A Cross-Device Drag and Drop using Compression Technique

Nilima R. Deore[1] Pratiksha P. Kiragi[2] Aditee S. Boraste[3] Harshada A. Kadam[4]
[1,2,3,4]NDMVP'S KBT COE

Abstract— Many interactions takes place across various devices with diff erent configurations. The data received by the mobile can be processed more efficiently on large screen devices, or vice- a-versa. Information is often created on one device but needed on another device , hence there is a need for communication between these devices. Early work related to this field includes Pick-and-Drop and Hyper Drag that demonstrated point-to-point interaction across devices. This systems use point to point and gesture tracking, where data transfer is based on gestures which includes distance limitations. Hence, this system introduce cross-device drag and drop technique where it can drag the files and drop to the specific application on target device. This system uses Huff man and Limpel-Ziv lossless data compression algorithm for compression. By using this it can transfer the files at high data rates over the long distance within a particular Wi-Fi range. In this study it introduce multicasting and broadcasting across cross devices i.e., from one source device to multiple target devices.

Key words: Cross Device, Huff man compression, Limpel-ziv compression, lossless, Wi-Fi

I. INTRODUCTION

Interactions frequently extend beyond a single device. A phone number is more easily searched on a larger screen, but once found the call is issued with the mobile. A photo can be quickly snapped with a mobile, but its integration in a document is easier on a larger screen. Friends text us place names on our mobiles, but route direction scan be better looked up and printed from a larger device. We might also look up an address on a desktop screen, and then use it on our mobiles to navigate to the location. All these are examples of interactions that require users to select data on one device and apply it to another. In this situation, current practices hinder this process by adding extra steps that divert users from the primary Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific goal of applying data from one device to another.

II. RELATED WORK

A. Existing System:

The existing system is based on gesture and point to point interaction techniques.

1) Process for Sending Data:

There are several transmission techniques developed for data transfer. For example, J. Rekimoto presented "Pick and Drop" a direct manipulation techniques for multiple computer environments. In the authors designed which allows users to pick up digital objects with a stylus and drop them onto another screen. The technique is different from tradition drag-and-drop because selecting an object virtually attaches it to the stylus, which can them be moved without physically contacting the screen. This also facilitates transferring objects across devices.

J. Rekimoto presented "Hyperdrag" Hyperdragging is a technique designed to facilitate information management in scenarios where a laptop computer is positioned on an interactive tabletop. In this case the table acts as an extension of the laptop's workspace. When a user wants to share information with a colleague, she selects an item on her laptop and uses the mouse cursor to drag it towards the edge of the screen and further to the interactive table surface.

"Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch and pen-operated systems" by P. Baudisch, E. Cutrell, D. Robbins. In Drag-and-Pop extends the traditional Drag-and-Drop technique by bringing the icons of applications that can accept a selected file, such as a text document, closer to its icon. This is useful for scenarios where a user is interacting with multiple displays and the bezel is interfering with the dragging gesture. Drag- and-Pick on the other hand temporarily moves all icons in the direction of the mouse cursor, making it easier to interact with them for example by using a stylus.

III. SYSTEM ARCHITECTURE

The fig 1. describes about system architecture of a cross device Drag and Drop using compression technique. User has to select the file first, then the file get compressed, after that the user has to select the recipients. In the system we use Wi-Fi as a communication medium. So through the Wi-Fi network the files get transferred to the recipients system. At the recipients system the original file is recovered by decompressing it.

At the receiver end there is a pop up menu for validate the file. The validation is required for storing the incoming files at proper places like image file to image folder etc.

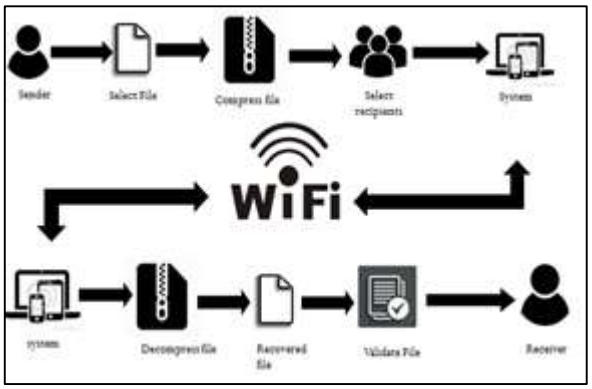


Fig. 1: Architecture Diagram

A. The Pc Drag Detector:

The PC Drag Detector is a background application that enables Drag-and-Drop on the desktop. It captures drag events at the system level (i.e., independent of any specific desktop application). When a drag gesture is detected, it displays a detector window at the sides of the screen. Once the user enters or exits the screen, the relevant window

captures outgoing or incoming data (Figure a). Depending on the drag direction, data is sent over the network to the phone or received from it to complete the drag gesture on the desktop. After which the detector window is hidden.

B. The Bridge Application:

The bridge application connects Drag-and-Drop to existent applications on the same- phone. It handles bidirectional interactions between desktop and mobile, using two diff erent UIs described in the following. After starting the bridge application, it presents a grid of icons to the user, each representing a distinct application class.Together, these six classes address arrange of typical usage scenarios. In particular, we are covering 1) phone dialers, 2) contact managers, 3) text messaging, 4) email clients, 5) maps and 6) picture viewers. For example, after finding a restaurant while browsing the web on the desktop, the user can simply select the corresponding address and drag it onto the detector window after which the bridge application will map the address onto the maps icon on the phone to immediately start the navigation. This basic concept readily extends to other use cases. For instance, to call a number displayed in a desktop application, the user applies it to the dialers icon to immediately get connected. In doing so, Drag-and-Drop off ers a quick and convenient alternative to manually transferring the required information from the desktop to the mobile phone, for example by retyping it. Through a mechanism called implicit intents, the Android OS determines eligible applications that can handle the data from those available in the users device.

C. Sending Data to a Desktop:

The Bridge Application also allows dragging items from the phone to the desktop to address situations where data originating from a mobile device is better view edited on a larger screen. In order to support this, we use Androids built-in share feature, a method to internally share data between various applications. In particular, users have to first select the data they wish to apply on another de- vice from within an arbitrary application on the phone (e.g., by opening a picture in the gallery viewer). Second, they invoke the share feature which will bring up the bridge application in send mode. Due to platform restrictions users cannot initiate a drag gesture directly from a mobile application. Thus, users are shown a list of icons representing data types. Compatible ones will be highlighted, in situations where data might be treated in diff erent ways (e.g.: a picture sent as a URL or as a binary file).

D. Need of Image Compression:

Images transmitted over the internet are an excellent example of why data compression is important. Suppose we need to download a digitized color photograph over a computer's 33.6 kbps modem. If the image is not compressed (a TIFF file, for example), it will contain about 600 kilo bytes of data. If it has been compressed using a lossless technique (such as used in the GIF format), it will be about one-half this size, or 300 Kbytes. If lossy compression has been used (a JPEG file), it will be about 50 Kbytes. The download time for these three equivalent files are 142 seconds, 71 seconds, and 12 seconds, respectively which is a huge difference .JPEG is the best choice for digitized photographs, while GIF is used with drawn images, such as company logos that have large areas of a single color. The idea behind image compression is to use the fewest possible bits per pixel to code the image while still having a compressed image comparable to the original image.

E. Huffman Coding For Image Compression:

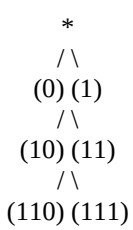
Huffman code is a technique for compressing data. Huffman's greedy algorithm looks at the occurrence of each character and it as a binary string in an optimal way. Huffman coding is a form of statistical coding which attempts to reduce the amount of bits required to represent a string of symbols. The algorithm accomplishes its goals by allowing symbols to vary in length. Shorter codes are assigned to the most frequently used symbols, and longer codes to the symbols which appear less frequently in the string (that's where the statistical part comes in)

F. Huffman Algorithm:

Code word lengths are no longer fixed like ASCII .Code word lengths vary and will be shorter for the more frequently used characters. We have created a code in Delphi for Huffman method. Following steps describes working and algorithm for Huffman coding.

1) Read a BMP image using image box control in Delphi language. The Image control can be used to display a graphical image - Icon (ICO), Bitmap (BMP), Metafile (WMF), GIF, JPEG, etc. This control will read an image and convert them in a text file.
2) Call a function that will Sort or prioritize characters based on frequency count of each characters in file.
3) Call a function that will create an initial heap. Then reheap that tree according to occurrence of each node in the tree, lower the occurrence earlier it is attached in heap. Create a new node where the left child is the lowest in the sorted list and the right is the second lowest in the sorted list.
4) Build Huffman code tree based on prioritized list. Chop-off those two elements in the sorted list as they are now part of one node and add the probabilities. The result is the probability for the new node.
5) Perform insertion sort on the list with the new node
6) Repeat STEPS 3,4,5 UNTIL you only have 1 node left.
7) Perform a traversal of tree to generate code table. This will determine code for each element of tree in the following way.

The code for each symbol may be obtained by tracing a path to the symbol from the root of the tree. A 1 is assigned for a branch in one direction and a 0 is assigned for a branch in the other direction. For example a symbol which is reached by branching right twice, then left once may be represented by the pattern '110'. The figure below depicts codes for nodes of a sample tree.



8) Once a Huffman tree is built, Canonical Huffman codes,

which require less information to rebuild, may be generated by the following steps:

Step 1. Remember the lengths of the codes resulting from a Huffman tree generated per above.

Step 2. Sort the symbols to be encoded by the lengths of their codes (use symbol value to break ties)

Step 3. Initialize the current code to all zeros and assign code values to symbols from longest to shortest code as follows:

- If the current code length is greater than the length of
- the code for the current symbol, right shift off the extra bits.
- Assign the code to the current symbol.
- Increment the code value.
- Get the symbol with the next longest code.
- Repeat from A until all symbols are assigned codes

9) Encoding Data- Once a Huffman code has been generated, data may be encoded simply by replacing each symbol with its code.
10) The original image is reconstructed i.e. decompression is done by using Huffman Decoding.
11) Generate a tree equivalent to the encoding tree. If you know the Huffman code for some encoded data, decoding may be accomplished by reading the encoded data one bit at a time. Once the bits read match a code for symbol, write out the symbol and start collecting bits again.
12) Read input character wise and left to the tree until last element is reached in the tree.
13) Output the character encodes in the leaf and returns to the root, and continues the step 12 until all the codes of Corresponding symbols is known.

IV. Conclusion

Usually data transfer is done by the devices which are near to each other. If the user is far away we cannot transfer the data. Our proposed system tackle this problem of data transfer. Hence, through our developed system we can easily transfer the data, if the device is in Wi-Fi range. The data get transferred to the devices. The system will immediately transfer the data within a particular Wi-Fi range. The files get compressed and easily get transferred. Hence it is expected that the data should transfer fastly, even in large distances with no limitations.

Acknowledgment

This research work was support by Prof. V. S. Tidake, NDMVP'S KBT COE Nashik. We thank her for guiding us and providing insight which greatly assisted our research work. We also thank Prof. B. S. Tarle, H.O.D. NDMVP'S KBT COE Nashik for his constant motivation. We would also like to show our gratitude to Dr. Prof. Jayant T. Pattiwar, Principal NDMVP'S KBT COE Nashik and Management of NDMVP Samaj for providing all necessary facilities and their constant encouragement and support.

References

[1] Adalberto L. Simeone Lancaster University Lancaster,UKa.simeone@lancaster.ac.ukJulian Seifert Ulm University Ulm, Schmidt Hasso-Plattner Institute Potsdam, Germany MUM 13, Dec 2013, Lule, Sweden 2013 ACM. A Cross -device Drag and Drop technique. December 2013
[2] J. Rekimoto. In Proc. UIST 97, pages 3139. ACM, Pick-and-drop: a direct manipulation technique for multiple computer environments. 1997
[3] GeilerIn ACM CHI 98 proceedings, pages 265266, ACM Press, Shu_e, throw or takeit! working efficiently with an interactive wall Manufacturing Technology Vol.-1 . 1998
[4] Jun Rekimoto and Masanori Saitoh. In ACM CHI 99 proceedings,pages 378385, ACM Press, [4] Augmented surfaces: a spatially continuous work space for hybrid computing environments. 1999
[5] N. A. Streitz, J. Geiler, T. Holmer, S. Konomi, C. Mller-Tomfelde, W. Reischl,P. Rexroth, P. Seitz, R. and Steinmetz. In ACMPRESS, i-LAND: an interactive landscape for creativity and innovation 1999
[6] P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski,P. Tandler, B. Bederson, and A.Zierlinger. In Proc. Interact 03, pages 5764. IOS Aress, Drag-and-pop and drag-and-pick:Techniques for accessing remote screen content on touch and pen-operated systems 2003
[7] M. Hasco In HCI2003, Designing for society, Volume 2, pages 7377. British HCI Group,Throwing models for large displays 2003
[8] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith In Proc. AVI 04, pages 2331. ACM, Stitching: Pen Gestures That Span Multiple Displays 2004
[9] M. Collomb, M. Hasco P. Baudisch, and B. Lee In Proceedings of Graphics Interface 2005, Victoria, BC, Improving drag-and-drop on wall-size displays 2005
[10] D. Schmidt, F. Chehimi, E. Rukzio, and H. Gellersen. In Proc. UIST 10, pages 1316.ACM, PhoneTouch: A Technique for Direct Phone Interaction on Surfaces 2010
[11] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg. . In Proc. UIST 11, pages315326. ACM, The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous-Computing Ecologies 2011

