast small amounts of application specific data alongside FM audio broadcasting, including RBDS [5] and Microsoft DirectBand [6]. Furthermore, the authors of [15] have

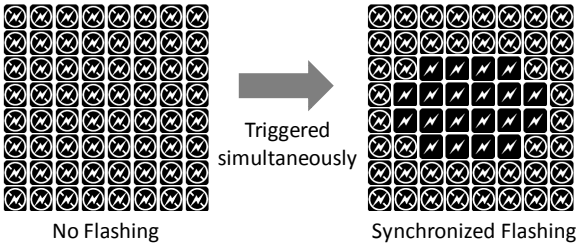

Figure 11: Sync-Flash performed by a concert audience

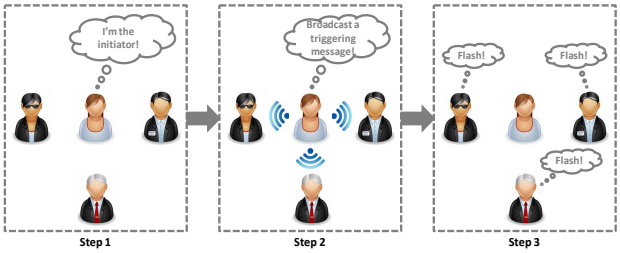

Figure 12: Three steps to realize Sync-Flash with µStation

proposed and evaluated a protocol for radio stations to transfer arbitrary files over RBDS. µStation is fundamentally different. First, µStation targets short-range data broadcasting from mobile devices, as opposed to metropolitan-area radio station broadcastings of RBDS. Second, µStation reuses the frequency band for analog audio broadcasting in each FM channel, while RBDS has its dedicated band allocation. Third and most importantly, µStation only works with the digital audio interface of the mobile FM radio, while the designers of RBDS have access to the radio baseband and are free to develop their own PHY protocols. Our work has its own challenges such as symbol synchronization.

Using voice channels for data communication is an existing concept, e.g., [16-20]. These works primarily focus on how to improve the communication performance while compatibly working with the voice codec in GSM systems. Our work, µStation, has a different focus and targets on data broadcasting from mobile devices, as well as the supported applications. Again, the use of FM radio exhibits its unique challenges such as FM channel selection.

CONCLUSION

In this work, we reported the design, realization and applications of µStation, a data broadcasting system using the mobile FM radio. µStation does not require any modification to device hardware, OS or FM radio driver, and supports efficient channel selection and transmitter-receiver symbol synchronization. µStation enables developers to easily implement a wide range of data broadcasting applications with a compact set of APIs and services, as demonstrated by Facebook-FM and Sync-Flash.

µStation departs from the traditional use of mobile FM radio for audio transmission and reception and is the first publicly reported exploration toward using it for short-range data broadcasting. While we have showed the feasibility of mobile FM radio for ubiquitous applications, its potential invites more serious efforts from both the research community and mobile hardware vendors.

REFERENCES

1. Color, *http://www.color.com/*.
2. Cisco StadiumVision, *http://www.cisco.com/web/strategy/sports/StadiumVision.html*.
3. Windows Phone 7 Series: Everything Is Different Now, *http://gizmodo.com/#!5471805/windows-phone-7-series-everything-is-different-now*.
4. 9 to 5 MAC: In-house Radio.app in the works for iPhone and iPod touch, *http://www.9to5mac.com/10167/In-house-Radio-app-in-the-works-for-iPhone-and-iPod-touch/*.
5. RBDS, *Radio Broadcast Data System*.
6. DirectBand, *Microsoft*.
7. SiriusXM Radio, *http://www.siriusxm.com/frequency*.
8. BlueCore BC7820, *http://www.csr.com/products/10/bluecore-bc7820*.
9. ANT, *http://www.thisisant.com/*.
10. S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications (JSAC),* 2005.
11. P. C. Gupta, *Data Communications and Computer Networks*: PHI Learning Pvt. Ltd., 2011.
12. Nokia Maemo, *http://maemo.org/*.
13. PulseAudio, *http://www.pulseaudio.org/*.
14. A. Matic, A. Papliatseyeu, V. Osmani, and O. Mayora-Ibarra, "Tuning to your position: FM radio based indoor localization with spontaneous recalibration," in *Proc. IEEE PerCom*, 2010.
15. A. Rahmati, L. Zhong, V. Vasudevan, J. Wickramasuriya, and D. Stewart, "Enabling pervasive mobile applications with the FM radio broadcast data system," in *Proc. ACM HotMobile*, 2010.
16. T. Chmayssani and G. Baudoin, "Data transmission over voice dedicated channels using digital modulations," in *Int. Conf. Radioelektronika*, 2008.
17. N. N. Katugampala, K. T. Al-Naimi, S. Villette, and A. M. Kondoz, "Real time data transmission over GSM voice channel for secure voice and data applications," in *Secure Mobile Communications Forum*, 2004.
18. C. K. LaDue, V. V. Sapozhnykov, and K. S. Fienberg, "A Data Modem for GSM Voice Channel," *IEEE Trans. Vehicular Technology,* 2008.
19. M. Rashidi, A. Sayadiyan, and P. Mowlaee, "A Harmonic Approach to Data Transmission over GSM Voice Channel," in *Proc. ICTTA*, 2008.
20. A. Dhananjay, A. Sharma, M. Paik, J. Chen, T. K. Kuppusamy, J. Li, and L. Subramanian, "Hermes: data transmission over unknown voice channels," in *Proc. ACM MobiCom*, 2010.