MobiUS: Enable Together-Viewing Video Experience across Two Mobile Devices

Guobin Shen
Microsoft Research Asia
Beijing, 100080, China
jackysh@microsoft.com

Yanlin Li
Tianjin University
Tianjin, 300072, China
v-yanlli@microsoft.com

Yongguang Zhang
Microsoft Research Asia
Beijing, 100080, China
ygz@microsoft.com

ABSTRACT

We envision a new *better-together* mobile application paradigm where multiple mobile devices are placed in a close proximity and study a specific *together-viewing* video application in which a higher resolution video is played back across screens of two mobile devices placed side by side. This new scenario imposes real-time, synchronous decoding and rendering requirements which are difficult to achieve because of the intrinsic complexity of video and the resource constraints such as processing power and battery life of mobile devices. We develop a novel efficient collaborative half-frame decoding scheme and design a tightly coupled collaborative system architecture that aggregates resources of both devices to achieve the task. We have implemented the system and conducted experimental evaluation. Results confirm that our proposed collaborative and resource aggregation techniques can achieve our vision of better-together mobile experiences.

Categories and Subject Descriptors

C.3.3 [**Special-Purpose and Application-based Systems**]: Real-time and embedded systems; H.4.3 [**Information Systems Applications**]: Communications Applications; C.5.3 [**Microcomputers**]: Portable devices

General Terms

Algorithms, Design, Performance

Keywords

Better-together, Together-viewing, mobile computing, collaborative decoding, proximity detection, energy efficiency, resource aggregation, screen aggregation.

1. INTRODUCTION

Mobile devices have seen a high market penetration in the past few years and have become increasingly indispensable in our daily lives. More and more of them are equipped with both audio/video and wireless networking capabilities, such as high-end mobile phones and personal media devices like Microsoft Zune. This enables more and richer mobile multimedia related user experiences. For instance, we have seen many new emerging technologies to allow efficient information sharing – sharing of files such as media like audio/video, Flash, ring-tone etc., and documents like MS Word, Powerpoint and PDF files etc.

Figure 1: Example scenarios of screen aggregation.

In this paper, we present MobiUS, a research project aiming to create new *better-together* experiences when multiple mobile devices are placed together. This represents a new application paradigm based on the collaboration and resource sharing and aggregation over mobile devices in close proximity. As a first exploration, we develop software mechanisms and system components to enable *together-viewing video* experience – a real-time, synchronized playback of a higher resolution video across screens when two mobile devices are put side-by-side (Figure 1). We choose this scenario because it is challenging and representative, with the expectation that our results and experiences can apply to other better-together applications like mobile gaming and collaborative mobile authoring. We start from two mobile devices because it is probably the most basic and common case.

In this together-viewing scenario, we assume that one device has downloaded from the Internet or otherwise obtained a higher resolution video whose size is about twice of its screen size. Given that the screen of today's mobile device are relatively small, this is a reasonable assumption. We also assume that the two devices can communicate effectively and directly via high-speed local wireless networks such as WiFi and Bluetooth, which are equipped in most of today's cell phones or PDAs. We further assume that

MobiSys'07, June 11-14, 2007, San Juan, Puerto Rico, USA.

the two devices are homogeneous, i.e., with same or similar software and hardware capabilities, though we explore how to relax this assumption in Section 6.4.

To be specific, this scenario places the following requirements on MobiUS. First, video decoding and playback must be in real-time[1] and must be in sync between the two devices. An effective synchronization mechanism must be in place to ensure the same video frame is rendered at two devices simultaneously, even if the their clocks may be out of sync. Second, MobiUS must work in a resource-constrained environment in which processing power, memory, and battery life may be barely enough for each device to just decode a video of its own screen size. MobiUS should also minimize energy consumptions in processing and communication so that the battery can last as long as possible. Further, it is desirable if the system is also adaptive. For example, it should expand the video on to two devices or shrink it on to one screen alone as the other device comes and goes.

Unlike previous screen aggregation work where screens from multiple PCs are put together to form a bigger virtual screen [1], MobiUS is more challenging because previous techniques like a *remote framebuffer* protocol would consume too much processing power or communication bandwidth. Naive approaches such as having one device do full decoding and send half frames to the peer, or having both devices do full decoding and each display only half will quickly saturate the limited resources that the mobile devices have.

The solution to this and other similar problems in MobiUS applications lays in a tightly coupled collaborative and aggregated computing model for resource-constrained mobile devices. To support the together-viewing video application, we develop a collaborative half-frame video decoding scheme that intelligently divides the decoding task between the two devices and achieves the real-time playback requirement within given constraints. We further optimize this scheme to improve energy efficiency. We also develop a system architecture and necessary components for MobiUS applications, and implement the together-viewing application on two mobile phones. The working system and subsequent experimental evaluation demonstrate that our proposed collaborative and resource aggregation techniques can achieve our vision of better-together mobile experience.

Ultimately, our goal for this better-together mobile paradigm is to explore the social impact of mobile phones as always-carried devices. Our objective is not only to let users enjoy higher quality video, but also make it natural for them enjoy it together, which will facilitate discussions and consolidate the friendship. In fact, the work presented in this paper has motivated a new Microsoft product plan called "Lover's Phone" for young couples. Microsoft has conducted a preliminary market survey (via a third-party independent vendor) on users' opinions on such new mobile devices. The results will be briefly reported later in the paper.

The rest of paper is organized as follows: in Section 2, we develop the collaborative half-frame decoding scheme. We perform algorithmic optimization for better energy efficiency in Section 3. In Section 4 we describe the MobiUS system architecture and implementation details. System evaluation results are presented in Section 5, followed by in-depth discussions in Section 6. We talk about related work in Section 7. Finally, Section 8 concludes the paper and highlights our future work.

[1]Here, real-time playback implies at least 15 frames per second (fps) for typical mobile video, and normally 24 fps is expected, depending on how video clips are produced.

2. COLLABORATIVE HALF-FRAME DECODING

There are many possible ways to achieve video playback on two screens. We elaborate all of them, from more straightforward to more subtle solutions, and discuss their feasibility and pros and cons. To facilitate the presentation, we name the two mobile devices M_A and M_B and assume M_A is the content host. Without loss of generality, we assume M_A is sitting on the left and M_B on the right. Recall that our primary target is to achieve real-time playback of the doubled resolution video on the computation constrained mobile devices.

2.1 Full-Frame Decoding-based Approaches

The most straightforward solution would be either to let M_A decode the whole frame, display the left half-frame and send the decoded right half-frame to M_B via network, or to let M_A send the whole bitstream to M_B and both devices perform full-frame decoding but display their own respective half-frames. We refer to these two methods as *thin client model* and *thick client model*, respectively.

The pros of these two methods are their simplicity in implementation. However, for the thin client model, the computing resource of M_B is not utilized and its huge bandwidth demand is prohibitive. For example, it would require more than 22 Mbps to transmit a 24 frame per second (fps) 320×240 sized video using YUV format.[2] The energy consumption is highly unbalanced between the two devices and therefore would lead to short operating lifetime since the application will fail if either device runs out of battery. The thick client model requires much less bandwidth and utilizes the computing power of both devices. However, it abuses the computing power to decode more content than necessary, which can lead to both devices not achieving real-time decoding of the double resolution video. The reason is that the computational complexity of video is in direct proportional to its resolution if the video quality remains the same, but mobile devices are usually cost-effectively designed such that their computing power is just enough for real-time playback of a video whose resolution is no larger than that of the screen.

In conclusion, the full-frame decoding-based approaches are not feasible.

2.2 Half-Frame Decoding-based Approaches

Another category of solutions is to let each device to decode their corresponding half-frame. These methods aggregate and utilize economically both devices' computing power. There are two alternative approaches that differ in transmitting whole or only partial bitstreams. We refer to the two approaches as *whole-bitstream transmission* (WTHD) and *partial-bitstream transmission* (PTHD), respectively.

Both approaches may reduce the decoding complexity since only half-frames need to be decoded. However, as will be elaborated shortly, to achieve half-frame decoding is challenging and requires substantial modifications to the decod-

[2]Bandwidth requirement will double if RGB format is used.

ing logic and procedure. PTHD saves about half of the transmission bandwidth, which is significant, as compared with WTHD, but adds to implementation complexity with the extraction of the bitstream parsing process to strip out the partial bitstream for M_B.

While both schemes are feasible, from an energy efficiency point of view, PTHD is more preferable since there is no bandwidth waste, i.e., only the bits that are strictly necessary are transmitted, which directly translates to energy savings. We adopt PTHD in our work. More specifically, we pre-parse the bitstream into two partial ones, stream one resulting bitstream to the other device and make both devices perform collaborative decoding. In the rest of the paper, we focus on practical ways to achieve and improve PTHD while bearing in mind the constraint of energy efficiency.

As a short summary, we compare all the above mentioned approaches in Table 1.

Scheme	Comput. complexity	BW efficiency	Impl. complexity	Feasibility
Thin/C	High/Low	Worst	Simple	No
Thick/C	High	Bad	Simple	No
WTHD	Low	Bad	Complex	Possible
PTHD	Low	Good	Complex	Preferred

Table 1: Summary and comparison between different approaches to playback video on two devices.

2.3 Challenges of Half-Frame Decoding

Note that, concluding the latter two approaches are feasible, we have assumed the ability to perform half-frame decoding. Seemingly straightforward, half-frame decoding is actually far more difficult than that we might have imagined, because of the inherent temporal frame dependency of video coding caused by prediction, and possible cross-device reference (i.e., reference to the half-frame on the other device) caused by motion. In the worst case, one may still need to decode all the frames in whole from the previous anchor frame (which is independently decodable) in order to produce the correct references for some blocks in a very late frame.

2.3.1 Background on Video Coding

A video sequence consists a series of frames, each of which is simply an image. Therefore, each video frame possesses strong spatial correlation as an image does. As the capturing instant of neighboring frames are very close to each other, video frames also exhibit strong temporal correlation. The basic logic of video coding is to maximally strip off such spatial and temporal correlation and so compress the video.

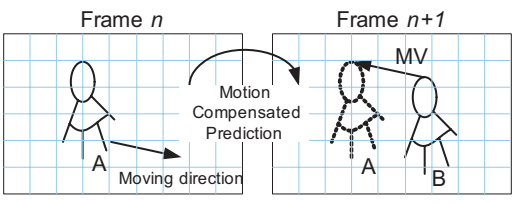


Figure 2: Motion compensated prediction.

Since either the camera or the objects in the scene may move, motion compensated prediction (MCP) is typically used to better exploit the temporal correlation. As shown in Figure 2, due to its movement, the same object is captured at different positions across frames, e.g., at position A in the n^{th} frame and at another position B in the $(n+1)^{th}$ frame. Evidently, the best prediction to the object is offset by a motion vector (MV) pointing from position B to A. This is exactly what the MCP does. The motion vector is obtained through a motion estimation process. After MCP, the resulting residual signal is further compressed using a procedure akin to JPEG image compression process.

Obviously, MCP creates temporal frame dependencies, that is, to decode a later frame, its previous frame (called the reference frame) must have been reconstructed (decoded). Such dependency is *recursive* in nature unless it reaches an anchor frame which is independently encoded/decoded (i.e., no MCP). The first frame is always an anchor frame because there is no previous frame available. Other anchor frames are periodically inserted to break the dependency chain and provide a random access capability.

To reduce the complexity, a video frame is divided into smaller blocks which are processed sequentially. MCP is typically carried out at 16×16 macroblock level, with one motion vector for one macroblock in MPEG-2 [2] .[3] Moreover, motion vector is typically limited to certain ranges but the ranges vary in different video coding standards. Although the principle of video coding is simple, the computational complexity is inherently high because of the huge volume of video data.

2.3.2 Challenge Arises from Motion

While the recursive temporal frame dependency certainly creates barriers for parallel decoding along the temporal domain, it indirectly affects our task that hopes to perform parallel decoding in the spatial domain, i.e., the two devices M_A and M_B decode the left and right half-frames, respectively. The real challenge arises from motion, but is worsened by the recursive temporal dependency.

Due to motion, an object may move from one half-frame to the other half-frame in subsequent frames. Therefore, dividing the whole frame into two half-frames creates a new *cross-boundary reference* effect. That is, some content of one half-frame is predicted from the content in the other half-frame. This implies that in order to decode one half-frame, one has to obtain the reconstructed reference of the other half-frame. Still take as example the object in Figure 2. Position A is in the left half-frame that belongs to M_A and position B is in the right half-frame that belongs to M_B. In order to decode the object at position B in the $(n+1)^{th}$ frame, M_B needs the reference data at position A in the n^{th} frame, which is unfortunately not available since it is not supposed to decode that in the previous frame.

One seemingly possible rescue to the problem caused by cross-boundary reference is to ask M_B to be more diligent and decode the content at position A in the n^{th} frame as well, assuming it can pre-scan the bitstream and know the content at position A will be required in future. Unfortunately, due to the recursive nature of frame dependency, M_B has to further decode some other parts in the left half-frame in the $(n-1)^{th}$ frame in order to be able to decode the content at position A in the n^{th} frame, so on and so forth. As said, in the worst case, M_B has to decode all the whole

[3]Recent video coding standards like H.264 [3] may go to finer block levels, but the general procedure is the same.

frames from the previous anchor frame in order to correctly decode a very late frame.

2.4 Collaborative Half-Frame Decoding

Albeit challenging, there are still methods to perform efficient half-frame decoding. Notice that the reference is always the decoded previous frame, therefore, it either exists on the left half-frame or the right half-frame. Furthermore, since the two mobile devices have communication capabilities, with minor extra effort we can make available the reference data by asking the two devices to help each other, i.e., transmitting the missing references to each other. In other words, half-frame decoding can be achieved through *cross-device collaboration* (CDC). The rationale of cross-device collaboration arises from the following two fundamental facts.

2.4.1 Fundamental Facts

A) Markovian effect of MCP. Although recursive, the temporal frame dependency exhibits a first-order Markovian effect. That is, a later frame only depends on a previous reference frame, no matter how the reference frame is obtained. This is the fundamental reason that we can perform cross-device collaboration and obtain correct decoding result.

B) Highly skewed MV distribution. The motion vector distributions and their corresponding cumulative distribution functions for a test sequence *BestCap* are shown in Figure 3. We inspect the motion vector distributions for the whole frames as well as those for only the two columns of macroblocks (referred to as the guardband) near the half-frame boundary. Only the horizontal component of motion vectors are shown since it is the only component that cause cross-device references. From the figures, we can see that the motion vector distribution is highly skewed, centered at the origin (0,0) and most motion vectors are indeed very small. More than 80% of motion vectors are smaller than 8, which is the width of a block. In fact, the distribution of motion vectors can be modeled by a Laplacian distribution [4]. This fact implies that the traffic involved in the cross-device collaboration is likely to be affordable.

2.4.2 Collaborative Half-Frame Decoding with Push-based CDC

The straightforward idea of collaborative half-frame decoding (CHDec) is to let each device decode their respective half-frame and request the missing reference data from the other device. However, there exists one practical barrier if the cross-device helping data is obtained through natural on-demand pulling: This on-demand pull-based request of the missing reference data incurs extra delay and stalls the decoding process accordingly. This will have severe negative impact on the decoding speed and the overall smoothness of the playback. For instance, for a 24 fps video, the average frame period is about 42 ms. According to our experiments with WiFi (on Dopod 838 PocketPC phone), the round-trip time (RTT) is typical in the range [10, 20] ms. Considering the extra time to prepare the helping data, the on-demand request scheme will prevent timely decoding and is therefore not practical.

To overcome this barrier, we change the on-demand pull-based scheme to a *push-based* cross-device helping data delivery scheme by looking ahead one frame. The purpose of looking ahead is to analyze what the missing data will be for both devices through motion vector analysis. In this way, we know in advance what reference data are missing for both devices and ensure this data will be sent.

We design the collaborative decoding scheme as follows: before decoding the half-frame of the n^{th} frame, we let the content hosting device look ahead by one frame through a light-weight pre-scanning process and perform motion analysis on the next frame ($(n+1)^{th}$ frame). We mark those blocks that will reference the other half-frame (for both devices) and record their positions and associated motion vectors. Based on such information, we can easily infer the exact missing reference data at the other party. Then we proceed to decode the respective half-frame but skip those marked blocks since they will not have the reference data, and prepare the helping data in the meantime. The helping data will be sent out immediately or buffered till the end of the decoding process and sent in a batch. Then each device will perform quick rescue decoding to those marked blocks.

As will be shown in the experimental results, with the collaborative half-frame decoding and push-based CDC data delivery scheme we can achieve real-time playback requirement on the computation capability constrained mobile devices.

3. ENERGY EFFICIENCY OPTIMIZATION

Although the collaborative half-frame decoding scheme fulfills the real-time requirement, as stated before, it is also highly desired to prolong the operating time by minimizing the energy consumption since mobile devices are typically battery operated. Because the CDC traffic is used to maximally reduce the computation, a natural question is if there exists a better trade-off among the computation reduction and the volume of the resulting CDC traffic.

3.1 Motivating Observation

In the CHDec scheme, all the missing reference contents are transferred between the two mobile devices. This may incur a large bandwidth consumption and cause more energy consumption. In Table 2, we list the percentage of boundary blocks (i.e., the column of macroblocks neighboring the half-frame boundary) that perform cross-boundary reference and their corresponding CDC traffic for the three test sequences. Note that the bandwidth requirement of CDC traffic is not proportional to the percentage of cross-device reference blocks because the motion vectors are different even though they are all referencing content on the other device. Clearly, the bandwidth requirement of the helping traffic is relatively high, reaching half of the bandwidth required for sending the half bitstream, because the cross-boundary referencing is still frequent.

Sequence	CHDec		GB-CHDec	
	CD Ref	BW Req	CD Ref	BW Req
BestCap	22.8%	253 kbps	3.4%	76.9 kbps
SmallTrap	26.2%	192 kbps	1.3%	30.6 kbps
Liquid	20.7%	231 kbps	2.5%	53.2 kbps

Table 2: Percentage of boundary blocks that require cross-device collaboration and their corresponding bandwidth requirement. Collected over the first 600 frames of three test sequences. The resolution is 480×320 and the bit rate is 1Mbps.

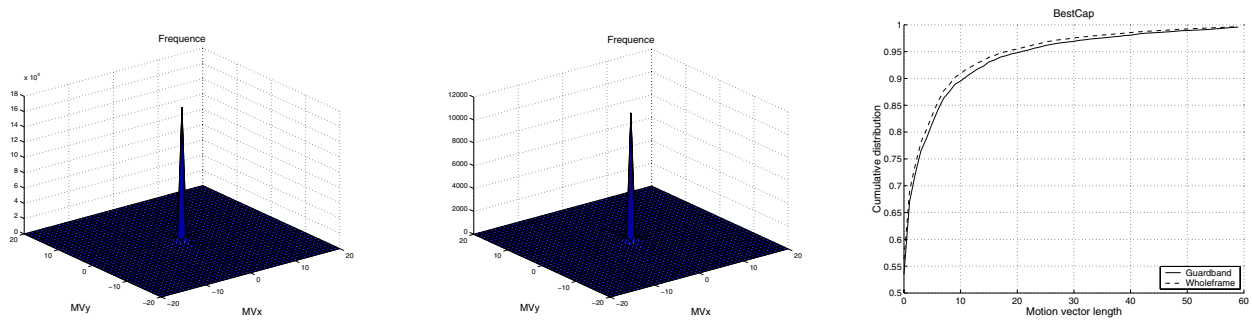


Figure 3: Plots related to motion vector distribution for 600 frames of the test sequence BestCap. The video resolution is 480×320. From left to right: distribution of horizontal component of motion vectors for the whole frames, for the boundary columns near the half-frame boundary (i.e., guardband), and their corresponding cumulative distribution functions.

Since WiFi consumes energy heavily, the CDC traffic should be reduced. As advocated in [5], adaptive use of multiple radio interfaces can lead to significant energy savings. However, the extent to which the adaptation can be made is subject to the applications' specific requirement. In their streaming experiments, the ideal case is to use the "Bluetooth-fixed" policy which always uses Bluetooth. The fundamental reason, as pointed out by the authors, is that the streaming data rate is low enough for the Bluetooth's throughput to be capable of. Nevertheless, if a higher data rate is required, then it will need to activate WiFi for most of the time. This implies that the CDC traffic has to be reduced to be eligible for adaptive use of multiple radio interfaces for better energy efficiency. These considerations lead to the design of the guardband-based collaborative half-frame decoding technique to be presented below.

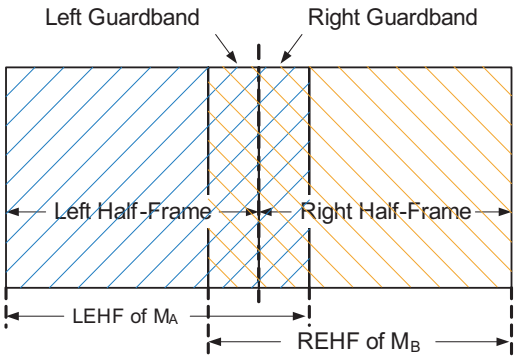


Figure 4: Illustration of left and right guardbands, left and right expanded half-frames (LEHF, REHF).

3.2 Algorithmic Optimization

From the motion vector distribution shown in Figure 3, we see that more than 90% motion vectors are smaller than 16, which is the width of a macroblock. This implies that more than 90% of boundary blocks can be correctly decoded without incurring any CDC traffic if we let each device to decode an extra column of macroblocks across the boundary. Such extra decoding area is called the *guardband* hereafter.

With this observation in mind, we design a *guardband-based collaborative half-frame decoding* scheme, in which each device not only decodes its own half-frame, but also decodes an extra guardband so as to reduce the CDC traffic. We refer to the half-frame area plus the extra guardband as expanded half-frame (EHF), as illustrated in Figure 4 where LEHF and REHF represents EHF for the left device M_A and right device M_B, respectively.

To see how much the guardband helps to reduce the CDC traffic, we also list in Table 2 the percentage and the corresponding CDC traffic of boundary blocks that still have cross-device references when the guardband is also decoded. We see that the guardband indeed significantly reduces the CDC traffic to as large an extent as 75%.

One might argue that the situation is actually not improved since we would need CDC for decoding the boundary blocks in the guardband. The argument is correct if we need to correctly decode the whole guardband correctly. But the introduction of guardband changes the situation: we do not have to ensure the guardband to be completely and correctly decoded. The subtle difference against the half-frame decoding case is that blocks of guardbands are not to be shown on screen while those belonging to the half-frame will. In fact, we only need to decode those guardband blocks that will be referenced, which can be easily achieved via the motion analysis process on the next frame. Furthermore, from the two fundamental facts of video coding presented in Section 2.4.1, we derive a third (implicit) fact: *multiplicative decaying motion propagation effect*, which says that the guardband blocks of one frame that are referenced by some boundary blocks of the next frame will have a much lower probability to reference to the area exterior to the guardband of its previous frame. To be concise, we omit the derivation.

Our guardband-based collaborative half-frame decoding scheme works as follows: as CHDec does, we also look ahead by one frame, perform motion analysis and adopt push-based CDC traffic delivery. The major difference lies in that each device now needs to decode the extra guardband. Further, the blocks in the guardband are differentiated according to their impact on the next frame: those not referenced by the next frame are not decoded at all; those referenced by the guardband blocks of the next frame are best-effort decoded, i.e., decoding without incurring CDC and no insurance of the correctness; and those referenced by the half-frame blocks of the next frame are ensured to be correctly decoded, resorting to CDC when necessary.

As a remark, the purpose of guardband is not to com-

pletely remove the need for cross-device collaboration, but to achieve better trade-off in energy efficiency with significant reduction of the collaboration traffic at cost of slightly more computations. From Table 2, we can calculate that to correctly decode the whole one-macroblock-wide guardband (which represents the worst case since in practice some not referenced blocks need not to be decoded at all), the extra computational cost is about 7%, but the average associated CDC traffic savings is about 76%, which is favorable even Bluetooth is used.

In this work, we empirically set the width of guardband to be one macroblock column. Such decision arises from the implementation simplicity because all the motion compensation is conducted on a macroblock basis in MPEG-2, and is dictated by the rule that real-time playback requirement must be observed. In fact, if we use a two-macroblock-wide guardband, it will incur another 7% computation overhead (in worst case) but brings in only about additional 10% CDC traffic reduction and is therefore not very beneficial. A systematic approach to achieve the optimal trade-off would exhaustively search or develop a model that allows systematically evaluation for all possible guardband widths. Another more proper way is to make it adaptive: we can look ahead for multiple frames (e.g., a group of picture, GOP), perform motion analysis and determine the optimal guardband width for that specific GOP. However, a prerequisite condition is the knowledge of energy consumption characteristics of WiFi and CPU, which could vary with different mobile devices. This fact suggests that a profile-based approach might be feasible.

4. SYSTEM ARCHITECTURE AND IMPLEMENTATION

Unlike traditional loosely coupled distributed systems, e.g., those for file sharing, the new better-together application paradigm of MobiUS requires a tightly coupled system because it involves not only the networking, but also the computing, shared states and data, and other resources. For the specific together-viewing video application, we have identified the following common modules: proximity detection, synchronization, resource coordination. We leave out those modules such as access control that are otherwise important in traditional loosely coupled distributed systems, thanks to the natural close proximity setting where the devices have to be physically close.

The system architecture of MobiUS is depicted in Figure 5, where we position the common modules as a middleware layer, sitting on top of a traditional operation system, and lay the together-viewing video application into the application layer. We elaborate their roles and implementation details below.

4.1 The Middleware Modules

4.1.1 Close Proximity Networking

The bottom substrate of the MobiUS system architecture is the close proximity networking layer, which sits directly on top of the traditional networking layer but further abstracts popular wireless technologies into a unified networking framework. It also incorporates the possible physical connection (e.g., through wire or hardware interface). The target of this close proximity networking layer is to automatically set up a network between two mobile devices [6,7],

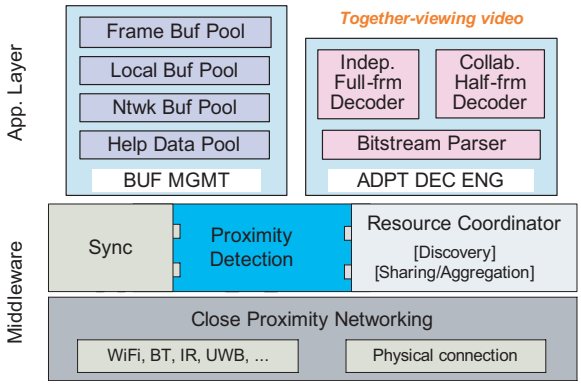


Figure 5: MobiUS system architecture.

without involving the users, such that resource discovery and aggregation can be effectively performed.

We adopt the EasySetup work [8] developed in our group that manages different wireless technologies into a unified framework. Based on this, MobiUS can use both Bluetooth and WiFi, and can save energy by dynamically switching between them, depending on the traffic requirements.

4.1.2 Proximity Detection

The proximity detection module is one of the most important components whose primary function is to ensure a close proximity setting. Depending on different application requirements, vague or precise proximity information can be obtained at different system complexities. For example, for normal applications, we can use a simple radio signal strength based strategy to determine a rough estimation of distance [9], involving only wireless signals. Typically, the radio signal strength is indicated by received signal strength index (RSSI) which is usually available from wireless NIC drivers. If high precision is desired, with additional hardware we can use both wireless signals and acoustic or ultrasonic signals to obtain precision up to a few centimeters [10, 11].

In the special application of together-viewing video experience, the proximity detection is mainly for the purpose of user convenience. Therefore, we have a low precision requirement that only needs to determine the arrival or departure of the other device. We found simple RSSI-based strategy suffices this requirement. Lacking of a universal model that can tell the proximity of two devices using solely the RSSI and noticing the fact that together-viewing is conducted with explicit intention, we adopt a simple heuristic: when RSSI is high (e.g., -50 dbm of WiFi signal on Dopod 838), we inform the user that another device is nearby and the user would confirm or reject the together-viewing request. Notification will be sent to the resource coordinator module if confirmed. When RSSI reduces significantly (under the normal quadratic signal strength decaying model), we simply conclude that the other device has left and inform the resource coordinator module accordingly.

We plan to look into acoustic signalling to achieve higher accuracy in future work.

4.1.3 Synchronization

The prominent "together" feature of MobiUS requires the mobile devices to operate in synchronization. The synchro-

nization can be achieved either at the application level or the system level, at different difficulties. We can rely on either network time protocol [12] or the fine grained reference broadcasting synchronization mechanism [13] to synchronization the mobile devices to a high precision, e.g., within one millisecond. However, such system level synchronization is difficult to achieve and is sometimes not necessary for the specific applications, especially the multimedia applications. For MobiUS, we adopt an application level synchronization strategy, which can satisfy our requirement and is easy to implement.

For the specific together-viewing video application, since we are displaying each video frame on the both screens, the two video playback sessions need to remain synchronized at the frame level. This implies that the tolerable out-of-sync range is about one frame period, e.g., 42 milliseconds for 24 fps video. Considering the characteristics of the human visual system, the tolerable range can actually be even larger. It is well known in the video processing field that people will perceive a continuous playback if the frame rate is above 15 fps, which translates to a 66 millisecond tolerable range.

Note that the target of the synchronization engine is to sync the video display, not the two devices. Towards this end, we use the video stream time as the reference and rely on the estimation of round-trip-time of wireless signals to sync the video playback. The content-hosting device performs RTT measurements; once obtaining a stable RTT, it will notify the client to display the next frame while waiting half RTT before displaying the same frame. Such RTT-based synchronization procedure is performed periodically throughout the video session. In our experiment, a typical stable RTT value is within 10 milliseconds and the RTT value stabilizes quickly in a few rounds.

4.1.4 Resource Coordinator

The resource coordinator typically has double roles: one role is to discover the resources including information resource, such as what files are being shared, and computing resources like if the other device is capable of performing certain tasks. The other role is to coordinate the resources to collaboratively perform a task and to achieve load balance among devices by shifting some tasks around.

In the together-viewing video application, we have developed a simple XML-based resource description schema for resource discovery purposes, that says what video files are available on a device and their basic features such as the resolution, bit rate etc. It also reveals some basic system configuration information such as the processor, system memory (RAM) and the registered video decoder. We have not fully exploited the resource coordinating functionality, i.e., the second role. Currently, in our system, the resource coordinating module does only one simple thing: check capability of a newly added device and inform the content hosting device about the arrival (if it passes a capability check) or departure of the other device. We plan to make it monitor the system energy drain and to dynamically shift partial decoding tasks between the devices.

4.2 Together-viewing Video Specific Modules

4.2.1 Buffer Management

The buffer management module manages four buffer pools: one frame buffer pool, one helping data buffer pool, and two bitstream buffer pools, namely local bitstream buffer (LBB) pool and network bitstream buffer (NBB) pool.

The frame buffer pool contains several buffers to temporarily hold the decoded video frames if they have been decoded prior to their display time. Such buffers sit in between the decoder and the display and are adopted to absorb the jitter caused by the mismatches between the variable decoding speed and the fixed display interval. The helping data buffer pools consists of two small buffers that hold and send/receive the cross-device collaboration data to the other device.

The two bitstream buffer (LBB and NBB) pools are to hold the two half-bitstreams that are separated by the pre-parser module in the adaptive decoding engine, for itself and the other party, respectively. The bitstream in NBB pool will be transferred to the other device. Note that for the content hosting device, two bitstream buffer pools are used. However, only one of them (i.e., NBB pool) is required for the client device. The reason of adopting the NBB pool at the content hosting device is three-fold: 1) to enable batch transmission (using WiFi) for energy saving; 2) to allow a fast switch back to single screen playback if the other device leaves; and 3) to emulate the buffer consumption at the client device so that when performing push-based bitstream delivery, the previously sent but unconsumed bitstream data will not be overwritten. Based on the fact that the two devices playback synchronously, the content hosting device can know exactly what part of the receiving buffer of the client can be reused in advance.

Throughout our implementation and optimization, we found that using a dedicated buffer management module is very preferable. It clarified the working flow and helped to remove all the memory copies, which is very costly on mobile devices. We overwhelmingly use the pointers throughout the process. Moreover, using multiple buffers helps greatly to the overall performance by mitigating the dependency among several working threads.

4.2.2 Adaptive Decoding Engine

The adaptive decoding engine, which is the core of MobiUS, consists of three modules, namely the bitstream pre-parser module, the independent full-frame based fast DCT-domain down-scaling decoding module [14] and the guardband-based collaborative half-frame decoding module.

The bitstream pre-parser parses the original bitstream into two half bitstreams prior to their decoding time and also extracting the motion vectors. The resulting two half bitstreams are put into the two bitstream buffers in the local buffer pool and the network buffer pool, respectively.

As indicated by the resource coordinator module, if only single display is available, the independent full-frame decoding engine will be called, which retrieves bitstreams from both bitstream buffers in LBB and NBB pools and directly produces a down-scaled version of the original higher resolution video to fit the screen size, eliminating the explicit downscaling process. Note that for single display case, the decoded frame needs to be rotated to match the orientation of video to that of the screen. The rotation process can be absorbed into the color space conversion process. If two screens are available, the guardband-based collaborative half-frame decoding module will be activated. The content hosting device will decode the bitstream from buffers in the LBB pool and send those in the NBB pool to the other de-

vice and, correspondingly, the other device will receive the bitstream into its NBB pool and decode from there. Note that in this case, the two mobile devices must work concurrently and send to each other the helping data periodically on a per-frame basis. The two decoding engines can switch to each other on the fly, and automatically under the indication of the resource coordinator.

In our experience, we found separating the networking, decoding and display into different threads is of crucial importance. Without using multiple threads, the benefit of using multiple buffers is rather limited. Moreover, it is important to assign correct priority levels to different threads. In our implementation, we assign a higher priority (Priority 2) to the display thread and networking thread, since we need ensure the synchronous display of the two devices and do not want the decoding process to be blocked because of waiting for bitstream or helping data. The decoding thread is assigned a lower priority (Priority 1) by default, which is still higher than other normal system threads, but will be dynamically changed if at risk of display buffer starvation. For sporadic events like proximity detection, we also assigned Priority 2 to ensure prompt response to the arrival or departure of the other device.

5. EXPERIMENTAL RESULTS

We have implemented the MobiUS system architecture and the several common components, and built the together-viewing video application on top of it, using the more energy efficient guardband-based collaborative half-frame decoding scheme. We used two off-the-shelf PocketPC phones, HP iPAQ rw6828 and Dopod 838 (HTC Wizard). Both devices are based on Microsoft Windows Mobile Version 5.0, Phone Edition, have WiFi and Bluetooth radios, QVGA resolution (320×240) display, 64 MB RAM. The only difference is that they have different processors: HP rw6828 features a more powerful Intel XScale 416 MHz processor while Dopod 838 is equipped a 195 MHz OMAP 850 processor which is much less powerful.

Recall that we have three primary requirements: real-time synchronous playback, energy efficiency, and dynamic adaptation. In this section, we first report some experimental results with regard to the first requirement to confirm the effectiveness of our design. We then present the experimental results on energy efficiency of the proposed collaborative decoding schemes. Due to lack of alternative tools, we have chosen to experiment by fully charging the battery and measure the overall session lifetime for different decoding schemes.

We also achieved the third requirement, as can be seen from the recorded demo on our website (`http://research.microsoft.com/wn/mobius.aspx`). We can immediately detect the arrival of the other device, but have a latency about 2 or 6 seconds to detect the departure for HP iPAQ rw6828 and Dopod 838 mobile phones, respectively. This is the actual latency caused by the NIC drivers of the mobile devices which reports the RSSI on a 2 or 6 seconds basis. This latency can be greatly shortened if we are allowed to modify the driver. For instance, we achieve almost instantaneous updated RSSI information on a laptop with a modified NIC driver for which we have source code access.

5.1 Decoding Speed

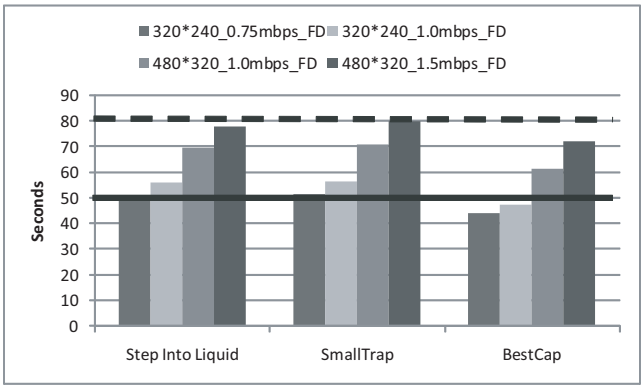
The prerequisite condition to the real-time synchronous playback is that the decoding speed is fast enough. To measure the decoding speed, we perform free-run tests, that is to let the device run as fast as it can. In real applications, proper timing will be applied to limit the decoding speed from running too fast.

Three test sequences were used in our experiments, namely SmallTrap, BestCap, StepIntoLiquid, which we have put on our website as well. Notice that the three selected test sequences are very representative in terms of video content. They consist of simple and complex scenes, low and high textures, slow and fast motions, abrupt and gradual scene changes, etc. Our primary target is to test the decoding speed of the doubled resolution (480×320) video on mobile devices. For comparison purpose, we also generated corresponding videos at normal resolution (320×240), the same as that of the screen, to benchmark the devices' intrinsic video playback capability. Since the bit rate of video has impact at the decoding speed, we used different but high bit rates to ensure good visual quality and also stress the tests. Specifically, we use 750 kbps and 1 Mbps for normal resolution video and use 1 Mbps and 1.5 Mbps for doubled resolution video. All the tests are performed on the first 1200 frames, which equals to 50 seconds playback time if interpreted at the frame rate of 24 fps or 80 seconds at the frame rate of 15 fps. To facilitate examination, in the figures we also draw two respective horizontal lines to indicate the deadlines for 24 fps and 15 fps real-time playback.

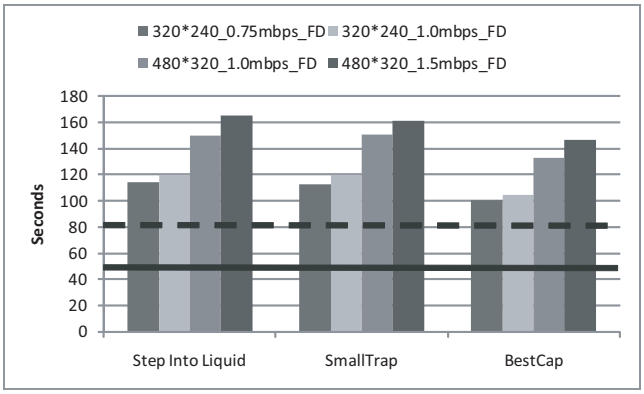
The experimental results are shown in Figure 6 and Figure 7. In the legends of the figures, FD, FFD, HFD and EHFD stands for full-frame decoding, fast full-frame downscaling decoding [14], collaborative half-frame decoding and expanded collaborative half-frame (i.e., guardband-based) decoding, respectively.

From Figure 6 we can see that, as said before, the mobile devices are indeed cost-effectively designed and are just able to meet the real-time playback requirement for videos at the same resolution of the screen. For normal resolution (320×240), the HP iPAQ rw6828 can achieve an average speed about 23 fps for 750 kbps video and about 20 fps for 1 Mbps video. However, the Dopod 838 is much weaker and can hardly meet the real-time requirement for the stress testing sequences. It achieves only about 10 to 11 fps. We further investigated the problem and found that the graphics rendering engine on mobile devices is weak and impairs significantly the overall achievable playback speed. This is especially the case for Dopod 838 since the HP iPAQ rw6828 supports DirectDraw which leads to a lower performance penalty.

However, even the HP iPAQ rw6828 can not meet the real-time requirement for the doubled resolution 480×320 videos. It achieves about 15 to 20 fps for different test sequences. In other words, it can ensure the real-time playback only if the video is produced at 15 fps. The performance of the Dopod 838 is far worse. It achieves only about 7 or 8 fps for the doubled resolution videos. We want to point out that there is big penalty in the display thread for both phones because of the rotation operation needed to match the orientation of the videos and the screen, for the full-frame decoding case. The extra downscaling process that converts a 480×320 video to its 320×240 version also incurs a performance penalty, even though it is already optimized by integration with the color space conversion procedure (i.e., when converting YUV to RGB format).

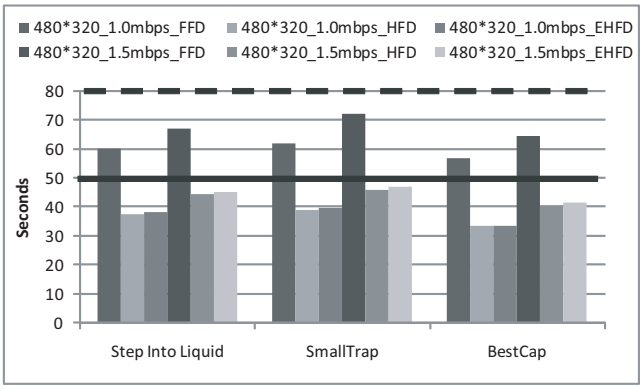


(a) HP iPAQ rw6828

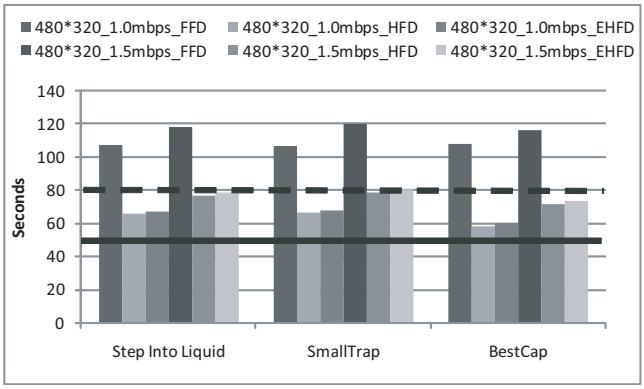


(b) Dopod 838

Figure 6: Benchmark of the two PocketPC phones' video playback capability. The thick solid and dashed horizontal lines indicate the time lines of real-time playback for 24 fps and 15 fps, respectively.



(a) HP iPAQ rw6828



(b) Dopod 838

Figure 7: Video playback performance of our proposed schemes on two PocketPC phones. The thick solid and dashed horizontal lines indicate the time lines of real-time playback for 24 fps and 15 fps, respectively.

In Figure 7, we show the performance of the proposed collaborative half-frame decoding and its improved guardband-based version. From the figure, we see that we can essentially achieve real-time playback on the HP iPAQ rw6828 for the doubled resolution video, which significantly outperforms the benchmark case. Even on the Dopod 838, we can achieve more than 15 fps in both collaborative half-frame decoding cases. One of our design goal is that MobiUS should fully function when only one device is available. Therefore, we also determine the decoding speed of our full-frame based fast DCT-domain downscaling decoder. The fast full-frame decoding achieves about 17 to 20 fps on the HP iPAQ rw6828 phone and only about 10 to 11 fps on the Dopod 838. While already significantly better than the benchmark case, it still needs improvement to reach real-time. We investigated the problem and found that the major reason is still the rotation process to match the orientation.

Comparing Figure 6 and Figure 7, we can further observe that 1) both collaborative half-frame decoding schemes significantly improve the decoding speed; 2) the guardband-based scheme is only slightly slower than the half-frame decoding case and the margin is less than 7%, due to the best-effort decoding strategy for the guardband blocks; 3) the rendering occupies a significant portion of the overall process time, especially when a rotation process is required.

5.2 Synchronization

As stated before, we have adopted a simple RTT-based synchronization mechanism. To see how it performs, we record the timestamps at which video frames are displayed at each device. Because the clocks of the two devices are not synchronized, direct comparison between the corresponding time stamps is not meaningful. Instead, we calculate the intervals between consecutive frames on each device and compare the resulting intervals, which is shown in Figure 8.

From the figure, we can see that most of the time, the synchronization mechanism works well despite a few spikes that are due to the periodical synchronization. However, there are a flurry of large discrepancy among frame display intervals between Frame 50 to Frame 110. Further investigation revealed that those frames correspond to a large scene change and have very high motion, which caused the decoding thread to temporarily grab a higher priority (in order to avoid display buffer starvation) and the display thread failed to get the computing slices timely. The immediate consequence is that some frames are more seriously delayed and the device will try to catch up in the following display periods. Since the catch up process is completely local to each device, it results in large oscillation in the display interval between the two devices. Fortunately, the discrepancy is not significant. It is tolerable because the human visual system

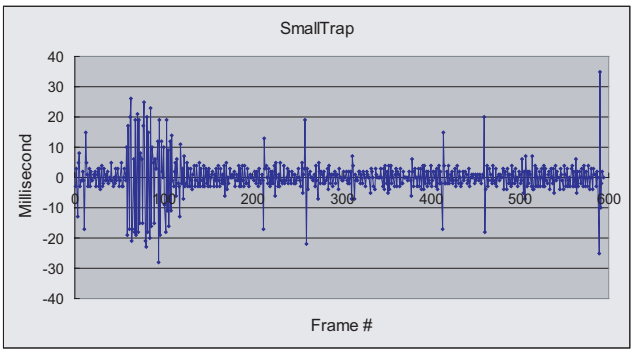


Figure 8: Comparison of display intervals at two devices, for the first 600 frames of SmallTrap sequence.

is far less sensitive for such slight asynchronism, especially when the motion is large.

5.3 Energy Efficiency

The energy efficiency is usually reflected through a high accuracy measurement of current and voltage of mobile devices, and requires special tools or hardware rewiring. In this work, we choose an alternate experiment scheme by fully charging the battery and measure the overall session lifetime for different decoding schemes. While the results are certainly less quantitative, we believe it is still very informative.

Our target here is to see the energy efficiency advantage of the proposed collaborative decoding scheme. We therefore design comparative experiments that only differs in the decoding schemes. In this case, we turned off WiFi. To show the real performance, we also conduct another set of experiments with WiFi on. Note that because none of our handy mobile devices support power saving mode for WiFi in ad hoc mode, we are not able to conduct experiments on the energy savings of dynamic switching between WiFi and Bluetooth even though our implementation does support this.

Decoding scheme	WiFi	Lifetime (seconds)
Full-frame	OFF	16438
	ON	7482
Half-frame	OFF	23736
	ON	8375

Table 3: Lifetime measurement for different experimental settings on the HP iPAQ rw6828.

By examining the first and the third row of Table 3, we can see that our collaborative half-frame decoding scheme indeed leads to significant energy savings. The lifetime increased by nearly 50%. Nevertheless, we want to point out that we have turned on the LCD backlight and set it to the maximum brightness level. Since backlight consumes a large percentage of energy [15], the actual energy efficiency of our proposed scheme is actually much more significant than that reported here.

Table 3 also reveals, by comparing the cases when WiFi is on or off, that WiFi is most energy starving. It consumes more than half of the overall energy. As a result, the advantage of our collaborative decoding scheme is also diminished, as can be seen from the second and the fourth row. We noticed that, in the half-frame decoding case, the client still has 18% energy left when the server dies, while it has only 4% left in the full-frame decoding case. These observations suggest that dynamic switching between different radio interfaces is crucial and that better load balancing schemes should be designed.

6. DISCUSSIONS

6.1 Further Optimization Opportunities

6.1.1 Computation Saving

We have performed algorithmic optimization in achieving the real-time doubled resolution video playback. As with any optimization applications, another large space is through code optimization. In fact, we have implemented two versions of our prototype, one implemented in C++ and one is in C. The one written in C is about 25% faster than that written in C++. One possible reason is that our C++ implementation emphasizes portability across different video applications such as transcoding.

We further performed complexity profiling of our code. We found the color space conversion (i.e., to convert YUV format to RGB16 format) consumes about 30% of the overall time, which is the most computational expensive module. The second and third expensive modules are inverse DCT module and motion compensation modules. Since all these modules involve only heavy but regular computation, they are excellent targets for assembly optimization.

6.1.2 Collaboration Traffic Reduction

The guardband-based collaborative decoding scheme significantly reduces the collaboration traffic. However, since the bandwidth and the corresponding power consumption is expensive, we want to further reduce the bandwidth consumption. One way is to use a larger guardband, but as previous mentioned, the gain of using a larger guardband (e.g., wider than one 16-pixel width) becomes marginal. Therefore, it is not a good tradeoff. Also, the available computing resources may not support a larger guardband because of the real-time decoding requirement. One alternate way is to apply simple compression. It is fairly easy to obtain several times compression at light computation overhead. Yet a third method that can also be conditionally applied is to apply error concealment technique. Since video signal has strong spatial and temporal redundancy, error concealment is usually effective when the number of erroneous pixels are small. Therefore, this is a potential method to be applied when there are few pixels need to be transmitted across devices.

6.1.3 Energy Consumption Reduction

We have explored the tradeoff between computing (decoding) and networking (cross device collaboration traffic) while fulfilling synchronous real-time playback requirement. Yet it is noticed that the screen backlight is also energy hungry. As evidenced by others work, it is possible to save the screen backlight energy consumption through the gamma adjustment of the video signal [15].

Moreover, in systems with dynamic voltage scaling capability [16, 17], half-frame decoding can be better exploited because of the good predicability of the frame complexity

and the larger optimization room offered by the efficient half-frame decoding. We may keep the CPU in a lower power state throughout the together-viewing session with better scheduling.

6.2 Service Provisioning

Most of the complexity of the collaborative decoding scheme arises from the cross-device reference. We took pain in order to ensure the incremental deployment, that is, users can directly download higher resolution video files from the Internet and achieve the together-viewing experience immediately. Nevertheless, there are alternative, more efficient ways if we target disruptive service provisioning.

One possible way is to develop new encoder profiles such that the input video is directly encoded into two substreams with each substream corresponding to half-frame and being completely self-contained, i.e., no cross half-frame reference. Considering the volume of existing video content on the Internet, one more feasible way is to develop transcoders that transcode pre-encoded video bitstreams into two self-contained substreams. Such transcoders can be integrated into desktop sync engines like ActiveSync. In either case, the cross-device reference is completely removed and the system complexity is maximally reduced.

There is a concern on the availability of video content with suitable resolution. This can be achieved using transcoding as well. We have developed fast transcoding algorithms that can perform efficient arbitrary resizing transcoding [14]. Currently, popular DVD video resolution is 720×480 and typical screen resolution of mobile phones are 320×240 (QVGA). When the two mobile phones' screen are aggregated together, it will form a virtual screen sized 480×320. In this case, we can apply 3:2 downscaling transcoding to obtain a suitable video. As the TV and movie industry moving to high definition (HD) video, the typical resolution will be 1280×720, and in this case we need to apply 8:3 downscaling transcoding.[4] Note that we have seen new mobile devices with VGA resolution (i.e., 640×480), if two such devices were to be aggregated, then a 4:3 downsizing transcoding process will readily convert a HD video to a suitable one.

6.3 User Study

Motivated by the concept of the better-together mobile application paradigm, a production plan on new mobile phone models (i.e., Lover's Phone) has been formed. An independent market survey was carried out through a third-party vendor.

The market survey result shows that Lover's Phone is warmly welcomed: over 80% people like it and 47% people think it is terrific. The details are shown in Figure 9.

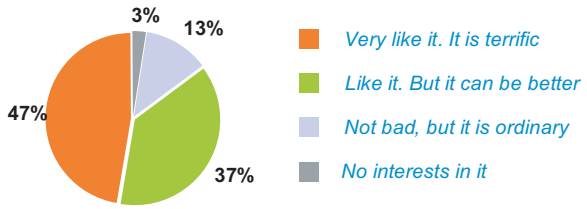


Figure 9: Market survey results on users' opinions on the Lover's Phone.

While the new mobile phone model has several improved features on user interface, information sharing, etc., the screen aggregation is the only brand new application that wins a good score (6 out of 10, with 8 being the highest score). Many users said that the thick frame boundary between the two screens will hurt the viewing experience. They would appreciate the feature much more if the two screens can be patched up seamlessly. Given the advances in the *micro-projector* that can be readily equipped to a portable device, for example PicoP [18] from Microvision, we think the frame boundary barrier will soon be removed.

[4]There will be two thin black bands on top and bottom of the screen in this case since the height of the resulting video will be 288 (rounded to multiples of 16). Another way is to slightly prune away the content on left and right sides of the video to make the transcoded one full screen. The most important rule is to keep the aspect ratio.

6.4 Assumption on Homogeneity

We have assumed homogeneity of the mobile devices in this paper, i.e., the two mobile devices should have the same or similar software and hardware capabilities. This is not a hard constraint. In fact, there is no extra technical constraint except that each device should be capable of playing back video on its own and have networking capabilities.

The major constraint is that the two screens should be the same resolution, which results from the concern on the ultimate user viewing experience. Therefore, it is possible that mobile devices of different models can be put together if they have the same screen size. For instance, we can place together the HP iPAQ rw6828 phone and the Dopod 838 phone, or even a PDA and a mobile phone. In fact, what really matters is the pixel size. The two screens should have same pixel resolution, i.e., same dots-per-inch, since the pixels of LCD displays are square in shape. In this case, only part of the larger screen will be used to match the small one. Given the fact that today's mainstream mobile devices have, as standard configuration, screens with QVGA resolution, we expect the constraint on the screen size will not be a hinderance.

However, the heterogeneity of the mobile devices will impose other challenges. For example, how to automatically achieve load balancing between the two devices while maximizing the user viewing experience, and how to maximize the life time of the together-viewing session given the two mobile devices have different initial energy deposits when they meet. We are currently working on adaptive solutions to these challenges.

7. RELATED WORK

There have been considerable work related to middleware and programming environments for mobile and context-aware applications, typically relying on pub-sub communication structure [19] for information sharing, and proxies for connecting to the Internet while avoiding changing the existing infrastructure [5, 20]. Many other works are devoted to collaborative applications for mobile users [21–23] and some of them are dedicated for video services [24,25]. MobiUS differs from them in its tightly coupled device-to-device collaboration architecture requirement, which is more demanding, to support the new better-together application paradigm.

Recognizing the limited resource of mobile devices, some work studied the resource aggregation of mobile devices, but

mainly focused on bandwidth aggregation [26,27]. CoolSpots [5] debuted a framework where multiple radio interfaces can be adaptively used (with some infrastructure support) so as to save the precious energy for mobile devices. MobiUS emphasizes more than merely bandwidth sharing. In the together-viewing video application, the computing resources from two devices are aggregated as well, besides the obvious aggregation of the display resource. Moreover, MobiUS introduces another dimension to optimize the energy consumption, namely the tradeoff between computation and communication.

The together-viewing video application of MobiUS seeks speedy distributed and parallel video decoding. Parallel video (e.g., MPEG-2) decoding has attracted many research efforts in order to achieve real-time playback. In [28], the authors identified the huge bandwidth requirement when exploiting temporal parallelism. Both temporal and spatial parallelism are exploited in [29] where shared memory is assumed. These works are not directly applicable to MobiUS because of the lack of shared memory among devices and the limited communication bandwidth in between.

The artifact of together-viewing is that the video is displayed on the two screens aggregated together, with each screen displays half a frame. It is therefore different from other remote desktop display scenarios where the desktop is remotely displayed as a whole. As pointed out in [30], supporting video is more challenging and conventional remote desktop technology can not deliver satisfactory performance. A decent approach was developed there to better support remote video display by intercepting the driver level primitives. However, our half-frame display problem is new and the solution requires non-trivial extensions, which we are still investigating.

8. CONCLUSION AND FUTURE WORK

In this paper, we presented MobiUS, a research project that targets a new better-together mobile application paradigm when multiple devices are placed in a close proximity. We studied a together-viewing video application which requires real-time synchronous video playback on the screens of the two devices. We identified the key challenges, namely the intrinsic complexity of video caused by the recursive temporal frame dependency and motion compensated prediction and the inherent constraints of mobile devices such as limited processing power and short battery life. We overcame these challenges through device collaboration and resource aggregation based on the design of a tightly coupled collaborative system architecture. We designed a novel collaborative half-frame decoding scheme that significantly reduces the computation complexity of decoding and further optimized it for better energy efficiency with a guardband-based collaborative half-frame decoding scheme. We have implemented the system and conducted experimental evaluations whose results demonstrate the effectiveness of our proposed collaborative and resource aggregation techniques.

We believe the better-together mobile application paradigm deserves further exploration. In our future work, our immediate plan is to conduct more systematic evaluation of our current system design and implementation. In the long run, we plan to abstract and generalize those common modules that have been identified in MobiUS to support a broader spectrum of collaborative applications such as collaborative streaming for wireless TV scenario where the downloading bandwidth of devices can be aggregated and more effectively utilized. We believe our work will motivate a new wave of technology developments addressing and promoting the social impact of mobile phones.

9. ACKNOWLEDGEMENT

We would like to thank Mr. Wenfeng Li for initial porting of the video decoder to Windows Mobile platform, and Mr. Feng Hu for conducting some experiments. We also thank many colleagues in the Wireless and Networking research group in MSR Asia, Kun Tan, Chunyi Peng, Yunxin Liu, to name a few, for their strong support, encouragement, many fruitful discussions and valuable suggestions throughout the project. We also thank all the reviewers for their insightful comments and valuable suggestions, and Dr. James Scott for shepherding the final revision of the paper.

Our particular thanks go to Mr. Weihun Liew who helped evangelizing our work, conducted third-party market survey, and initiated the product plan.

10. REFERENCES

[1] [Online]. Available: http://www.networkmultimedia.org/

[2] *Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video.* ITU-T and ISO/IEC JTC 1, ITU-T Recommendation H.262 and ISO/IEC 13818-2 (MPEG-2), 1994.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003.

[4] B. Zeng, R. Li, and M. L. Liou, "Optimization of fast block motion estimation algorithms," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, pp. 833–844, Dec. 1997.

[5] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces," in *Proc. of the Fourth Intl. Conf. on MobiSys*, Uppsala, Sweden, 2006, pp. 220–232.

[6] [Online]. Available: http://www.upnp.org/

[7] [Online]. Available: http://www.apple.com/macosx/features/bonjour/

[8] F. Wu, G. Shen, K. Tan, F. Yang, and S. Li, "Next generation mobile multimedia communications: Media codec and media transport perspectives," *China Communications*, vol. 3, pp. 30–44, Oct. 2006.

[9] J. Hightower, R. Want, and G. Borriello, "SpotON: An indoor 3D location sensing technology based on RF signal strength," University of Washington, UW CSE 00-02-02, 2000.

[10] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. of the Sixth Annual ACM MobiCom*, Boston, MA, USA, Aug. 2000.

[11] G. Borriello, A. Liu, T. Offer, C. Palistrant, and R. Sharp, "Walrus: Wireless acoustic location with room-level resolution using ultrasound," in *Proc. of the Third Intl. Conf. on MobiSys*, Seattle, WA, USA, 2005, pp. 191–203.

[12] [Online]. Available: http://www.eecis.udel.edu/ mills/ntp.html

[13] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. of the Fifth OSDI*, Boston, MA, USA, 2002, pp. 147–163.

[14] G. Shen, Y. He, W. Cao, and S. Li, "Mpeg-2 to wmv transcoder with adaptive error compensation and dynamic switches," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, pp. 1460–1476, 2006.

[15] S. Pasricha, M. Luthra, S. Mohapatra, N. Dutt, and N. Venkatasubramanian, "Dynamic backlight adaptation for low-power handheld devices," *IEEE Design and Test of Computers*, vol. 21, no. 5, pp. 398–405, 2004.

[16] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proc. of the Eighteenth ACM SOSP*, Banff, Alberta, Canada, 2001, pp. 89–102.

[17] W. Yuan and K. Nahrstedt, "Practical voltage scaling for mobile multimedia devices," in *Proc. of the Twelfth Annual ACM Intl. Conf. on Multimedia*, New York, NY, USA, 2004, pp. 924–931.

[18] [Online]. Available: `http://www.pcmag.com/article2/0,1895,2080442,00.asp`

[19] M. Caporuscio and P. Inverardi, "Yet another framework for supporting mobile and collaborative work," in *Proc. of the Twelfth Intl. Workshop on Enabling Technologies*, Washington, DC, USA, 2003.

[20] M. Caporuscio, A. Carzaniga, and A. L. Wolf, "Design and evaluation of a support service for mobile, wireless publish/subscribe applications." *IEEE Trans. Software Eng.*, vol. 29, no. 12, pp. 1059–1071, 2003.

[21] D. Buszko, W.-H. D. Lee, and A. S. Helal, "Decentralized ad-hoc groupware api and framework for mobile collaboration," in *Proc. of the 2001 Intl ACM SIGGROUP Conf. on Supporting Group Work*, Boulder, Colorado, USA, 2001, pp. 5–14.

[22] E. Kirda, P. Fenkam, G. Reif, and H. Gall, "A service architecture for mobile teamwork," in *Proc. of the 14th Intl Conf. on Software Eng. and Knowledge Eng.*, 2002, pp. 513–518.

[23] V. Sacramento, M. Endler, H. K. Rubinsztejn, L. S. Lima, K. Goncalves, F. N. Nascimento, and G. A. Bueno, "Moca: A middleware for developing collaborative applications for mobile users," *IEEE Distributed Systems Online*, vol. 5, no. 10, 2004.

[24] E. de Lara, R. Kumar, D. S. Wallach, and W. Zwaenepoel, "Collaboration and multimedia authoring on mobile devices," in *Proc. of the First Intl. Conf. on MobiSys*, San Francisco, CA, USA, 2003, pp. 287–301.

[25] N. J. McCurdy and W. G. Griswold, "A systems architecture for ubiquitous video," in *Proc. of the Third Intl. Conf. on MobiSys*, Seattle, WA, USA, 2005, pp. 1–14.

[26] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin, "Handheld routers: Intelligent bandwidth aggregation for mobile collaborative communities," in *Proc. of the First Intl Conf on Broadband Networks*, Washington, DC, USA, 2004, pp. 537–547.

[27] J. Karlsson, H. Li, and J. Eriksson, "P2p video multicast for wireless mobile clients," in *The Fourth Intl. Conf. on MobiSys (poster)*, Uppsala, Sweden, June 2006.

[28] J. Wang and J. C. L. Liu, "Parallel mpeg-2 decoding with high speed network: Part ii," in *Proc. of the IEEE Intl. Conf. on Multimedia and Expo (ICME), 2001.*, Aug. 2001, pp. 333–336.

[29] A. Bilas, J. Fritts, and J. P. Singh, "Real-time parallel mpeg-2 decoding in software," in *Proc. of the 11th Intl. Symposium on Parallel Processing*, Washington, DC, USA, 1997, pp. 197–203.

[30] R. A. Baratto, L. N. Kim, and J. Nieh, "Thinc: A virtual display architecture for thin-client computing," in *Proc. of the Twentieth ACM SOSP*, Brighton, United Kingdom, 2005, pp. 277–290.