A Novel Approach for Sharing White Board between PC and PDAs with Multi-Users

Xin Xiao[1], Yuanchun Shi[2],Weisheng He[1],

[1,2] Department of Computer Science and Technology, Tsinghua University
100084, Beijing, China
[1]{xiaoxin00, hws99}@mails.tsinghua.edu.cn, [2]shiyc@tsinghua.edu.cn
http://media.cs.tsinghua.edu.cn/~pervasive

Abstract. White board sharing between PC and PDAs is a typical interactive application between PC and mobile device in ubiquitous environment. Due to the limited size of PDA' s screen, when a large amount of clients simultaneously use PDAs to share the server's screen while their viewed areas (or *viewports*) are different in size and position, the server must transform the viewports to fit the size of PDA for each client and thus will suffer from heavy burden. A novel approach is proposed in this paper to reduce the burden of the server and provide the clients with real time white board sharing. Instead of zooming the viewport to fit PDA's screen for every user, it only zooms the screen nine times; then, chunked encoding is adopted to reduce the cost for encoding the overlapped area of different viewports. The experiment results have demonstrated the efficiency and effectiveness of our approach.

Keywords: white board sharing, mobile device, multi-users

1 Introduction

In recent years, mobile devices such as PDA, Smart Phone have been experiencing a significant growth, which considerably affect people's life. Especially, they are widely used in interactive applications with PC, such as SlideShow Commander [5], whiteboard-based interactions [3], VNC-based access to remote computers [12] and so on. Currently, in our SEMIC [6] project, we are exploring the following area in mobile environment: screen or white board is shared between PC and multi-PDAs so that the underlying scenarios can be achieved: a large amount of remote students could browse the PowerPoint screen of the Smart Classroom [7, 8] by PDA in any places; we could hold Smart Phones to continue our discussion by means of white board sharing on our way home and so on.

In order to achieve these goals, many problems need to be overcome. Among them, the crucial challenge is how to enable multi-users with mobile devices to simultaneously share the same remote screen while their viewed areas (or viewports) are different in size and position. Unlike the traditional desktop PC, mobile devices have much smaller display area, e.g. PDA only has the resolution of 240×320 pixels, therefore, the viewed range of the remote screen and the image clarity are conflict with each other on mobile devices. On one hand, the whole remote screen sometimes needs to be zoomed to fit the size of the mobile device so that the user could have an

overview of the entire remote screen. However, in this way the image on the mobile device is much smaller than its original size, preventing the user from clearly watching the content. On the other hand, sometimes an area with the same size as the mobile device screen may need to be captured from the remote screen and delivered to the users so that they can see a clear image. However, in this situation the users can just see a part of the remote screen. Hence, the above two schemes are not sufficient to provide users with satisfactory vision experience. Instead, if a series of viewports with gradually changed size captured from the remote screen can be provided, the user can choose a preferred one. E.g. if an area with resolution of 480×640 pixels captured from the remote screen is zoomed to 240×320, it could provide clearer image than the scheme zooming the whole screen to 240×320; and compared with the scheme only capturing 240×320 from the remote screen, it can make the user see larger area of the remote screen.

If there are many users, however, the following problem arises: each user has its own preference for the size and position of the viewport from the remote screen. This brings us great challenge to meet all users' needs. On one hand, if the whole screen is sent to the users and zoomed on the mobile devices, it will result in both the redundant network load and extra time expenditure on mobile devices. On the other hand, if the server captures each user's the required area one by one and sends them to the remote users, it will suffer from heavy burdens when there are many users.

A novel approach is proposed in this paper to cope with the above problem, which aims to reduce the burdens on both the server and the client sides and to provide the remote users with real time screen/white board sharing. The theoretic analysis and experiment result show that our approach is efficient when multi-users share the remote white board with PDAs.

The rest of the paper is organized as follows: in section 2 we introduce the related work; then details of our approach are presented in section 3 and 4. We present the experiment result and evaluation in section 5. Section 6 concludes the paper and outlines our future work.

2 Related Works

In literature [1], a Rapid Serial Visual Presentation (RSVP) based approach is proposed to enable automatic browsing of large pictures on mobile devices. Firstly, an image attention model, which manipulates each region-of-interest in the image separately, is employed to illustrate the information structure within an image. Then, an optimal image browsing path is calculated based on the image attention model to simulate the human browsing behaviors. However, the concept "region-of-interest" may not always hold. For instance, if it is a PowerPoint slide, the whole image usually contains evenly distributed information, thus it's difficult to tell which regions are of interest. Besides, it just focuses on how to browse large pictures on local mobile device.

There have also been several research results on shared white board applications [2][9][10][11], among which, PebblesDraw [2] allows all of the users to draw simultaneously on different PDAs while sharing the same PC display. This project

explores the field that multi-users with PDAs interactive with PC through white board sharing. However, it still does not take into account how to deliver viewports with different sizes and positions captured from the same white board to remote users. Therefore, the users can't choose the browsing resolution and position as they like.

SVNC [12], a VNC[4]-based application, enables users to access and control remote computers from cellular phones. SVNC makes some efforts to solve problems inherent to the cellular phone's small screen. For instance, frequently used screen area can be assigned and restored quickly by using the *Shortcut* function. Also, the viewport can be widened (zoom out) to browse its contents and narrowed (zoom in) to see the display in greater detail. In our work, we expand zoom out and zoom in to gradually change in viewport's size which will be described in detail in the next section.

As is discussed above, there are already some works concerning about large image's browsing on PDA, white board sharing between PC and PDAs. They don't consider, however, the problem of multi-users' share of the same screen while each user's viewport is different; and accordingly say nothing of how to improve the efficiency of the server to achieve this goal. Hence, we propose a novel approach to solve the problem.

3 Classification of Users and Screen Snatch Based-on Viewports' Sizes

Our approach can be divided into two parts, whose first part—classification of users and screen snatch based on viewports' sizes will be described in this section and whose second part—chunked encode of viewports will be described in section 4. Here we must point out that we assume the resolution of the PC is 1024×768 pixels and 240×320 for PDA in the following sections.

Recall that to meet the needs of multi-users with different viewports, a possible solution is to send the encoded whole screen to all of them and then decode the received data, capture the required viewport from the screen and zoom the viewport to fit the size of the mobile device. However, this approach has obvious shortcomings: since each user only needs to browse part of the whole screen, it wastes network bandwidth and CPU time to process the unnecessary data. Moreover, the PC server is much more powerful than the PDA client in CPU speed, memory capacity and bandwidth, which makes it unrealistic for the client to keep up with the server. Instead, in our approach, the server processes the captured screen and only sends the necessary parts to the clients to reduce the extra cost and to make the clients synchronous with the server.

Step 1: our approach firstly defines nine gradually changed sizes (width×height) of viewports on the white board so that the size of each user's viewport belongs to one of them:

{240×270, 288×324, 346×389,
415×467, 498×560, 598×672,
718×768, 862×768, 1024×768}

Two neighbored sizes have the relation that:

$$\begin{bmatrix} Width_{n+1} \\ Height_{n+1} \end{bmatrix} = f * \begin{bmatrix} Width_n \\ Height_n \end{bmatrix} \quad (1)$$

Here, f is so-called *Gradient Factor (GF)*. In our approach we set it to 1.2 to gain the effect of gradual change in size among different groups of viewports and we found in our experiment that users could have a very natural experience when browsing the white board with f =1.2. Besides, when we use the program "Windows Picture and Fax Viewer" in Windows XP to zoom out or zoom in a picture, we find the variation is also 1.2 times. Here, we must point out that the first group has size 240×270 but not 240×320 because 320 is the height of PDA's whole screen; then, the available height of the application is 270 with the heights of title and menus deducted.

Users can choose what sizes of area to be viewed. If a user chooses group n, he means to view an area with size $Width_n \times Height_n$ on remote screen. Finally, after our approach completes, his required viewport is zoomed to 240×270 to be sent to his PDA. Figure 1 illustrates the effect of the nine gradually changed sizes zoomed to fit the PDA's screen.


1(a) 240×270


1(b) 288×324


1(c) 346×389


1(d) 415×467


1(e) 498×560


1(f) 598×672


1(g) 718×768


1(h) 862×768


1(i) 1024×768

Fig. 1. Nine viewports with gradually changed sizes zoomed to fit PDA's screen. The original sizes of these viewports are marked in the figure.

Note that although we just present the algorithm for PDAs, it's quite similar for other mobile devices with different screen sizes. For example, for a mobile devices with screen size 160×120 pixels, the gradually changed viewports are with size

{160×120, 192×144, 230×173,
276×208, 331×250, 397×300,
476×360, 571×432, 685×518,
822×622, 986×746, 1024×768}

Step 2: zoom the screen nine times and fetch a specific area with size 240×270 from one of the zoomed screens for each user:

In the first step, we have defined nine groups of gradually changed sizes of viewports on the remote screen, and the size of each user's viewport must belong to one of the groups.

If we capture, zoom and encode the required area for each user, the cost (on average 32 ms per time for a user in our experimental environment, in which the time spent zooming plays a crucial role) will be high if the count of users is large. Alternatively, our approach only zooms the whole screen nine times. More specifically,

- Firstly, for users with the same viewport size, it zooms the whole screen to a specific size so that their viewports are zoomed to 240×270. For instance, for the users whose viewed areas are 346×389 on the remote screen, since their viewed areas will be zoomed to 240×270, the whole screen should be zoomed to:

$$\frac{1024}{346/240}\times\frac{768}{389/270}=711\times533\cdot$$

Figure 2 illustrates how to zoom the whole screen to specific sizes. Because there are totally nine groups of gradually changed viewport sizes, the server only zooms the screen for nine times rather than n times (n is the count of users).

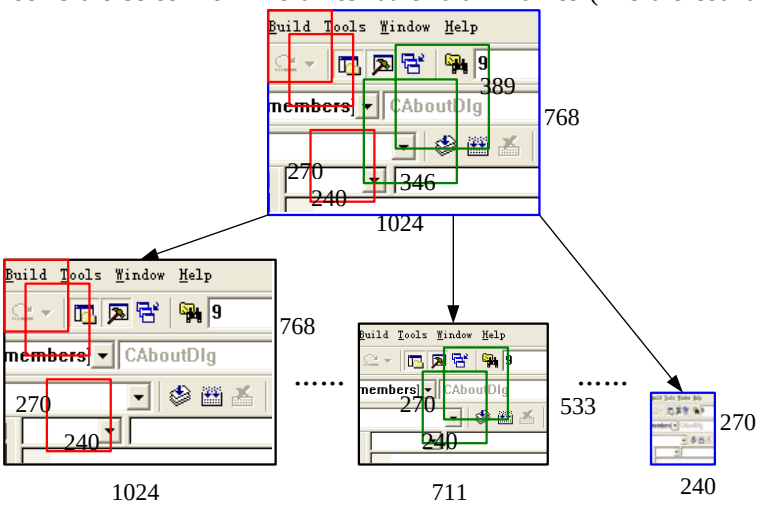


Fig. 2. Zoom the whole screen to nine different sizes. Colored rectangles are the required viewports by the users. The rectangle on the top is the original screen; the rectangles at the bottom are zoomed from the original screen. Note, the original required viewports are all zoomed to 240×270.

- Then, it just needs to copy the zoomed viewports from the zoomed screen and to encode them(on average 6 ms per time for a user to copy and encode in our experimental environment).

Note that for the approach zooming the viewport for every user, if there are 100 users, it needs to zoom 100 times; and for our approach, it only needs to zoom nine times. Also note, zooming a viewed area to 240×270 takes much more time than copying an area with 240×270 from the zoomed screen. Suppose there are n users, the former approach will cost time T_1:

$$T_1 = t_1 * n \tag{2}$$

where t_1 is the time to capture, zoom and encode the required area for each user. And our approach will cost time T_2:

$$T_2 = \tau + t_2 * n \tag{3}$$

where τ is the time spent on zooming the whole screen for nine times and t_2 is the time to copy and encode an area of 240×270 for each user. We find in our experiment that $t_1 \approx 32ms$, $\tau \approx 150ms$ and $t_2 \approx 6ms$. With the increase of n, our approach will be significantly faster than the former one because it reduces the zooming times.

In the following section, the last step (step 3), which can further improve the efficiency of our approach, will be explained.

4 Chunked Encode of Viewports

In last section, we classify the users according to their viewports' sizes and zoom the whole screen nine times followed by copying only 240×270 from the zoomed screen for each user, which can effectively reduce the additional cost compared with the approach zooming the viewports for all the users. However, in step 2, it doesn't consider the following problem:

The viewports of two users in the same group may overlap with each other, which is shown in Figure 3. Since all zoomed viewports need to be encoded to compress the data before transmitting to remote PDAs, if the two overlapped viewports are encoded separately, there is extra cost for the common part of them. Even worse, the extra cost might be significant when there are many users in the same group, because it causes the probability of the viewports overlapping to increase but no measure is taken to deal with it.

Step 3: Therefore, chunked encoding is applied to encode the viewports to reduce the cost for their overlapping parts:

The nine zoomed screens are all divided into several chunks where each chunk has the size of 80×90 pixels so that every viewports on the zoomed screen contains 3×3

chunks, and when the user moves his/her viewport in the horizontal or vertical direction once, the distance should be equal to the width or height of a chunk, which is illustrated in Figure 3. Although this way makes the movement of the viewports non-continuous, the user will not feel any big difference when watching due to the small size of a chunk.

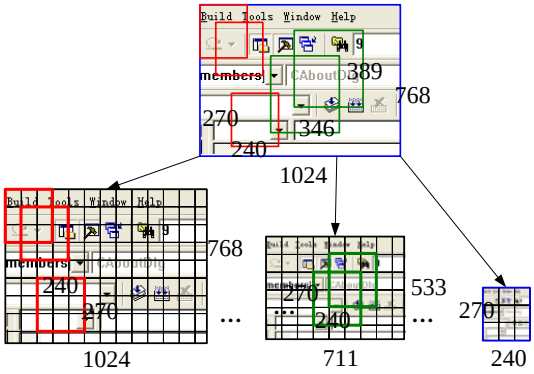


Fig. 3. Chunked encoding

When encoding a user's viewport, our approach is implemented as follows:

```
for each of the 3*3 chunks of the viewport do
begin
    if the chunk has not been encoded then
        encode it;
    else begin    /* it has been encoded when
                     encoding other users' viewports */
        doesn't encode it again and just use the encoded data;
    end-if
end-for
```

In this way, much time spent in encoding the overlapping parts is saved.

Finally, after processing all the nine chunks, the server can send the chunked encoded data to the user. And when the user's PDA receive these data, it decodes for every chunk and displays viewports on the screen.

5 Implementation and Evaluation

We have implemented our approach for white board sharing between PC and PDAs in our SEMIC project. To demonstrate the efficiency and effectiveness of this approach, two experiments are carried out. The server runs on a PC with CPU Pentium 4, frequency 1.4GHZ; memory 512MB; operating system Windows XP; and the PDAs are HP 5550 and Lenovo TJ 218. The two experiments are carried out as follows:

5.1 Experiment One

Initially, the server records 100 users' information in its *user-info-list*. Their viewports' sizes are evenly distributed on the nine groups defined in step 1. Since we don't have 100 PDAs at hand currently, 98 of them are simulators and 2 of them are real devices.

After the server begins to capture the screen, the sizes and positions of these users' viewports change once by probability P (we call P *Variation Probability*, which varies from 10% to 100%) every 1000ms. The server continues capturing screen and implementing our approach. The time for capturing and generating the frames is measured.

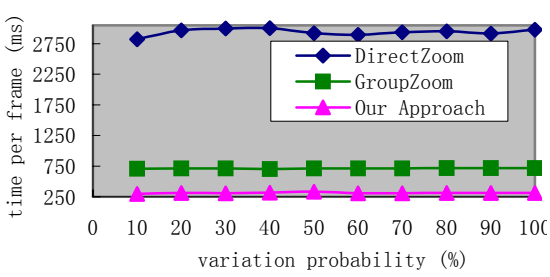
In contrast, other two approaches are also implemented. One is:

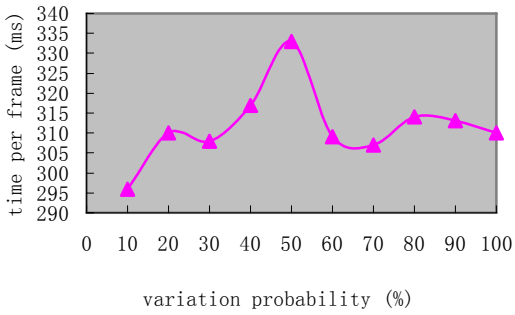
- Zooming the required area to 240×270 and encoding it for every user. We call this approach "DirectZoom" in the figure of the experiment result.

and the other is:

- Implementing our approach by step 1 and step 2 (Zooming the captured screen nine times, then copying the required parts from the zoomed screen for every user), but not adopting chunked encoding. We call this approach "GroupZoom" in the figure of the experiment result.

The experiment result is illustrated in Figure 4(a).





4(a) 4(b)

Fig. 4(a). Average time-per-frame with 100 users; **4(b).** Average time-per-frame with 100 users for our approach

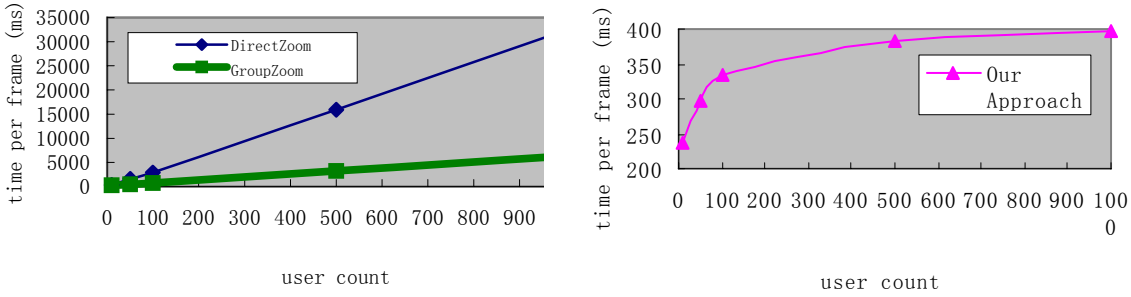
From Figure 4 (a), we can see that our approach is much faster than Approach "DirectZoom" and Approach "GroupZoom" when user count equals 100. Compared with Approach "DirectZoom", it saves time 89.4%; and compared with Approach "GroupZoom", it saves time 56.4% per frame.

An interesting phenomenon occurs when running our approach: the maximum value of time-per-frame occurs at *variation probability* P equals 50% rather than 100% or other values (see Firugre 4(b)). Intuitively, we may think the maximum value of time-per-frame should occur when every user's viewports change (at this time P equals 100%). The explanation to this phenomenon is as follows: although previous overlapped areas don't exist any more when every user changes his viewed

area (P=100%), it may lead to many new overlapped areas of the new viewports. In contrast, if P equals 50%, there are only approximately half of the users changing their viewports, which leads to less overlapped areas of the new viewports and the amount of previous overlapped areas also decreases. Since chunked encoding is adopted in our approach, it takes less time when P equals 100%.

5.2 Experiment Two

Experiment two is somewhat similar to experiment one, but the *variation probability* P is fixed to 50% and the user count varies from 10 to 1000. Figure 5 showes us the results.



5(a) 5(b)

Fig. 5(a). Average time-per-frame for Approach "DirectZoom" and "GroupZoom" when P=50%; **5(b).** Average time-per-frame for our approach when P=50%

From Figure 5(a) we can see that the time-per-frame is indeed linear to user count in Approach "DirectZoom" and Approach "GroupZoom", which is in accord with equation (2) and (3). When user count is large, time spent generating a frame for everyone is accordingly significant.

Figure 5(b) illustrates the relation between time-per-frame and the user count of our approach. With the increase of user count, time-per-frame increases slowly. Even if there are 1000 users, only 397ms needs to be spent generating the frame for all of them. At this time, the frame rate is about 2.5 frame/s. When user count is smaller, say, less than 100, the frame rate is higher than 3 frame/s. In general, the content displayed on the white board (such as PowerPoint, electronic map, etc.) doesn't vary as fast at the video, therefore, this rate is sufficient to provide the users with high quality.

In a word, the results of experiments 1 and 2 demonstrate the efficiency and effectiveness of our approach for white board sharing between PC and PDAs with multi-users. Compared with two other approaches, it avoids the linear increase of time-per-frame with the increase of user count and meets the need for frame rate in our application scenario described in the Introduction part. This approach has been

used in out SEMIC [6] project. In this project, based on this approach users can also annotate on the shared white board on their PDAs.

6 Conclusion

In this paper, we present a novel approach for white board sharing application between PC and PDAs with multi-users in ubiquitous environment. It enables a large amount of users to simultaneously browse viewports with different sizes and positions on the white board. The experiment result shows its efficiency and effectiveness.

Our future work will continue in two aspects. Firstly, we will utilize a user model to simulate the user behavior when they browse the remote screen so that a new experiment based on the user model can be carried out to test our approach. Secondly, in the current implementation, we use our own encode module. In future, we'll replace it with a mature encode tool to improve the efficiency of our approach.

Acknowledgments. This research is supported by NCET-04-0079 (Program for New Century Excellent Talents in University, China).

References

1. Hao Liu, Xing Xie, etc., "Automatic Browsing of Large Pictures on Mobile Devices", MM'03, November 2-8, 2003, Berkeley, California, USA
2. Brad A. Myers, Herb Stiel, Robert Gargiulo, "Collaboration Using Multiple PDAs Connected to a PC", submitted for publication
3. Jun Rekimoto, "A Multiple Device Approach for Supporting Whiteboard-based Interactions", SIGCHI conference on Human factors in computing systems, 1998, pp. 344~351
4. Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, etc., "Virtual Network Computing", IEEE Internet Computing, January and February, 1998, pp. 33~38
5. BRAD A. MYERS. "Using Handhelds and PCs Together", COMMUNICATIONS OF THE ACM. November 2001. Vol.44, No.11, pp34-41.
6. http://media.cs.tsinghua.edu.cn/~pervasive/projects/semic
7. Y.C. Shi, W.K. Xie, G.Y. Xu, etc. "The Smart Classroom: Merging Technologies for Seamless Tele-Education." Pervasive Computing. April-June 2003. pp47-55.
8. W.K. Xie, Y.C. Shi, G.Y. Xu and Y.H. Mao. "Smart Platform - A Software Infrastructure for Smart Space (SISS)" , ICMI'02, October 14 - 16, 2002. Pittsburgh, Pennsylvania. p. 429.
9. Bier, E.A. and Freeman, S. "MMM: A User Interface Architecture for Shared Editors on a Single Screen," in Proceedings UIST'91. Hilton Head, SC: pp. 79-86.
10. Pederson, E., et al. "Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings," in Proceedings INTERCHI'93: Human Factors in Computing Systems. 1993. Amsterdam, The Netherlands: pp. 391-398.
11. Stewart, J.E. "Single Display Groupware," in SIGCHI'97 Adjunct Proceedings: Human Factors in Computer Systems, Extended Abstracts. 1997. Atlanta, GA: pp. 71-72.
12. Buntarou Shizuki, Masato Nakasu, and Jiro Tanaka, "VNC-BASED ACCESS TO REMOTE COMPUTERS FROM CELLULAR PHONES", Proceedings of the IASTED International Conference on Communication Systems and Networks (CSN 2002), pp. 74-79