IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-4, Issue-5 E-ISSN: 2347-2693

# Touch Screen Device Swipe n Share

Praful Khokale[1*], Narendra Gunjal[2] , Prof. Suhas B. Gote[3]

[1,2]*Department of Computer Engineering, Shatabdi Institute of engineering & Research, Agaskhind, Sinnar, Nashik*
[3]*Assistant Professor Department of Computer Engineering, S.I.E.R., Nashik*

**Available online at: www.ijcseonline.org**

Received: Mar/10/2016 Revised: Mar/24/2016 Accepted: Apr/17/2016 Published: May/31/2016

***Abstract—*** There are communication commonly extend across smart-phones and devices with wider screens. Indeed, data might be received on the Smart devices but more conveniently processed with an application on a smart device, or vice versa. Such communication require automatic data sharing from a sending location on one screen to a receiving location on the other device screen. We bring out a touch screen device Swipe n Share technique to facilitate these communication involving multiple touchscreen devices, with minimal effort for the user. The technique is a two-handed device gesture, where one hand is used to suitably align the mobile phone with the larger screen, while the other is used to select and swipe an object between devices and choose which device receive the data.

***Keywords- Touch Screen Devices; Mobile devices; data transfer; Swipe-and Share***

## I. INTRODUCTION

Communication 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**[1]**. 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,**[2]** but route directions can be better looked up and printed from a larger device**[3]**. 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**[4]**. In this situation, current practices hinder this process by adding extra steps that divert users from the primary goal of applying data from one device to another Frequently, users recur to typing numbers, names or addresses off a screen because transfer via a sharing protocol is more cumbersome**[5]**. This type of interaction between mobile and situated devices has been widely studied, including recent work focused on interaction with mobile phones as these have become data-rich devices**[6]**. However, we are concerned with cross-device interactions that have distinct characteristics. The interaction is spontaneous, and the data concerned frequently only emerges during interaction, for example as a result of a search, or of a communication received**[7]**. The data does not dictate the application, and users might want to use data items in different ways, for instance apply contact data to an address book, navigation tool, or phone application, depending on their interaction goals.

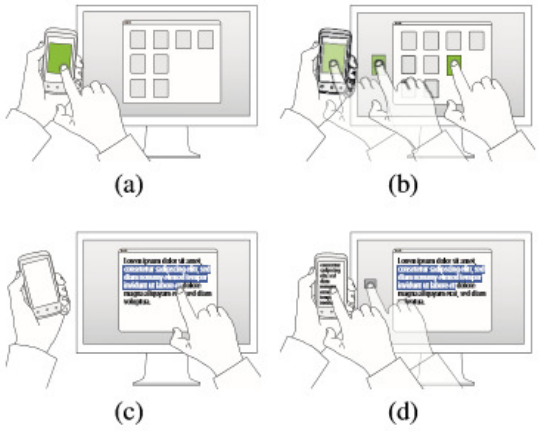


Figure 1: Sharing concept: (a) a user holds the mobile phone next to the desktop screen and selects a data item. (b) The user swipes it inside the screen. (c) In the other direction, a user selects data on the PC and drops it on the phone.

## II. RELATED WORK

A range of techniques have been developed to allow users to transfer data between mobile devices and situated computers. Most mechanisms support fine-grained selection of objects on the source device but only coarse-grained selection of a target device. Assuming device discovery in the background, users can select target devices by name and initiate transfer of selected objects. On the target device, the objects are delivered to a default location, which may be the application used for transfer (e.g., Email) or a dedicated drop box, from where users can pick the data up for further processing. In contrast, Drag-and-Drop is designed for direct point-to-point transfer from a location on one device to a specific target on the receiving device**[8]**. Early work that demonstrated point-to-point interaction across devices includes Pick-and-Drop and Hyper Drag, that relied on

 

instrumentation of the devices or environment. Recent work includes Touch & Interact and Phone Touch, both enabling users to select objects on a phone and to transfer it to a target location on a touchscreen, by directly touching the screen with the phone. However, data transfer from the screen to the phone is generic and requires additional interaction for transfer to a specific target on the phone. Other recent work has been geared towards interaction over a distance, for example by combining touch on a smart phone with pointing at are mote screen to facilitate data transfer to a specific target location on the screen, and vice versa**[9]**. However, this is limited to environments instrumented for gesture tracking. Close to our work is also Mistry et al.'s "SPARSH" concept, envisioning touch-based pick-and-drop between devices . Drag-and-Drop differs from SPARSH as it can integrate with existing technologies, without needing any external cloud-based service. As a technique, Drag-and-Drop is characterized by spontaneous alignment of phone and screen for interaction, and extension of drag-and-drop to work transparently across the temporarily aligned devices. Transparent drag-and-drop across aligned devices was previously demonstrated by Hinckleyetal. In work that investigated pen interaction across multiple displays. Their work emphasized the "stitching" of the involved devices to create a combined display space for cross-device interaction, while we designed our technique for spontaneous transfer where the alignment of the devices is fast and just long enough to facilitate the continuous dragging gesture. Alignment of devices for cross-device interaction has also been investigated in work on proximal interaction and direct touch interaction with phones on touch surfaces**[10]** . In our work, drag-and-drop is extended transparently across devices. Other work has explored techniques for settings in which the target is out of reach. This includes Drag-and-Pop and Drag-and Pick which have been investigated for use on wall-size displays but also demonstrated for interaction across multiple touchscreens . Our technique exploits two-handedness in an asymmetric manner. Guiard's Kinematic Chain model is therefore of key relevance to our work, describing use of the non-dominant hand for setting the frame of reference in which the dominant hand operates [4]. In our case, the non-dominant hand is used to roughly align phone and screen, thus setting the frame of reference for the finer-grained drag-and-drop movement with the other hand.

## III. DESIGN OF TOUCH SCREEN DRAG N DROP

On a conceptual level the Drag-and-Drop technique allows users to drag data from a mobile device and drop it into a desktop screen and vice versa. It is performed through one single uninterrupted touch gesture and hence,it requires that both screens support touch input**[11]**. The technique itself is articulated into four contiguous phases: placing the mobile device in proximity of the desktop screen, identifying the source object of the drag-and-drop action, performing the actual movement into the other screen and finally, identifying the target destination where the content being dragged will be "dropped". The first phase builds on the assumption that placing the mobile device next to the screen will form an ideal "bond" between them allowing users to perceive the two different screens as contiguous**[12]**. This requirement is mainly needed to instil the idea that it is possible to drag data "outside" the physical boundaries of one screen and inside another one due to their close proximity to each other. In fact, the devices have no knowledge of their respective locations and do not need to: it makes the way it works seem more easily understandable by first time users. Although we have mainly focused on the technique itself, this "bonding" moment could be also used to establish the pairing itself between the devices. In our implementation we used an explicit point-to-point connection which has to be initiated before each session and lasts until the mobile device is explicitly disconnected**[13]**. Sensorsplacedon the sidesof the desktop screen could streamline the process.

## IV. IMPLEMENTATION

Our prototype implementation consists of a server application on the PC and two different clients on the mobile phone, communicating over a wireless network. The desktop, a Windows 7 PC connected to a23"HP2310titouch screen, runs the PC Drag Detector, an application that handles touch detection and networking. The mobile phone, an Android 4.0.3 device, alternatively runs one of the two client applications which address different usage scenarios: first, the Bridge Application enables existent applications on the phone(e.g., a map applications or image galleries) to use Drag-and Drop without modifications; second, the email client demonstrates an application extended by Drag-and-Drop to simulate a possible future native integration of our technique by the operative system

### IV.I. 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. 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.

### IV.II. The Bridge Application

The bridge application, the first of our two mobile implementations, connects Drag – and – Drop to existent applications on the same phone. It handles bidirectional interactions between desktop and mobile, using two different UIs described in the following.

**Receiving Data** from the Desktop 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 a range of typical usage scenarios. In particular, we are covering 1) phone diallers, 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 maps icon on the phone to immediately start the navigation. By applying data onto a particular application class, the user disambiguates how the phone handles the corresponding data. 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 diallers icon to immediately get connected. In doing so, Drag-and-Drop offers a quick and convenient alternative to manually transferring the required information from the desktop to the mobile phone, for example by re-typing it. From a technical point of view, once the user starts a drag gesture on the desktop, the finger will eventually enter a drag detector window. The PC application then queries the data to discern its underlying type and provides feedback on the mobile once the finger enters its screen. Then, an icon representing the dragged data is displayed as a preview and moves alongside the user's finger. Feedback about the validity of the drop location on the mobile is shown to the user through background changes of the currently highlighted application class (i.e., yellow for a valid drop target, red otherwise). When the finger is released, the dragged data is applied to the chosen class. Through a mechanism called implicit intents, the Android OS determines eligible applications that can handle the data from those available in the user's device.

**Sending Data** to the 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 viewed or edited on a larger screen. In order to support this, we use Android's 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 device 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 different ways (e.g.: a picture sent as a URL or as a binary file). From a technical perspective, once the finger enters the detector window on the desktop, the dragged data will be encapsulated into a simulated local drag-and-drop event, transparent to both the user and the target application on the PC.

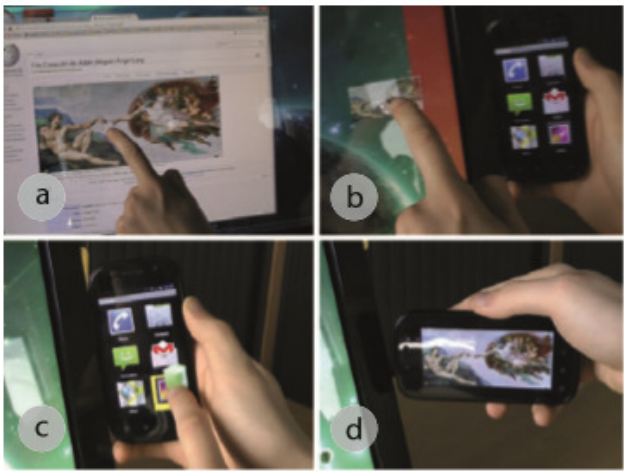


Figure 2: Picture transfer from desktop to phone: the user selects a picture displayed in a webpage(a) and proceeds to drag it across the screen and over the semi-transparent detect or window (b); the drag gesture is continued on to the mobile screen where the user drops the picture over the viewer icon (c); finally, the picture is automatically displayed(d). The technique also works in the opposite direction.

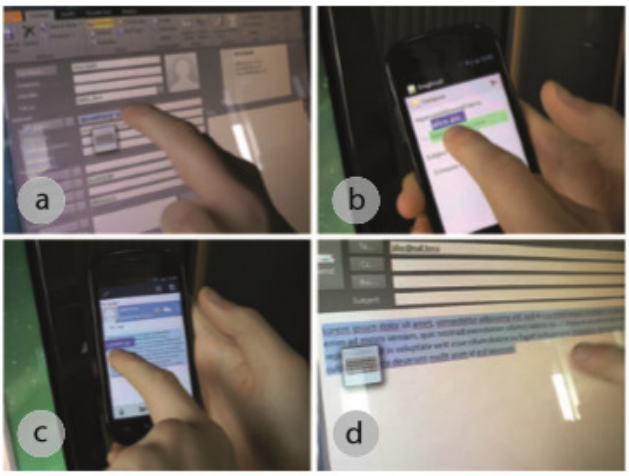


Figure 3: To send an email address available on a PC to a mobile email client, the user can drag it towards the phone (a); once on the mobile, the user can drop it over the relevant field. To copy text available on a mobile to a desktop application, the user selects it normally on the mobile and drags it towards the screen (c); once on the desktop screen, the user can release the finger over any application accepting text to apply it there.

Actions where data originating from a mobile device is better viewed or edited on a larger screen. In order to support this, we use Android's 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 device 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 user scan not 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 different ways (e.g.: a picture sent as a URL or as a binary file). From a technical perspective, once the finger enters the detector window on the desktop, the dragged data will be encapsulated into a simulated local drag-and-drop event, transparent to both the user and the target application on the PC.

## CONCLUSION

In this paper we have presented a novel interaction technique that allows user to intuitively share data between desktop computers and mobile devices. Through touch gestures, the cross-device Swipe n Share technique provides a practical solution to a need that is still today not adequately supported. The user feedback we gained highlighted its positive aspects of being easy to learn and perform. We plan to further investigate how to adapt the technique for use in public and semi-public settings such as train stations, airports, museums, retail stores, etc.

## REFERENCES

[1] P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch and pen-operated systems. In Proc. Interact '03, pages 57–64. IOS Aress, 2003[1].

[2] A. Bragdon, R. DeLine, K. Hinckley, and M. R. Morris. Code Space: Touch + Air Gesture Hybrid Interactions For Supporting Developer Meetings. In Proc. ITS '11, pages 212–221. ACM, 2011[2].

[3] M. Collomb and M. Hascoët. Extending drag-and-drop to new interactive environments: A multi-display, multi-instrument and multi-user approach. Interacting with Computers, 20(6):562 – 573, 2008[3].

[4] Y. Guiard. Asymmetric division of labour in human skilled bimanual action: The kinematic chain as a model. Journal of Motor Behaviour, 19:486–517, 1987[4].

[5] R. Hardy and E. Rukzio. Touch & Interact: Touch-based interaction of mobile phones with displays. In Proc. MobileHCI, pages 245–254. ACM, 2008[5].

[6] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith. Stitching: Pen Gestures That Span Multiple Displays. In Proc. AVI '04, pages 23–31. ACM, 2004[6].

[7] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In Proc. UIST '11, pages 315–326. ACM, 2011[7].

[8] P. Mistry, S. Nanayakkara, and P. Maes. Touch and Copy, Touch and Paste. In Proc. CHI EA '11, pages 1095–1098. ACM, 2011[8].

[9] B. A. Myers. Using handhelds and PCs together. Comm. ACM, 44:34–41, 2001[9].

[10] J. Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In Proc. UIST '97, pages 31–39. ACM, 1997[10].

[11] J. Rekimoto and M. Saitoh. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In Proc. CHI '99, pages 378–385. ACM, 1999[11].

[12] D. Schmidt, F. Chehimi, E. Rukzio, and H. Gellersen. PhoneTouch: A Technique for Direct Phone Interaction on Surfaces. In Proc. UIST '10, pages 13–16. ACM, 2010[12].

[13] A. D. Wilson and R. Sarin. BlueTable: Connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In Proc. GI '07, pages 119–125, 2007[13].

 