# Design and Evaluation of a Testbed for Mobile TV Networks

MOHAMED HEFEEDA, Simon Fraser University
CHENG-HSIN HSU, National Tsing Hua University



This article presents the design of a complete, open-source, testbed for broadcast networks that offer mobile TV services. Although basic architectures and protocols have been developed for such networks, detailed performance tuning and analysis are still needed, especially when these networks scale to serve many diverse TV channels to numerous subscribers. The detailed performance analysis could also motivate designing new protocols and algorithms for enhancing future mobile TV networks. Currently, many researchers evaluate the performance of mobile TV networks using simulation and/or theoretical modeling methods. These methods, while useful for early assessment, typically abstract away many necessary details of actual, fairly complex, networks. Therefore, an open-source platform for evaluating new ideas in a real mobile TV network is needed. This platform is currently not possible with commercial products, because they are sold as black boxes without the source code. In this article, we summarize our experiences in designing and implementing a testbed for mobile TV networks. We integrate off-the-shelf hardware components with carefully designed software modules to realize a scalable testbed that covers almost all aspects of real networks. We use our testbed to *empirically* analyze various performance aspects of mobile TV networks and validate/refute several claims made in the literature as well as discover/quantify multiple important performance tradeoffs.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]**:** Network Architecture and Design

General Terms: Design

Additional Key Words and Phrases: Mobile TV, DVB-H, testbed, energy saving, mobile multimedia, wireless streaming, channel switching delay, broadcast networks

**ACM Reference Format:**

Hefeeda, M. and Hsu, C.-H. 2012. Design and evaluation of a testbed for mobile TV networks. ACM Trans. Multimedia Comput. Commun. Appl. 8, 1, Article 3 (January 2012), 23 pages.
DOI = 10.1145/2071396.2071399 http://doi.acm.org/10.1145/2071396.2071399

## 1. INTRODUCTION

Many of the mobile devices currently available in the market are almost full-fledged computers with high resolution displays, fast network links, large memory and storage space, and fast processors. Thus, multimedia content can be rendered on most of these mobile devices. The capabilities of the mobile devices as well as the users' desire to access multimedia content anywhere and anytime have created a strong demand for mobile TV services. Market research anticipates several hundred

This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the British Columbia Innovation Council (BCIC), and Nokia Canada.
This work was done while C.-H. Hsu was a Ph.D. student at Simon Fraser University.
Author's address: M. Hefeeda, School of Computer Science, Simon Fraser University, 250-13450 102nd Ave., Surrey BC V3T 0A3, Canada; email: mhefeeda@cs.sfu.ca.

DOI 10.1145/2071396.2071399 http://doi.acm.org/10.1145/2071396.2071399

million subscribers for such services over the coming few years [EU DVB-H 2008]. Since low-quality videos may drive users away from mobile TV services, network operators *must* provide high-quality, commercially-viable services in order to increase their market shares and revenues. Hence, analyzing and fine-tunning the performance of networks providing mobile TV services are important research problems with real-life impacts.

Mobile TV networks are fairly complex systems with many software and hardware components interacting with each other. Thus, the performance of such networks should be considered from many aspects, such as energy consumption of mobile devices, channel switching delay, network capacity in terms of number of TV channels that can be broadcast, perceived visual quality of TV channels, interactivity between users and the network, mobility management, among many others. Despite this complexity, many researchers designing algorithms and protocols for mobile TV networks had to resort to simulation and/or theoretical analysis to assess the performance of their algorithms. Simulation and theoretical analysis, unfortunately, typically abstract away many details that could impact the actual performance in real networks. Therefore, although simulation and theoretical analysis are great tools for preliminary assessment of the potential of new ideas, actual implementation in a running testbed network and conducting reproducible experiments are essential to evaluate the real performance of any new idea. For academic researchers, however, actual implementation is currently not possible. This is because commercial, carrier-grade or pilot, products for mobile TV networks are not only very expensive, but also are sold as "black boxes," that is, their source codes are not available for modifications.

In this article, we present the detailed design of a testbed for one of the most widely deployed standards for mobile TV networks: Digital Video Broadcast–Handheld (DVB-H) [DVB-H Home Page 2009; Faria et al. 2006; Kornfeld and May 2007]. We also conduct empirical evaluation of various performance aspects of mobile TV networks. In particular, our contributions can be summarized as follows.

—We present the design and implementation of a testbed for mobile TV networks. To the best of our knowledge, there exists no other complete open-source testbed for DVB-H networks. The implementation of the testbed is indeed a challenging research problem, as it involves thousands of lines of low-level code and numerous scripts to configure and control several hardware and software components.[1] The proposed testbed includes a base station, data analyzer, and management tools. For the base station, we present a multithreaded software implementation of the whole DVB-H protocol stack on Linux. We use this multithreaded design to utilize the multicore processor architectures that are quite common nowadays, in order to provide the needed capacity to broadcast multiple TV channels. We experimentally show that our base station implementation is efficient and scalable: It can concurrently prepare and broadcast almost 50 TV channels on a *commodity PC* with a dual-core Intel processor, while keeping the CPU utilization below 20%. The 50 TV channels are practically the maximum possible as they saturate the whole available capacity of the wireless spectrum in almost all real deployments. The data analyzer, developed in Java on Linux, captures traffic from the DVB-H network and then analyzes several performance metrics such as the channel switching delay. The management tools easily configure all parameters of the hardware and software components of the base station using a web-based interface. In addition to our open design and source code, we summarize our experiences in building and using the mobile TV testbed, which could be of interest to other researchers in the area.

—We empirically show the importance of minimizing the energy spent in receiving and decoding DVB-H signals, as our measurements on mobile phones indicate that this energy could be as high as 40% of the total energy consumption. Our measurements confirm previous simulation results [Yang et al.

[1]The testbed has been demonstrated at the ACM Multimedia 2008 Conference [Hefeeda et al. 2008].

2004; ETSI 2007a] that demonstrate the importance of the time slicing schemes, in which the base station broadcasts the data of a TV channel in bursts with bit rate higher than the encoding bit rate of the video such that mobile devices can receive a burst of traffic then turn off their DVB-H signal receivers till the next burst in order to conserve energy.

—We analyze the time slicing scheme defined in the DVB-H standard [ETSI 2007a] and its impact on the channel switching delay. Our analysis shows the existence of a tradeoff between the energy saving achieved by the timing slicing scheme and the resulting channel switching delay: Increasing energy saving (by sending data in larger, but less frequent, bursts) results in an increase in the channel switching delay, which might be annoying to users. We then show that a good balance between energy saving and channel switching delay can be achieved using simulcast and scalable video coding, in which a TV channel is encoded in multiple quality layers and the lowest layers are broadcast more frequently in small bursts. Finally, we demonstrate the importance of encoding TV channels carrying diverse types of video content using different bit rates. We also show that choosing the appropriate bit rate not only maximizes the visual quality of the user and bandwidth utilization of the wireless medium, but also reduces the energy consumption of mobile devices.

—We outline how the proposed open-source testbed can be used to enhance and validate several algorithms proposed in the literature to optimize the performance of mobile TV networks from different angles.

The rest of the article is organized as follows. In Section 2, we present an overview of different architectures for providing mobile TV services, and we present some details on the Digital Video Broadcast (DVB) standards family, focusing on the DVB-H standard for which we built the testbed. We present the design of the proposed testbed in Section 3. We present the details of our empirical study in Section 4, and we outline possible other uses of the testbed in Section 5. We conclude the article in the Section 6.

## 2. BACKGROUND

This section first presents an overview of different architectures for delivering mobile TV programs. Then, it provides some details on DVB-H networks, for which we have designed the testbed.

### 2.1 Mobile TV: The Big Picture

Delivering TV programs to mobile devices such as cell phones and PDAs can be done over wireless cellular networks or over dedicated broadcast networks [Furht and Ahson 2008]. Traditional cellular networks use unicast model, which does not efficiently utilize network resources especially in urban areas where there are many users interested in the same content. To cope with this problem, extensions to support multicast and broadcast models have been proposed for cellular networks. For example, the 3G Partnership Project (3GPP) has defined an integrated multicast and broadcast extension, called Multimedia Broadcast Multicast Service (MBMS) [Hartung et al. 2007], for Universal Mobile Telecommunications Systems (UMTS). Video streaming in cellular networks are outside the scope of this article. We focus on dedicated broadcast networks, which have the potential to serve TV content to a large number of subscribers. We note that dedicated broadcast networks usually employ cellular networks to enable user interaction with some TV programs, but not to transmit video.

There are several example systems and standards for dedicated video broadcast networks, including T-DMB (Terrestrial-Digital Multimedia Broadcasting) [Cho et al. 2007], ISDB-T (Integrated Services Digital Broad-casting-Terrestrial) [Takada and Saito 2006], MediaFLO (Forward Link Only technology) [FLO Overview 2009], and DVB-H (Digital Video Broadcast-Handheld) [Faria et al. 2006; Kornfeld and May 2007]. A brief overview of each follows. T-DMB [Cho et al. 2007] is an extension for the DAB

(Digital Audio Broadcast) standard [ETSI 2001] to add video broadcast services to the high-quality audio services offered by DAB. The extension includes both source coding, such as using MPEG-4/AVC encoding, and channel coding, such as employing Reed-Solomon code. The development of T-DMB is supported by the South Korean government, and T-DMB is the first commercial mobile video broadcast service. ISDB-T [Takada and Saito 2006] is a digital video broadcast standard defined in Japan, which is not only for fixed video receivers but also for mobile receivers. ISDB-T divides its band into 13 segments, where 12 of them are used for broadcasting HDTV and one is for broadcasting to mobile devices. MediaFLO [FLO Overview 2009] is a video broadcast system developed by Qualcomm and the FLO forum [FLO Forum Home Page 2008]. MediaFLO is designed from scratch for video broadcast services to mobile devices. The details of the design are not public. In contrast, DVB-H [Faria et al. 2006; Kornfeld and May 2007] is an open international standard [ETSI 2004b].

### 2.2 Digital Video Broadcast (DVB) Standards Family

With participants from over 35 countries, the Digital Video Broadcast (DVB) consortium started designing a series of international standards to support digital television and data broadcast services in 1993. Many of these standards, such as DVB-S for broadcasting over satellite links, DVB-C for broadcasting over cables, DVB-T for broadcasting over airwaves, have been widely deployed. It is reported that more than 170 million DVB receivers have been sold at the time of writing [DVB Home Page 2008]. Several updates of these standards, the second-generations, are being actively developed. In the late 90's, DVB consortium investigated the potential of receiving DVB-T signals using mobile devices. They have concluded that with a spatial diversity antennas for better reception, DVB-T signals can be received by mobile devices [ETSI 2004a].

While DVB-T can support some mobile usages, it is not suitable for streaming multimedia contents to small handheld devices such as cellular phones for three reasons. First, most handheld devices are battery-powered and have very limited power. Unfortunately, power consumption is never a considered factor when designing DVB-T standard as DVB-T receivers are often powered by external sources. Second, handheld devices suffer from serious and varying Doppler shift, fading and interference because their various degree of movements and heterogeneous environments (in-door, out-door, or in-car). In addition, many of these devices concurrently support broadcast and cellular networks, which leads to more intersystem interference. This problem is even more severe considering these handheld devices can only adopt small built-in antennas with stringent power constraints, which rules out most of the advanced antennas such as direction and spatial diversity antennas, and results in poor antenna gain. Thus, radio performance better than what can be achieved by the DVB-T standard is desired for robust broadcasting. Third, as hand-held devices are mobile, a faster, soft, handoff mechanism is required. To cope with these challenges, the DVB consortium developed the Digital Video Broadcast–Handheld (DVB-H) standard [ETSI 2004b], which is an extension of the DVB-T standard but tailored to handheld devices. DVB-H, published in 2004, addresses the three requirements mentioned above. Currently, trial or full-service DVB-H networks have been deployed in many countries [DVB-H Home Page 2009].

The DVB-H standard defines protocols below the network layer and uses IP as the interface with the higher-layer protocols as illustrated in Figure 1. The DVB-H standard uses a physical layer compatible with the DVB-T standard, which employs Orthogonal Frequency Division Multiplexing (OFDM) modulation. To support interactive end-to-end broadcasting systems, the DVB consortium initialized the development of the IP Datacast standard in 2004, which not only specifies higher layer protocols but also enables cooperation with cellular networks such as UMTS. Incorporating access to cellular networks provides bidirectional communications and enables many interactive services such as Electronic Service Guide (ESG). The IP Datacast standard was finalized in 2007 [ETSI 2007b]. More details about

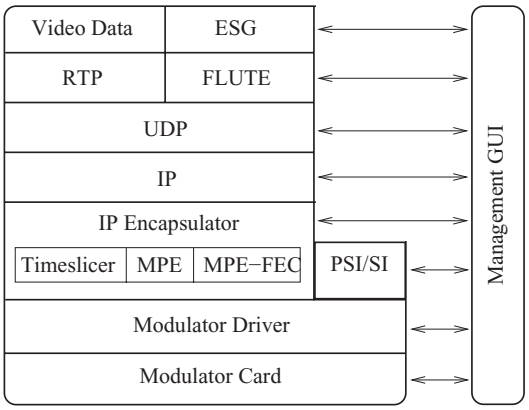


Fig. 1. The protocol stack of mobile TV networks using the DVB-H standard.

the physical layer and IP Datacast standard can be found in Faria et al. [2006] and Kornfeld and May [2007].

DVB-H encapsulates IP packets using Multi-Protocol Encapsulation (MPE) sections to form MPEG-2 transport streams (TS). Thus, data from a specific TV channel form a sequence of MPEs. The MPE encapsulation is done in a module called the *IP encapsulator*. The encapsulated data is then fed to an RF signal modulator to be transmitted over the wireless medium. The IP encapsulator realizes two additional features of DVB-H: time slicing and forward error correction (FEC). In the following, we describe these two features, which provide several performance optimization opportunities.

*Time Slicing.* To save energy of mobile devices, MPEs belonging to a given TV channel are transmitted in *bursts* with a bit rate much higher than the video stream itself. Thus, mobile devices can receive a burst of traffic and then turn off their RF circuits till the next burst. This is known as time slicing. The time period between two adjacent bursts (the off period) is flexible in the sense that the time offset between the start time of the next burst and the current MPE section is sent as part of the MPE header. This enables DVB-H systems to adopt variable burst durations and off durations not only for different video streams but also for the same video stream at different times. We note that the activation of the RF circuits is not instantaneous for two reasons: delay jitter and channel synchronization. Since the next burst start time is broadcast as an offset to the current time, constant transmission delay does not affect its accuracy. However, delay jitter skews the start time of the next burst, and may cause data loss. This jitter is accommodated by adding an extra jitter delay after each off duration, which prevents collisions among bursts. In a typical DVB-H network, the jitter delay is in the order of 10 msec [ETSI 2007a]. In addition, synchronization time refers to the time for RF circuits to search for and lock on the broadcast frequency. The synchronization time is in the range of 50–250 msec [Kornfeld and May 2007; ETSI 2007a]. We collectively refer to the sum of jitter delay and synchronization time as *overhead duration* since no data is transmitted during this period. With time slicing, handheld devices can use their RF circuits to search for signal in adjacent cells between two burst transmissions for seamless, soft, handoffs. Handoffs happen when a handheld device move from a broadcast cell to an adjacent cell. Soft handoff means a device locks up to the signal in the new cell before ignoring the signal in the old cell; hence, there is no interruption during soft handoffs, which leads to good user-experience. Notice that, without time slicing, auxiliary RF circuits are required to support soft handoffs, which incurs significantly higher costs.

Table I. Approximate Costs for Testbed Components

| Vendor | Model | Cost (USD) | Mandatory |
|---|---|---|---|
| DekTec | DTA-110T-SP Modulator | $2,000 | yes |
| Enensys | 1 Watt RF Booster | $2,000 | yes |
| Spectrum | LP49-DTV Antenna | $50 | yes |
| Hauppauge | WinTV-NOVA-T USB Stick | $50 | yes |
| Enensys | DiviCatch RF T/H Tester | $6,000 | no |
| Decontis | dvbSAM Software Analyzer | $5,000 | no |
| Nokia | N96 Handset | Carrier Dependent | no |
| Configuration | | Total Cost (USD) | |
| Complete | | | $15,100 |
| Minimum | | | $4,100 |

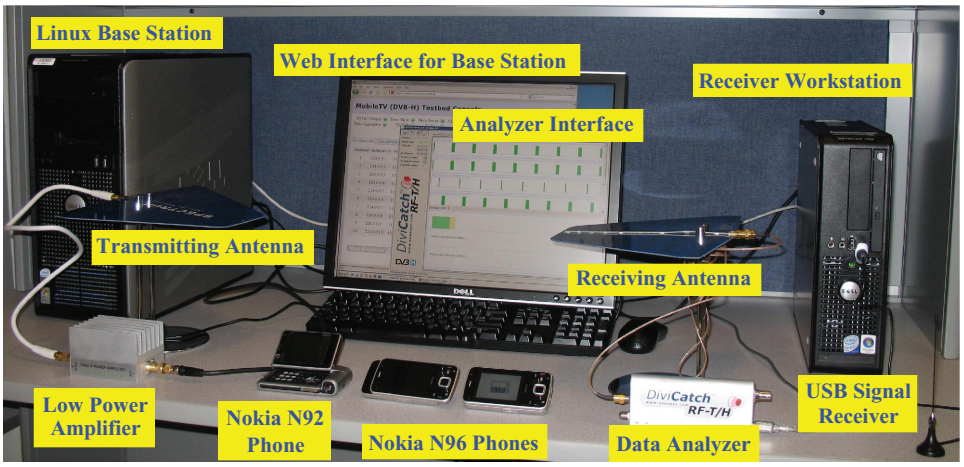


Fig. 2. The hardware setup of the mobile TV testbed. Left: the base station; Right: the receivers/analyzers.

*Forward Error Correction.* The DVB-H standard applies Reed-Solomon (R-S) code and time interleaving in its link layer to protect IP packets in each burst transmission, which largely reduces the minimal carrier-to-noise (C/N) ratio for successful decoding with the same antenna gains. This C/N improvement is reported to be equivalent to the antenna gain given by the spatial diversity [ETSI 2004b], however, no additional space is taken nor more power is consumed by the antenna module. The computation of the R-S parity bytes in DVB-H is detailed in Appendix A.

## 3. DESIGN OF MOBILE TV TESTBED

This section provides the details of the hardware and software components of the proposed mobile TV testbed. Figure 2 shows a picture of the testbed in our laboratory. We divide our description of the testbed into three parts: base station, mobile receivers and data analyzers, and signaling and electronic service guide. Each part is described in one of the following three sections.

Since the cost of building a mobile TV testbed is an important factor for researchers considering to build one, we list the approximate costs for the different parts of the testbed in Table I. The prices in the table are the academic prices obtained in the summer of 2008. We also list the total cost of a complete testbed with all options as well as the cost of the minimum possible configuration of a testbed that is still useful for conducting mobile TV research.

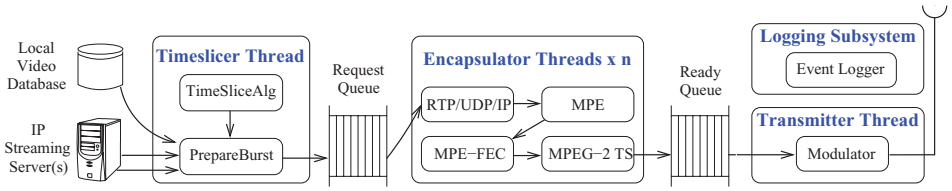


Fig. 3. The proposed design for mobile TV base stations.

We note that our design is modular with well-defined interfaces between the components. Therefore, different hardware/software components can be updated with minimal effects on the others. Thus, we believe our testbed implementation will be useful for future generations of mobile TV systems.

### 3.1 Base Station

We implement the base station in a Linux box in which we install the RF signal modulator DTA-110T-SP available from DekTec [Dektec Modulator 2008]. This modulator implements the physical layer of the protocol stack and transmits DVB-H standard compliant signals via an indoor antenna. We use the low-cost antenna LP49-DTV [Spectrum Indoor Antenna 2008]. The RF output level of the modulator, however, is quite low ($\sim -29$ dBm) and can only reach up to 1-meter broadcast range for receivers with 6dB gain antenna. Mobile devices typically have antenna gains much lower than 6dB. We use the low-power amplifier available from Enensys [Enensys amplifier 2008] to boost the signal to about 0 dBm, which gives us approximately 20-meter range for cellular phones in our lab environment.

A simplified view of our software architecture of the base station is shown in Figure 3. Notice that this software architecture represents only the data plane of the testbed. The control plane is described in Section 3.3. In addition to this software architecture, we have developed a graphical user interface for managing the testbed; a snapshot of this interface is shown in Figure 4. Using this interface, many parameters of the base station can easily be configured. The parameters include: choosing the time slicing algorithm, setting the MPE-FEC frame size, adding and removing TV channels, choosing the physical layer modulation scheme and bitrate, setting the channel PIDs (packet identifiers) and IP multicast addresses, and so on. This interface facilitates conducting various experiments, because it does not require researchers to modify and compile the source code of different components of the system.

*Software Architecture.* We now briefly describe the software architecture in Figure 3. The base station performs many operations in real time. Some of these operations involve blocking I/O tasks (e.g., reading data from a network socket) and expensive processing tasks (e.g., computing checksums and/or FEC codes of bursts). To efficiently support these operations, we employ a multi-threaded design for the base station. As shown in Figure 3, our design has three thread groups: Timeslicer, Encapsulator, and Transmitter. Each of the first and third group has a single thread, while the second group has a pool of $n$ threads, where $n$ is a configuration parameter. As mentioned in Section 1, this design allows us to utilize the multicore processor architectures that are quite common nowadays. In addition, by adjusting thread priorities, the multithreaded design with either single or multicore processors can meet the real-time nature of broadcasting TV channels on a commodity PC or low-end server. For example, we make the priority of the Transmitter thread higher than other threads such that bursts are broadcast on time.

All threads process video data contained in *bursts*. A burst is a data structure that has fields for all needed protocol headers as well as a payload section. Every burst is treated as an independent unit, which allows us to move it from one processing stage to the next without copying data in the user space; only pointers are manipulated and passed between stages. The burst structure has fields for

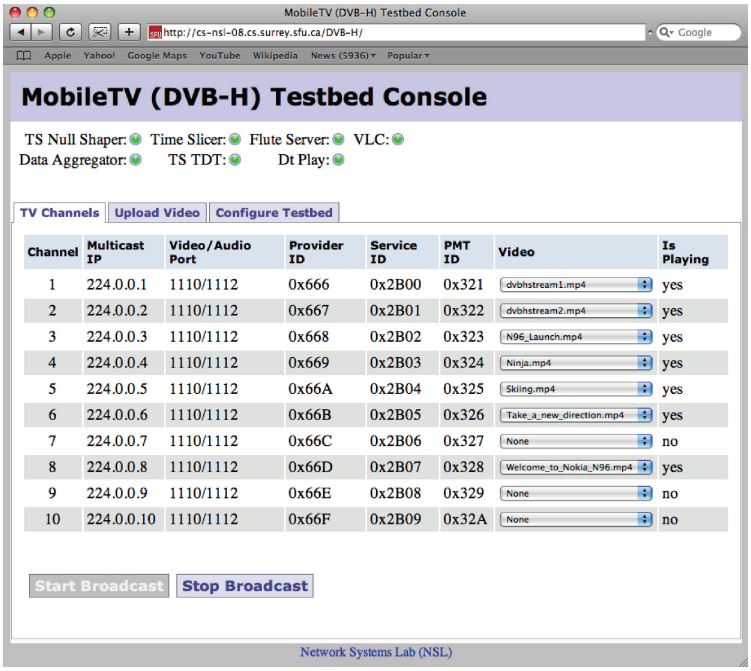


Fig. 4. A snapshot of the Web-based interface for managing the mobile TV testbed.

the RTP, UDP, IP, MPE, MPE-FEC, and MPEG-2 TS protocols in order to accelerate the encapsulation process. In addition, each burst has a timestamp field, which specifies the scheduled broadcast time for that burst. As shown in Figure 3, once a burst finishes processing in a stage, it is put in a priority queue for the next stage. Both queues in our system are sorted based on the timestamp in order to process bursts with closer deadlines earlier.

*Timeslicer Thread.* The Timeslicer thread receives video data from a local video database and/or remote IP video streaming servers. The local database may have pre-encoded TV programs such as TV series, documentary, and movies. The streaming servers can provide live content such as sports events and live talk shows. They also could provide pre-encoded content from online video databases. The Timeslicer thread has two main functions: TimeSliceAlg and PrepareBurst.

The TimeSliceAlg function implements the burst transmission scheduling algorithm, which determines the start time and size of each burst for every TV channel. The scheduling algorithm must ensure smooth playback of the video data by the mobile receivers, that is, the receivers should not have buffer underflow or overflow instances. We designed the TimeSlicer thread to allow different scheduling algorithms to be easily implemented and evaluated. The PrepareBurst function takes the burst schedule produced by the TimeSliceAlg, and divides the data streams of different TV channels into bursts with appropriate sizes and timestamps. The output of the PrepareBurst function is bursts with filled payload sections and timestamps as well as empty header fields (to be completed later by Encapsulator threads).

*Encapsulator Threads.* The Encapsulator threads complete the header fields for the different protocols according to the DVB-H standard, as shown in Figure 3. One of the functions performed by the Encapsulator threads is computing checksums and FEC codes for bursts, which is relatively expensive

given that it needs to be done in real time for potentially many bursts. It is why we create a pool of Encapsulator threads to utilize any idle processing unit in the base station. We also separate the Encapsulation threads from the TimeSlicer thread because the latter could block on reading data from the local data base or from the network socket. Thus, the Encapsulator threads never block unless there is no burst in the Request Queue to be encapsulated. We set the priority of the Encapsulator threads to be lower than the priority of the TimeSlicer thread, such that once the TimeSlicer thread is ready to create bursts it will get a chance to do so.

*Transmitter Thread.* The Transmitter thread is assigned the highest priority among all other threads in our design in order to ensure the on-time delivery of bursts. It reads the bursts from the Ready Queue based on their timestamps and calls the appropriate function in the APIs of the RF signal modulator card. We notice that the APIs employs DMA (direct memory access) to bypass the IP stack and move data from user-space memory to the onboard buffer. It also monitors the actual transmission of bursts and reports any abnormal delays. The Transmitter thread also supports time multiplexing of DVB-H bursts with non DVB-H services, such as PSI/SI (Program Specific Information/Service Information) information. This multiplexing is done as follows. During burst scheduling, the TimeSliceAlg considers the available channel bandwidth as $b - x$ kbps, where $b$ is the total channel bandwidth and $x$ is the amount of bandwidth assigned to non DVB-H services. $b$ is a function of physical- and link-layer settings, while $x$ is a configurable system parameter. Since the TimeSliceAlg schedules bursts for a channel bandwidth lower than the actual one, the resulting schedule consists of reserved time slots for carrying non DVB-H services at bit rate $x$. Therefore, the Transmitter thread can insert packets of non DVB-H services without affecting DVB-H services. Last, we note that DVB-H provides a constant bit rate channel: the base station *must* send traffic packets even when there is no data to transmit. To achieve this, the Transmitter thread sends null packets between any two bursts and for any unused bandwidth reserved for non DVB-H services. We intentionally delay the null packet insertion to the Transmitter in order to reduce the processing overhead of other components such as Encapsulator threads: a statically allocated null packet is repeatedly reused without re-encapsulation.

*Logging Subsystem.* We have implemented an event logging module for analyzing the correctness and performance of the base station. The logging module supports multiple levels of events, which allows researcher to collect logs with the most appropriate amount of details. For example, to validate the implementation, we can enable DEBUG-level logs, which consist of many information including packet dumps. However, once the system is stable, we can switch to STATS-level logging which suppresses all DEBUG info but still saves statistical samples. Finally, for real deployments, the logging module can be disabled to avoid any processing and memory overhead.

### 3.2 Mobile Receivers and Data Analyzers

The goal of the proposed testbed is to analyze various quantitative as well as qualitative performance metrics of mobile TV networks. These metrics include visual quality, energy consumption of mobile devices, channel switching delay, user interactivity with the system, and ease and intuitiveness of the electronic service guide. To assess these performance metrics, we have integrated different types of receivers into the proposed testbed, which are briefly described below.

*Nokia N96 Mobile Phones.* We use Nokia N96 (and its predecessor N92) phones as receivers. These phones are equipped with a DVB-H signal receiver, the receiver-side part of the DVB-H protocol and a video player. These phones are used to verify the correctness and compliance of our base station implementation to the DVB-H and IP Datacast standards. The phones can also be used in user studies to assess the subjective visual quality of the broadcast videos. This can be useful, for example, when

new video encoders are being tested or when searching for the bit rates for different types of video content to maximize the visual quality on actual mobile receivers. Also mobile phones can be used to evaluate the user interface of the mobile services and test new applications such as integrating cell phone and video broadcasting services (e.g., interactive TV shows with user inputs/votes).

At lower layers, the actual energy consumption of receiving DVB-H signals and switching delay among TV channels can be measured on the Nokia phone. The energy consumption can be measured using the Nokia Energy Profiler application, which runs on the mobile phone to monitor energy consumption in real time. The Energy Profiler is quite flexible, and it supports exporting measurement data logs to files for further analysis. The channel switching delay, on the other hand, can be measured either by instrumenting the video player or by implementing a utility as the middle-box between the DVB-H receiver and the video player. The second approach is more feasible because the video player on the Nokia phone is proprietary, which prevents us from augmenting it.

*DVB-H Signal Analyzers.* As the mobile TV application is proprietary, the mobile phones can not be used for evaluating the correctness and performance of different protocols in the testbed implementation. To cope with this, we add DVB-H analyzers to the testbed. We consider two popular analyzers: DiviCatch RF T/H tester, available from Enensys [DiviCatch Analyzer 2008] and dvbSAM, available from Decontis [dvbSAM Analyzer 2008]. The DiviCatch analyzer is attached to a PC via a USB port, and comes with a visualization software that runs on Windows. The software provides detailed real-time information on the RF signals, the MPE frames, the burst schedules, the burst jitters, among others. It also comes with visualization software, which presents crucial information in an intuitive and user-friendly way, and is suitable for field testing, for instance, to locate dead zones in a real deployment. However, based on our experience, we believe that this analyzer is not very useful in validating the testbed implementation. This is because the analyzer, as a real time tester, does not allow byte-level packet analysis. Hence, it can only diagnose serious implementation errors such as early bursts.

Different from DiviCatch, dvbSAM is a software based solution that is compatible with many DVB-T USB receivers in the market. For example, we use a low-cost WinTV-NOVA-T receiver from Hauppauge [WinTV-NOVA-T USB stick 2008]. dvbSAM is also a Windows application. It allows us to dump various information, such as PSI/SI tables and ESG fragments, in hexadecimal format to debug the testbed implementation. Hence, we believe dvbSAM is more useful for researchers to validate the algorithms and protocols implemented in the base station, while DiviCatch is suitable for planning broadcast networks. We note that the costs of these two analyzers are comparable: $5,000–$6,000.

*Our Open-Source Analyzer.* The above commercial analyzers are useful for debugging the implementation of new protocols and algorithms. However, their source codes are not available, and therefore cannot be programmed to measure new performance metrics that may be of interest to researchers. For example, while measuring the channel switching delay is quite important [Hsu and Hefeeda 2009; Rezaei et al. 2008, 2007], none of the commercial analyzers provides a systematic way to collect samples of the channel switching delay. In addition, they are relatively expensive. Hence, an open-source DVB-H analyzer is desirable.

We have developed an open-source analyzer in Java that runs on Linux. The analyzer leverages several utilities developed by the Linux TV project [Linux TV Project 2008]. The Linux TV project, part of the official Linux 2.6 kernel, supports many DVB-T TV receivers, including the low-cost WinTV-NOVA-T receiver [WinTV-NOVA-T USB stick 2008]. Our analyzer supports capturing TS and IP streams as follows. Upon setting up the DVB drivers, the USB receiver shows up as /dev/dvb, which allows us to perform a signal scan using the dvbscan utility. The dvbscan utility returns the modulator parameters, such as frequency, bandwidth, and modulation scheme. With these parameters, the USB receiver can

be tuned to the right network using the tzap utility. Then, we can use the dvbsnoop utility to record and analyze the MPEG-2 TS streams. To extract IP streams from the TS streams, the analyzer uses the dvbnet utility to configure a network interface for each DVB-H channel, and demultiplex TS packets based on the PIDs. The analyzer then uses libpcap library to capture the IP streams.

With the capability of capturing both TS and IP streams, our analyzer can measure system performance from different aspects, such as the energy saving and the channel switching delay. Our analyzer has four parts: tuner, channel monitor, video player, and user interface. The tuner sets up the DVB-T receiver and network interfaces. Each channel monitor is a thread capturing IP packets for a specific DVB-H channel, where each packet is associated with a receiving timestamp. Based on the timestamps, the channel monitor clusters packets into bursts, then measures the performance metrics. At any moment, a channel monitor can relay the captured packets to the video player for play-out. The user interface presents the statistics collected by the channel monitors, and allow users to select a channel to watch. Since the analyzer is open-source, it can be easily extended to evaluate other performance metrics.

### 3.3 Signaling and Electronic Service Guide

This section describes the control plane of the testbed. Unlike the data plane, which handles broadcasting the actual data of TV channels, the control plane manages the auxiliary information transmitted by the base station in order to enable mobile devices to receive and successfully decode different TV channels. The control plane has two parts: PSI/SI (Program Service Information and Service Information) and ESG (Electronic Service Guide). We describe each of them in the following.

*PSI/SI.* The PSI/SI provides link-layer signaling for carrying network configurations. It is organized as tables, where each table is encapsulated in one or more TS packets. The TS packets of the same PSI/SI table share a PID, which allows receivers to reassemble PSI/SI tables. Several tables are defined in the standard, for instance, Network Information Table (NIT) carries the tuning information such as frequency, bandwidth, and modulation scheme. To enable new receivers to find the DVB-H services, every PSI/SI table is retransmitted every few hundred milliseconds (the standard specifies various maximum retransmission period for different tables). As aforementioned, the TS packets of PSI/SI tables are multiplexed with the DVB-H packets by the Transmitter thread. This can be done very efficiently by repeatedly sending the same packets in memory, as most PSI/SI tables are fairly stable over time.

In the testbed, we abstract each PSI/SI table as a class that provides three functions. First, it allows users to modify table content on-line, through a graphic user interface. Second, it can read/write the content from/to a human-readable file, which enables users to change the content offline. Third, and most importantly, it can pack table content into TS packets following the packet format defined in the standard. The packed TS packets are then sent by the Transmitter thread.

*ESG.* The ESG provides application-layer signaling to describe the offered TV services, which allows mobile devices to present information about available programs to users. In addition, ESG also carries session information that enables mobile devices to locate and play the specific IP streams. ESG information is written as XML files, which are then divided into fragments. There are several fragment types defined in the standard, for instance, Content fragments provide textual description of TV programs. To reduce the processing time at receivers, ESG fragments are encapsulated into a few binary container files with indexing fields. These files are sent to mobile devices using the File Delivery over Unidirectional Transport (FLUTE) protocol. To support new receivers, ESG fragments are also periodically retransmitted, but with a longer period (in the order of seconds). The IP streams of the

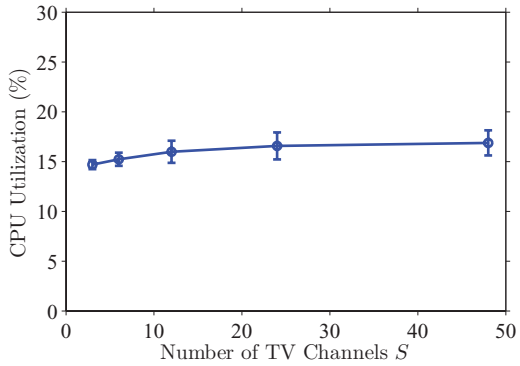


Fig. 5. The CPU utilization of the testbed implementation with different number of concurrent TV channels.

FLUTE protocol are encapsulated as DVB-H channels, which are then time-sliced with other DVB-H TV channels.

We implemented ESG as XML files in the testbed. We developed scripts to systematically create these XML files, and encapsulate them into binary container files. To send these container files, we integrated an open-source FLUTE implementation [MAD-FLUTE Project 2008] into our testbed after several modifications/customizations. The customizations are required in order to comply with the IP Datacast standard [Kornfeld and May 2007], because the FLUTE implementation targets many applications other than DVB-H ESG.

## 4. EMPIRICAL EVALUATION OF MOBILE TV NETWORKS

In this section, we first show that our design of the proposed mobile TV testbed is efficient and scalable. Then, we use the testbed to evaluate various important aspects of mobile TV networks, including time slicing, energy saving, channel switching delay, encoding bit rate of TV channels, and actual power consumption of mobile devices. Due to space limitations, we present the evaluation of channel switching delay in Appendix B, and the impact of encoding bit rate in Appendix C.

### 4.1 Efficiency and Scalability of the Testbed Implementation

The testbed implementation needs to be efficient and scalable, so that it allows researchers to add their algorithms onto it. For an illustration, if the testbed implementation consumes more than 80% CPU power while broadcasting only few TV channels, there is little CPU power left for other algorithms to run, which renders the testbed useless. To quantify the efficiency and scalability of the testbed implementation, we conduct experiments as follows. We broadcast multiple ($S$) TV channels using a commodity PC with an Intel 1.86GHz dual-core processor. The DVB-H modulator card is configured to use an 8MHz radio channel with QPSK (Quadrature Phase-Shift Keying) modulation scheme. The guard interval is 1/8, and the convolutional coding rate is 3/4. According to the DVB-H standard documents, this gives us a wireless medium with 8.29Mbps bandwidth [ETSI 2007a]. All TV channels are encoded at 128kbps. We use the burst scheduling algorithm defined in the DVB-H standard [ETSI 2007a], and we configure it to send bursts without FEC at 5-sec inter-burst intervals. For each value of $S$, we broadcast for three minutes. We use the Linux sar (system activity reporter) utility to collect the CPU activity at 1sec intervals (a small warm-up period is excluded). This utility reports the average idle time of the two CPU cores, which allows us to compute the CPU utilization of the testbed implementation. We vary $S$ from 3 to 48. This covers all practical values of $S$. We report the average CPU utilization with 95% confidence interval in Figure 5. This figure illustrates that the testbed

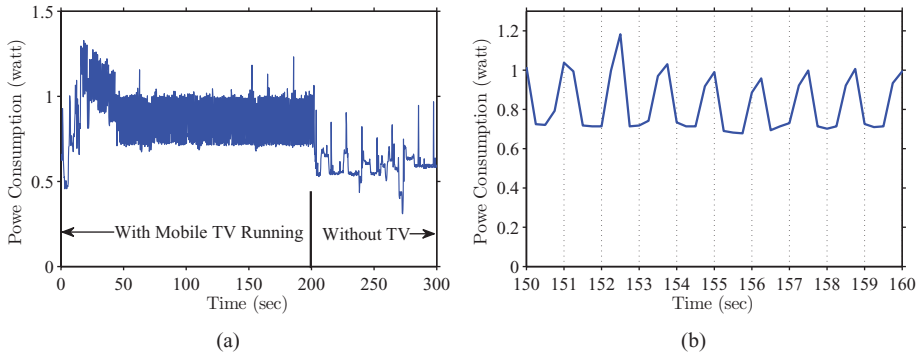


Fig. 6. Power consumption of the DVB-H component measured on Nokia N96 phones. (a) overview and (b) zoom-in to a 10-sec interval.

implementation is efficient: more than 80% of CPU power is still available for researchers to implement and evaluate various algorithms. This figure also shows that the testbed implementation is scalable: adding more TV channels incurs negligible, about 2%, overhead in terms of CPU utilization.

### 4.2 Energy Saving Achieved by Time-Slicing on Mobile Phones

We use Nokia N96 phones to study the amount of energy saving that can be achieved by time-slicing. We use an 8MHz radio channel to broadcast three videos coded at 330kbps. The QPSK modulation scheme is adopted with the convolutional coding rate at 2/3 and guard interval at 1/8. We set the inter-burst interval $\Delta T = 1.25$sec, while bursts are about 0.25sec long each. We use the Nokia phone to watch the TV program, and measure the power consumption using the Nokia Energy Profiler application mentioned. We configure the profiler to take a measurement every 0.25sec, which is the finest granularity it supports. We use the mobile TV application that comes with the Nokia phone to watch one of the TV programs for 200sec. We also measure the energy consumption when the mobile TV application is not running for another 100sec. Figure 6(a) plots the energy consumption during this experiment, which shows that the additional power consumption incurred by watching mobile TV is about 0.3 watt. We also observe many fluctuations of power consumption when the mobile TV application is running. To see more details about the fluctuations, in Figure 6(b), we zoom into a 10sec time interval. We draw two observations from this figure. First, the fluctuations are due to turning on/off the DVB-H radio component: a clear 1.25sec period is observed (8 spikes in 10sec). Second, the power consumption of the radio component is about 0.25 watt. Notice that, 0.25 watt is nontrivial for cellular phones; it is actually more than 40% of the power consumption of the N96 phone when it is not playing TV programs (which is about 0.6 watt as shown in Figure 6(a)). Hence, this experiment confirms the importance of energy conservation mechanisms, such as time slicing, to battery-powered mobile devices.

### 4.3 Impact of Time Slicing on Energy Saving and Channel Switching Delay

We next consider streaming the same video sequence with different DVB-H parameters, and measure various performance metrics. One important parameter in DVB-H networks is the inter-burst interval $\Delta T$, which specifies how frequent bursts are allocated to a specific TV channel. In many deployed networks, $\Delta T$ is manually configured by operators. In this experiment, we encode a video stream using the $\times$264 encoder [$\times$264 Project Page 2008] at bitrate 512kbps. We then transmit this stream over four TV channels with different $\Delta T$, ranging from 90 to 360 msec, for 5 minutes. We instrument the base station software to save detailed logs of all bursts: for each burst we store its start and end

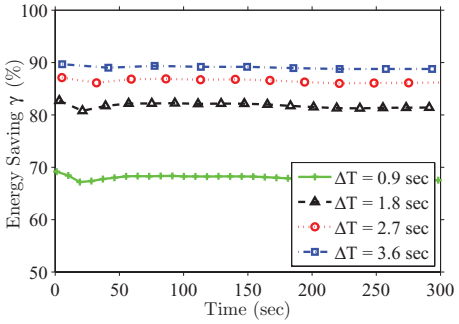


Fig. 7. The energy saving achieved by streaming the same video sequence with different inter-burst intervals.

times and the amount of carried data (including protocol overhead). We then analyze the log files for the performance of the network as well as individual TV channels.

Notice that receivers must turn on their radio components slightly earlier than the burst start time, because it takes time for radio components to lock on to the radio frequency. Therefore, in addition to the time for the actual video data, each burst also incurs a fixed overhead (denoted by $T_o$). To quantify the implication of changing $\Delta T$ on energy consumption, we let $\gamma_s$ be the energy saving achieved by TV channel $s$ because of the time slicing scheme. $\gamma_s$ is defined as the ratio of time the radio component is in off mode to the total time [ETSI 2007a; Yang et al. 2004]. We assume $T_o = 250$msec [Kornfeld and May 2007; ETSI 2007a] and compute the energy saving of each TV channel. The results are plotted in Figure 7. This figure shows that sending the same video with different $\Delta T$ can lead to quite diverse energy saving: up to 22% difference is observed. Therefore, to achieve higher energy saving, longer inter-burst periods ($\Delta T$) are preferred.

Although longer $\Delta T$ values result in higher energy saving, they also result in longer channel switching delay, which is the time a user waits before s/he starts viewing a selected channel when a new channel is requested by that user. With time-slicing, the switching delay increases because a user who switches the channel between two bursts must wait until the next burst for the video data is broadcast. Channel switching delay is annoying because many viewers frequently switch channels before they decide to watch a specific one. To quantify the increase in switching delay due to time slicing,[2] we consider the worst-case scenario where the channel switching happens right after a burst ends, and the receivers have to wait until the next burst starts. We define the time the receivers must wait as the maximum switching delay. We compute the maximum switching delay for individual TV channels in the experiment, and plot the results in Figure 8. This figure illustrates that longer $\Delta T$ increases the switching delay, it could be as high as 3.5 sec. The excessive delay clearly degrades service quality and may drive subscribers away from the service. Therefore, there exists a trade-off between energy saving and switching delay, which needs to be carefully considered while choosing the inter-burst interval $\Delta T$.

We should note that since the video stream is coded at constant bitrate, adopting a longer $\Delta T$ results in longer burst size as well as larger receiver buffer size. Therefore, $\Delta T$ is constrained by both maximal burst size allowed by the DVB-H standard and the receiver buffer size, and can not be arbitrary large. In particular, the standard specifies that each burst consists of exactly one MPE frame, while an MPE frame cannot have more than $1024 * 199 = 195,584$ bytes of payload data [ETSI 2007a]. $\Delta T$ cannot

[2]As mentioned in Section 5, there may be other components contributing to the total switching delay such as the frame refresh delay.

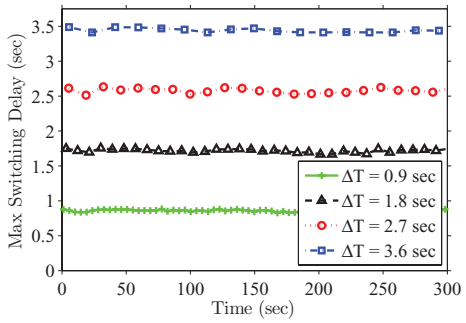


Fig. 8. The maximum switching delay of streaming the same video sequence with different inter-burst intervals.

be too small either. If $\Delta T \leq T_o$, the radio components are never turned off, and the DVB-H service is degraded to continuous receiving with zero energy saving.

## 5. OTHER POTENTIAL APPLICATIONS OF THE TESTBED

We showed in the previous section that our testbed can be used to evaluate various performance metrics in mobile TV networks. In this section, we outline further applications of our testbed as well as demonstrate how other research proposals in the literature could benefit from it.

*Measuring and Controlling Frame Refresh Delay.* Several previous works address the problem of reducing the frame refresh delay which contributes to the total channel switching delay [Vadakital et al. 2007; Rezaei et al. 2007, 2008, 2006a, 2006b]. The frame refresh delay refers to the time period between receiving the first bit of a new video stream and reaching the next random access point, typically an intra-coded frame, of that video. Shorter frame refresh delays lead to shorter total channel switching time, and thus more responsive systems. To reduce frame refresh delays, Vadakital et al. [2007] propose to periodically add redundant intracoded frames into video streams coded by H.264/AVC [Wiegand et al. 2003]. By frequently adding low quality intra-coded redundant frames into a video stream, more random access points are created, which in turn reduces the refresh delay. Alternatively, Rezaei et al. [2007, 2006a] propose transmitting the intra-coded frames in the beginning of the bursts to shorten frame refresh delays. All of these approaches can be validated and evaluated using our testbed, instead of resorting to simulations. For example, the video streams belonging to different TV channels can be processed according to the above proposals before feeding them into the TimeSlicer thread in our design. Then, detailed statistics can be collected by our data analyzers. In addition the actual visual quality of the modified streams can be assessed on the mobile phones.

*Integrating DVB-H, WiFi and Bluetooth Networks.* Wireless LANs (or WiFi spots) have been deployed in numerous locations, including homes, hotels, coffee shops, restaurants, and airports. They provide high bandwidth and cost-effective access to the Internet. In addition, bluetooth networks require no network infrastructure and are supported by many modern devices. Given that many mobile phones (e.g., the Nokia N96) are already equipped with both WiFi and bluetooth network access cards, it might be interesting to integrate the operation of the DVB-H network with wireless LANs and bluetooth network when they are accessible by the mobile phones. For instance, the WiFi network can provide the feedback channel needed in broadcasting interactive TV programs, instead of using more expensive cell phone lines. In addition, wireless LANs and bluetooth networks could provide opportunities for more energy saving for the mobile phones, which is important to extend the viewing time on them. For example, Zhang et al. [2007] consider mobile TV receivers with an auxiliary short range wireless interface and construct a cooperative network among several receivers over this short range wireless network.

Mobile TV receivers share received IP packets over this short range network, so that each mobile TV receiver only receives a small fraction of IP packets directly from the DVB-H network. This allows these receivers to reduce the frequency of turning on their RF circuits. Assuming sending/receiving IP packets through the short range network is more energy efficient than receiving DVB-H sections, this cooperative strategy can save energy consumption. The actual energy saving from the proposal in Zhang et al. [2007], as well as the protocols needed for enabling cooperation among phones, can be evaluated using our testbed. Adding a wireless access point and bluetooth interface to the testbed is straightforward and inexpensive. Also, the N96 phones can be configured to access a wireless LAN or a bluetooth network with only a few keystrokes.

*Joint Design of Streaming Server and Base Station.* Mobile TV broadcast networks have different characteristics than other general Internet streaming systems, including energy consideration of mobile receivers, limited bandwidth of the wireless medium, and limited resolution and screen size of mobile receivers. Thus, customizing the multimedia content explicitly for mobile TV networks can enhance the user experience as well as the utilization of the network resources. For example, Rezaei et al. [2008] propose a rate control algorithm for broadcasting multiple video channels using DVB-H systems. The objective of this rate control algorithm is to determine the best quantization parameter (QP) for individual TV channels for equalizing the video quality across all TV channels. The algorithm consists of two components: a rate controller based on fuzzy logic and a heuristic quality controller that determines QP values based on historical rate-distortion (R-D) curves. The QP values are then sent back to the video encoder for rate adaptation. Our proposed testbed can be extended to support and fine-tune the algorithm in Rezaei et al. [2008] and similar algorithms.

*Cross-Layer Optimization.* The DVB-H system has several parameters in different layers. The interaction of these parameters should be analyzed and studied for optimal performance of the overall system. The DVB-H system parameters can be classified into three sets: physical layer, time slicing module, and MPE-FEC module. The interaction among parameters in different sets is outlined in Kornfeld [2007]. Kornfeld [2007] also studies the performance optimization of the radio signals in DVB-H networks using *simulations*. The results indicate that extending the time interleaving depth in MPE-FEC to at least 100 ms results in good radio link performance, while increasing interleaving depth beyond 300 ms yields no further improvements. Meanwhile, extending time interleaving depth also increases burst durations and imposes negative impact on power consumption. These simulation results would become more convincing if they are obtained from actual implementation, which our testbed can provide. In addition, our testbed can be augmented by implementing various wireless channel models to emulate different environments in which the mobile phones will be used. For example, channel models for interference and multipath fading for receivers in urban, rural, and highway settings [Rappaport 1996, Chapter 3 and 4], can be implemented in the testbed, in the data analyzers (receivers side) to be more specific. The receivers will modify the characteristic of the received radio signals based on these models, before passing the data to upper layers for decoding.

## 6. CONCLUSIONS AND FUTURE WORK

We have presented the design and implementation of a complete, end-to-end, testbed for evaluating the *real* performance of mobile TV networks from various angles. The proposed testbed is totally open source and available to the research community. Thus, we believe it will stimulate and enable more research in improving the performance of mobile TV networks, which are expected to be pervasive in the near future. The testbed is developed for networks employing the popular Digital Video Broadcast–Handheld (DVB-H) standard. The design of the proposed testbed is modular with well-defined interfaces between the hardware and software components. This enables updating different

hardware/software components with minimal impacts on the others. Therefore, the testbed is useful for current and future generations of mobile TV systems. In addition, because of its modularity, the testbed can easily be extended to support other standards in the larger family of Digital Video Broadcast (DVB), such as DVB-T (Terrestrial), DVB-C (Cable), and DVB-S (Satellite). This extension, part of our future work, would enable studying the benefits/costs of integrating different standards in a comprehensive framework for providing digital video content to anybody, anywhere, and at anytime.

The testbed integrates the standard-compliant software components that we developed with off-the-shelf, low-cost, hardware components. Our software components include: (i) complete, multithreaded, implementation of the broadcasting base station on a commodity PC running Linux, (ii) data analyzer for collecting and analyzing several performance metrics, and (iii) web-based management tools to remotely and easily configure all parameters of the base station. We experimentally demonstrated the scalability and efficiency of our testbed design, as it can broadcast as many TV channels as the current wireless medium spectrum can support. We also summarized the experience gained while building this testbed, which could be interest to other researchers. We believe that the current mobile TV networks and standards still have numerous opportunities for performance optimization and enhancements to the offered services. Many of the deployed networks currently offer few channels and in early stages. Thus as these networks see wide adoption, users will demand better video quality and more interactivity with the network, which will motivate the network operators to seek every possible opportunity to maximize the use of their resources and the allocated (costly) wireless spectrum. Our proposed testbed could aid in finding these opportunities for performance optimization, and therefore it is a timely and useful contribution to the research community.

In addition, we conducted an empirical study to evaluate important performance metrics in mobile TV networks. These performance metrics include: (i) energy consumption of mobile receivers, and (ii) channel switching delay. Our study validated some claims made based on simulations in the literature, and discovered critical performance trade-offs. For example, we experimentally showed that the energy consumed to receive and decode mobile TV signals can reach up to 40% of the total energy consumption of mobile phones, and that time slicing schemes are effective in reducing this energy consumption. We also discovered and quantified the existence of a tradeoff between the energy saving achieved by the timing slicing scheme and the resulting channel switching delay, and we showed that employing scalable video coding and simulcast could achieve a target channel switching delay without sacrificing the energy saving for mobile devices. Finally, we showed that encoding TV channels at bit rates commensurate with the visual complexity of the video content improves the bandwidth utilization of the wireless medium, reduces the energy consumption of mobile devices, and enhances the viewing experience for the users.

## APPENDIXES

### A. COMPUTATION OF FEC BYTES IN DVB-H

DVB-H uses an MPE-FEC frame to compute the R-S parity bytes, and each MPE-FEC frame has 255 columns and at most 1024 rows, where each element is a byte. Therefore, the maximum size of MPE-FEC frame is about 255KB. As illustrated in Figure 9, each MPE-FEC frame is divided into an application data table (ADT) with 191 columns of IP packets and an R-S data table (RSDT) with 64 columns of parity bytes. The R-S bytes are computed in three steps. First, the IP packets are filled into the ADT table column-wise as shown in Figure 9(a). Next, the parity bytes are calculated row-wise from the IP packets in the same MPE-FEC frame as illustrated in Figure 9(b). Once the R-S bytes are ready, IP packets in the same MPE-FEC frame are sequentially transmitted column-wise as MPE sections, which are followed by the R-S bytes encapsulated column-wise in multiprotocol encapsulation sections.

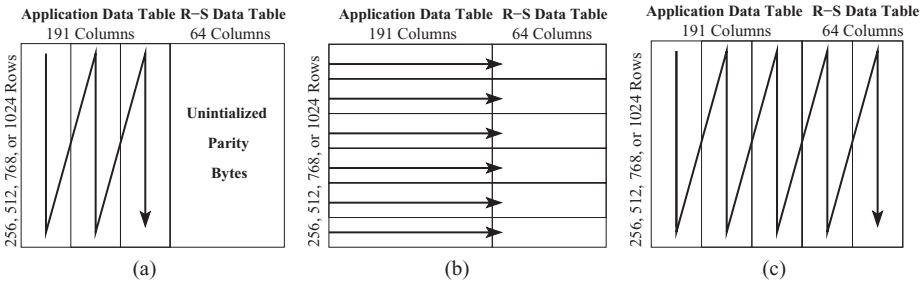


Fig. 9. R-S parity bytes are computed and sent in three steps: (a) IP packets are filled into ADT, (b) R-S bytes are computed based on ADT, and (c) ADT and RSDT are sequentially transmitted.

This is shown in Figure 9(c). At the receiver side, CRC-32 section trailers and R-S parity bytes are used for error detection and correction from transmission errors. Notice that, accessing MPE-FEC frames is column-wise while calculating parity-bit is row-wise, which leads to time interleaving that also helps to combat fading and interference. In common cases, default R-S coding ratio supports recovery from as high as 25% packet loss ratio [Faria et al. 2006]. However, DVB-H supports adaptive R-S coding ratio: padding all zero packets in ADT leads to stronger R-S coding, while puncturing some parity bytes results in weaker R-S coding. When MPE-FEC is jointly used with time slicing, each transmission burst is correspondent to a MPE-FEC frame. As a consequence, the burst size can not go beyond 255KB, which simplifies the receiver hardware design as a fixed memory can be reserved for decoding.

## B. CONTROLLING CHANNEL SWITCHING DELAY

In Section 4.3, we showed that $\Delta T$ can be used to control the trade-off between energy saving and channel switching delay. For example, to reduce the channel switching delay, we can reduce the $\Delta T$ value. This, however, degrades the energy saving of mobile devices. For example, as shown in Figures 7 and 8, if we need to achieve a delay guarantee of 1sec, the mobile device can only achieve less than 70% of energy saving compared to the maximum achievable energy saving of 90%. Hence, using $\Delta T$ to control channel switching delay is not ideal, and a better solution is needed.

We studied the problem of controlling channel switching delay and proposed a time slicing scheme to achieve both low switching delay and high energy saving [Hsu and Hefeeda 2009]. The new time slicing scheme, called SIMU, simulcasts every TV program over two TV channels. For every TV program, one TV channel is optimized for energy saving (referred to as the primary channel), and the other TV channel is optimized for low switching delay (referred to as the bootstrap channel). A mobile device that just switches to a new TV program tunes to the bootstrap TV channel first, which enables the device to start playing the video very quickly. The device tunes to the primary TV channel upon a burst of the primary TV channel is available, which enables the device to save more energy. Moreover, we proposed to broadcast reduced quality videos over the bootstrap channels to save network bandwidth. This can be achieved by encoding a video into multiple layers and using the lowest layer(s) for the reduced-quality versions. Full quality videos are not required for bootstrap channels because mobile devices only receive bootstrap channels for a transient time period, and it takes human eyes some time to detect the visual details in a new scene/channel. In fact, a recent work [Buchinger et al. 2009] shows that users prefer a shorter reduced-quality startup phase than a longer one, which indicates that reduced-quality video streams are not annoying to users. Using the idea of simulcasting, we designed time slicing schemes to broadcast TV programs that provide channel switching delay guarantees for a target delay bound.

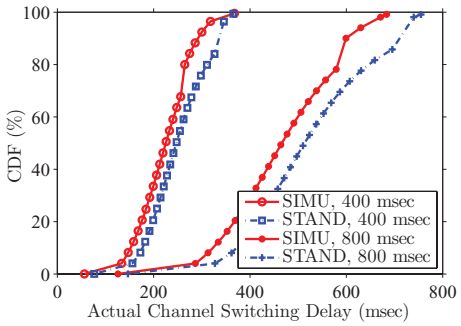


Fig. 10. The actual switching delay for two different delay bounds achieved by two scheduling algorithms.

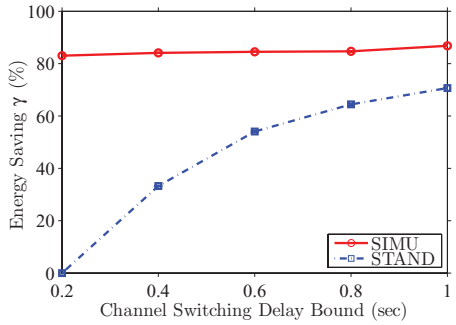


Fig. 11. The energy saving achieved by two scheduling algorithms under different delay bounds.

We implement SIMU in the testbed and compare it against the time slicing scheme defined in the DVB-H standard, which we refer to as STAND. We encode video sequences into two versions: a full quality version at 512kbps and a reduced quality version at 128kbps. We then broadcast these videos over multiple TV channels using both SIMU and STAND algorithms. For a target delay bound, we broadcast these TV channels for five minutes. To collect enough samples for statistical meaningful results, we save detailed log files while broadcasting, which consists of the start and end times of each burst and its size. We also develop a utility to emulate the behavior of a large number of users. We present the result of one million users in this article. For every user, we model his/her channel switching behavior using Bernoulli trials, in which we toss a biased coin every second and issue a channel switching command if the trial is success. The probability of success of the trials is configured in a way that users have on average $w$ sec of watch time for each channel. The new selected channel is randomly chosen from all broadcast channels other than the currently watched one. We run this emulator against each of the log files, and compute the channel switching delay and energy saving using the start and end times recorded in the log files.

We first fix the average watch time $w = 100$ sec, and vary the target delay bound from 0.2 to 1sec. We measure the energy saving and channel switching delay for each instance of channel switching event. We plot the CDF of the actual channel switching delay in Figure 10. For clarity, only sample results are shown in the figure. This figure confirms that both SIMU and STAND provide maximum delay guarantee for a given delay bound. We then present the energy saving achieved by SIMU and STAND in Figure 11. This figure shows that controlling $\Delta T$ in STAND results in poor energy saving when the delay bound is small. In fact, when the delay bound approaches the overhead time period $T_o$,

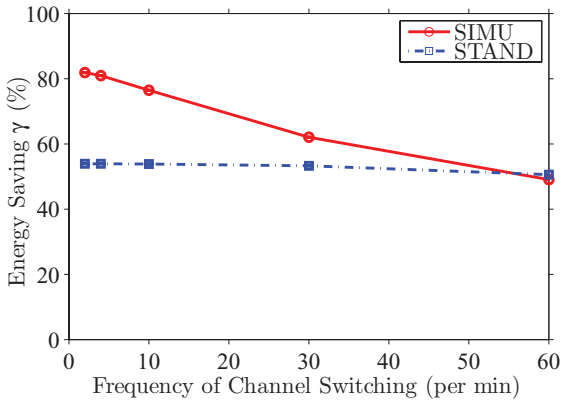


Fig. 12. The energy saving achieved by two scheduling algorithms under different channel switching frequencies.

the energy saving diminishes as the mobile devices cannot turn off their radio components. In contrast, small delay bound does not impose negative impacts on the energy saving of SIMU: the average energy saving is always higher than 80%. The energy saving gap between SIMU and STAND is significant: between 10% to 80% is observed in this experiment. Hence, this experiment shows that SIMU is more energy efficient than STAND in controlling the channel switching delay. We should mention that the SIMU algorithm imposes a slight increase in the usage of the wireless medium bandwidth, because it transmits lower quality versions of the video in addition to the full quality ones.

Next, we consider more aggressive situations, in which users flip through TV channels at higher frequencies than once every 100sec. We vary the watch time $w$ from 1 to 60sec, which covers a wide range of user behavior: from switching the TV channel once every minute to every second. We fix the channel switching delay bound at 0.6sec. We run the experiment with SIMU and STAND algorithms, and plot the energy saving in Figure 12. This figure shows that the SIMU algorithm achieves at least 75% energy saving if the user switches the TV channels less than 10 times per minute, which is at least 20% higher than the STAND algorithm. This figure also illustrates that this difference diminishes if the user switches the TV channels for more than 30 times per minute. This is because, in this case, receivers never stay in a TV channel long enough to receive the primary channel of the SIMU algorithm for higher energy conservation. We, however, believe this setup is extreme and rare, because users who want to *watch* any TV programs can not *constantly* flip through the TV channels. Therefore, normal users will still benefit from the SIMU algorithm for longer watch time, while the "flippers" will turn off the mobile TV receivers soon anyway.

## C. IMPACT OF ENCODING TV CHANNELS AT DIFFERENT BIT RATES

In addition to the aforementioned metrics such as energy saving and switching delay, perceived video quality is also an important performance metric that has a direct impact on user satisfaction. In general, higher encoding rates lead to better video qualities, but also consume more network bandwidth and reduce the number of TV channels that can be concurrently broadcast. Therefore, network operators are interested in broadcasting TV programs that meet a given minimal video quality to retain their subscribers with minimal costs. Perceived video quality, however, is a nonlinear function of the encoding rate and the video characteristics. In other words, to achieve a given minimum quality, different types of video require various bitrates. As an illustrative example, we encode four 5-minute TV programs all at 256kbps using the $\times$264 encoder [$\times$264 Project Page 2008]. We then measure the reconstructed video quality in PSNR (Peak Signal-to-Noise Ratio) of each TV program. Table II

Table II. Perceived Quality of Different Videos Coded at the Same Bitrate

| Name | Resolution | PSNR (dB) |
|---|---|---|
| Game Show | QCIF | 45.24 |
| Documentary | QCIF | 40.58 |
| Talk Show | CIF | 39.85 |
| Sports | CIF | 33.31 |

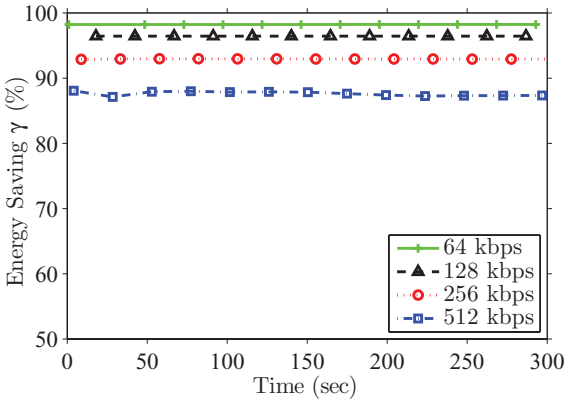


Fig. 13. The energy saving achieved when TV channels are encoded at different bitrates.

summarizes the results. This table indicates that encoding all types of videos at the same bitrate results in quite diverse perceived quality. Assume the target perceived quality is chosen as 35 in PSNR. Clearly the Sports program must be encoded at bitrates higher than 256kbps, while all other programs can be encoded at bitrates lower than 256kbps. Hence, encoding different types of TV programs at different bitrates is necessary for network operators for better service quality.

Broadcasting TV channels at different bitrates, however, is not trivial for network operators who need to determine $\Delta T$ for each TV channel. A simple way of choosing $\Delta T$ is to assign *uniform* $\Delta T$ to all TV channels. Since receivers have limited buffer size, the chosen $\Delta T$ must be small enough to accommodate the TV channel with the highest bitrate. This approach, however, leads to lower energy saving because the receiver buffers are not fully-utilized for the TV channels with smaller bitrates. Therefore, a better way of choosing $\Delta T$ is to assign *different* $\Delta T$ to different TV channels. However, assigning arbitrary $\Delta T$ can easily lead to conflicting bursts and thus infeasible transmission schedules, which may force the network operators to resort to trial-and-error approach that is error-prone.

To design a systematic way to select the $\Delta T$, we studied the burst scheduling problem, which constructs the burst schedule for all TV channels with different bitrates to maximize the energy saving for all receivers. Unfortunately, we proved that the burst scheduling problem for arbitrary bitrates is NP-complete [Hefeeda and Hsu 2009]. Then, we solved a practical simplification of the scheduling problem where TV channels are classified into several classes, and the bitrates of the classes have power of two increments, for instance, 200, 400, and 800. In this case, the proposed solution (called P2OPT) is efficient and produces optimal burst schedule in terms of energy saving. P2OPT achieves the optimal energy saving by allocating bursts to utilize the receiver buffer, that is, assigning the largest possible $\Delta T$ to each TV channel. Meanwhile, P2OPT systematically assigns offsets to individual TV channels to guarantee that the burst schedule has no conflict and is feasible.

We implement P2OPT scheduling algorithm in the testbed. We use the scheduling algorithm to broadcast four video sequences coded at bitrates 64, 128, 256, and 512 kbps for 5 minutes. We assume

the receiver buffer size is the same as the maximal burst size, which is 195,584 bytes as defined in the DVB-H standard [ETSI 2007a]. We collect the detailed log files to compute the energy saving, and we plot the results for each TV channel in Figure 13. This figure shows that broadcasting TV channels with different bitrates indeed leads to different energy saving: about 10% difference is observed. These results also indicate that sending a video with too high quality not only wastes network bandwidth but also degrades the energy saving of mobile devices, and thus reduces the battery life and watch time. Therefore, sending TV channels at the minimal bitrates that can lead to acceptable perceived quality is beneficial to both network operators and subscribers.

## ACKNOWLEDGMENTS

Yi Liu and Cong Ly helped in setting up the testbed and we thank them.

## REFERENCES

BUCHINGER, S., KRIGLSTEIN, S., AND HLAVACS, H. 2009. A comprehensive view on user studies: Survey and open issues for mobile TV. In *Proceedings of the European Interactive Television Conference (EuroITV'09)*. 179–188.

CHO, S., LEE, G., BAE, B., YANG, K., AHN, C., LEE, S., AND AHN, C. 2007. System and services of Terrestrial Digital Multimedia Broadcasting (T-DMB). *IEEE Trans. Broadcas. 53,* 1, 171–178.

DEKTEC MODULATOR 2008. Dektec DTA-110T PCI modulator. http://www.dektec.com/Products/DTA-110T/.

DIVICATCH ANALYZER 2008. Divi Catch RF-T/H transport stream analyzer. http://www.enensys.com/.

DVB-H HOME PAGE 2009. Digital video broadcasting - handheld (DVB-H) homepage. http://www.dvb-h.org/.

DVB HOME PAGE 2008. Digital video broadcasting (DVB) homepage. http://www.dvb.org/.

DVBSAM ANALYZER 2008. dvbSAM DVB-H solution for analysis, monitoring, and measurement. http://www.decontis.com/.

ENENSYS AMPLIFIER 2008. Enensys 1-watt RF power amplifier. http://www.enensys.com/.

ETSI 2001. Radio broadcasting systems: Digital audio broadcasting (DAB) to mobile, portable and fixed receivers. European Telecommunications Standards Institute (ETSI) Standard EN 300 401 Ver. 1.3.3.

ETSI 2004a. Digital video broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television. European Telecommunications Standards Institute (ETSI) Standard EN 300 744 Ver. 1.5.1.

ETSI 2004b. Digital video broadcasting (DVB); Transmission system for handheld terminals (DVB-H). European Telecommunications Standards Institute (ETSI) Standard EN 302 304 Ver. 1.1.1.

ETSI 2007a. Digital video broadcasting (DVB); DVB-H implementation guidelines. European Telecommunications Standards Institute (ETSI) Standard EN 102 377 Ver. 1.3.1.

ETSI 2007b. Digital video broadcasting (DVB); IP datacast over DVB-H: Set of specifications for phase 1. European Telecommunications Standards Institute (ETSI) Standard EN 102 468 Ver. 1.1.1.

EU DVB-H. 2008. Mobile TV across Europe: Commission endorses addition of DVB-H to EU list of official standards. http://europa.eu/rapid/pressReleasesAction.do?reference=IP/08/451&format=PDF.

FARIA, G., HENRIKSSON, J., STARE, E., AND TALMOLA, P. 2006. DVB-H: Digital broadcast services to handheld devices. *Proc. IEEE 94,* 1, 194–209.

FLO FORUM HOMEPAGE 2008. FLO forum homepage. http://www.floforum.org/.

FLO OVERVIEW 2009. FLO technology overview. http://www.mediaflo.com/news/pdf/tech_overview.pdf.

FURHT, B. AND AHSON, S., Eds. 2008. *Handbook of Mobile Broadcasting: DVB-H, DMB, ISDB-T, and MediaFLO* 1st Ed. Auerbach Publications, Boca Raton, FL.

HARTUNG, F., HORN, U., HUSCHKE, J., KAMPMANN, M., LOHMAR, T., AND LUNDEVALL, M. 2007. Delivery of broadcast services in 3G networks. *IEEE Trans. Broadcast. 53,* 1, 188–199.

HEFEEDA, M. AND HSU, C. 2010. On burst transmission scheduling in mobile TV broadcast networks. *IEEE/ACM Trans. Netw. 18*, 2, 610–623.

HEFEEDA, M., HSU, C., AND LIU, Y. 2008. Testbed and experiments for mobile TV (DVB-H) networks. In *Proceedings of the ACM Multimedia'08 Demo Session*.

HSU, C. AND HEFEEDA, M. 2011. Using simulcast and scalable video coding to efficiently control channel switching delay in mobile TV broadcast networks. *ACM Trans. Multimedia Comput. Comm. Applic. 7*, 2.

KORNFELD, M. 2007. Optimizing the DVB-H time interleaving scheme on the link layer for high quality mobile broadcasting reception. In *Proceedings of the IEEE International Symposium on Consumer Electronics (ISCE'07)*. 1–6.

KORNFELD, M. AND MAY, G. 2007. DVB-H and IP datacast—broadcast to handheld devices. *IEEE Trans. Broadcas. 53,* 1, 161–170.

LINUX TV PROJECT 2008. Linux TV project, television with Linux. http://www.linuxtv.org/.

MAD-FLUTE PROJECT 2008. MAD-FLUTE project. http://mad.cs.tut.fi/.

RAPPAPORT, T. 1996. *Wireless Communications: Principles & Practice* 1st Ed. Prentice Hall.

REZAEI, M., BOUAZIZI, I., AND GABBOUJ, M. 2008. Joint video coding and statistical multiplexing for broadcasting over DVB-H channels. *IEEE Trans. Multimedia 10,* 7, 1455–1464.

REZAEI, M., HANNUKSELA, M., AND GABBOUJ, M. 2006a. Video encoding and splicing for tune-in time reduction in IP datacasting (IPDC) over DVB-H. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'06)*. 601–604.

REZAEI, M., HANNUKSELA, M., AND GABBOUJ, M. 2006b. Video splicing and fuzzy rate control in IP multi-protocol encapsulator for tune-in time reduction in IP datacasting (IPDC) over DVB-H. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'06)*. 3041–3044.

REZAEI, M., HANNUKSELA, M., AND GABBOUJ, M. 2007. Tune-in time reduction in video streaming over DVB-H. *IEEE Trans. Broadcast. 53,* 1, 320–328.

SPECTRUM INDOOR ANTENNA 2008. Spectrum LP49-DTV indoor antenna. http://spectrum.co.kr/.

TAKADA, M. AND SAITO, M. 2006. Transmission system for ISDB-T. *Proc. IEEE 94,* 1, 251–256.

VADAKITAL, V., HANNUKSELA, M., AND GABBOUJ, M. 2007. Time-interleaved simulcast and redundant intra picture insertion for reducing tune-in delay in DVB-H. In *Proceedings of the IEEE International Packet Video Workshop (PV'07)*. 123–132.

WIEGAND, T., SULLIVAN, G., BJNTEGAARD, G., AND LUTHRA, A. 2003. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circ. Syst. Video Techn. 13,* 7, 560–576.

WINTV-NOVA-T USB STICK 2008. WinTV-NOVA-T USB dvb-t receiver. http://www.hauppauge.co.uk/.

X264 PROJECT PAGE 2008. x264 - a free H264/AVC encoder. http://www.videolan.org/developers/x264.html.

YANG, X., SONG, Y., OWENS, T., COSMAS, J., AND ITAGAKI, T. 2004. Performance analysis of time slicing in DVB-H. In *Proceedings of the Joint IST Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC'04)*. 183–186.

ZHANG, Q., FITZEK, F., AND KATZ, M. 2007. Cooperative power saving strategies for IP-services supported over DVB-H networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'07)*. 4107–4111.

Received January 2009; revised September 2009; accepted January 2010