SyncTap: An Interaction Technique for Mobile Networking

Jun Rekimoto, Yuji Ayatsuka, and Michimune Kohno

Interaction Laboratory, Sony Computer Science Laboratories, Inc.
3-14-13 Higashigotanda, Shinagawa-ku, Tokyo 141-0022 Japan
{rekimoto,aya,mkohno}@csl.sony.co.jp
http://www.csl.sony.co.jp/person/rekimoto.html

Abstract. This paper introduces "SyncTap", a user interface technique for making a network connection between digital devices. When a user wants to connect two devices, he or she synchronously presses and releases the "connection" buttons on both devices. Then, multicast packets that contain press and release timing are sent to the network. By comparing this timing with locally recorded one, both devices correctly identify each other. This scheme is simple but scalable because it can detect and handle simultaneous overlapping connection requests. It can also be used for making secure connections by exchanging public keys. This paper describes the principle, the protocol, and applications of SyncTap.

1 Introduction

When many networked devices – ranging from personal computers to various digital appliances – are used in combination, to provide an intuitive user interface with which people can easily and correctly establish network connections becomes important[1]. For example, we may frequently need to create ad-hoc network connections for multiple purposes:

– Printing a hardcopy of a document contained in your PDA with an available printer nearby.
– Showing presentation data on a meeting room screen. Your note-PC transmits data to the presentation computer using wireless networking.
– Your PDA becomes a remote commander for a TV at hand.
– At public wireless LAN hotspot, you transfer files to your colleague's computer, but you would like to make sure this transmission is secure.

These network connections are different from traditional network communication. These connections are frequently made and broken according to users'

[1] In this paper, we use the term "connection" to refer a network service association, and we also assume all the devices already have IP packet access. For example, when printing a document from the PDA to the printer, these two devices must have an "association", by knowing each other's addresses, and by optionally sharing a session key for security.

real world activities, and the durations of these connections are generally short. These differences create new user interface challenges.

Traditionally, a device's network addresses, such as IP addresses or machine names, are used to specify devices. However, as network configuration becomes more complex and dynamic, such address-based targeting becomes ineffective. To inspect IP addresses is often a tedious task. In particular, finding IP addresses of digital appliances with limited IO capability is not easy (i.e., getting IP address of a printer often requires unfamiliar maintenance commands). In addition, as dynamic host configuration protocol (DHCP) becomes popular, many devices use dynamically assigned network addresses, and this makes situations even more difficult for users.

To address these situations, more "direct" ways to specify target devices are desirable. For example, an infrared beam that contains an IP address could be used; a user could "beam" a target device with a mobile device, and that triggers a wireless communication between the two [1, 2]. To create a secure connection, infrared beaming might contain a one-time session key. This solution works if all the devices have same sensors, but in reality this assumption is not always valid.

This paper proposes an even simpler way to dealing with such a problem with minimum hardware and sensor requirements. One only assumes that both devices have at least one button, called the "SyncTap" button.

The SyncTap button is used to create a network connection. When a user wants to establish a network connection, he or she synchronously presses and releases SyncTap buttons on both devices (Figure 1). By checking this press-release synchronicity, both side can correctly identify each other and can establish a network connection. As we explain in more detail later, the SyncTap button can be used for other purposes (e.g., a keyboard key such as the "Escape" key can be used as a SyncTap button without neglecting its original key function).

More generally, SyncTap is the simplest instance of synchronous actions. The concept of synchronous actions is that when two networked devices are operated with synchronizied operations, such as button taps, keystrokes, mouse strokes, pen gestures, light changes, device motions, or even voice inputs, these device

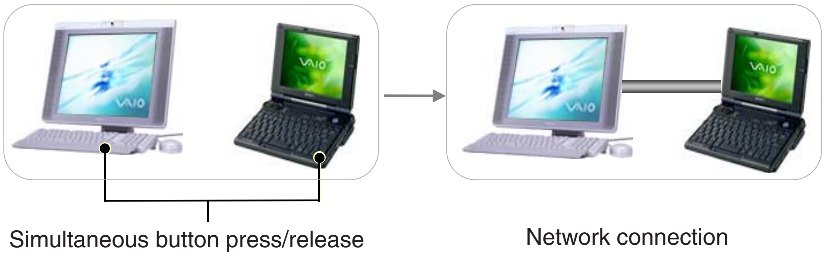

Fig. 1. SyncTap: user synchronously presses and releases the buttons on the devices, then these device establish a network communication

should be able to identify each other by multicasting the received operation information on the network.

2 SyncTap: Synchronous User Actions for Creating Device Association

2.1 Principle

The idea of SyncTap is very simple. Assume that one needs to *connect* two network devices such as a notebook PC and a digital camera. To achieve this, one first synchronously presses and releases SyncTap buttons on both devices. On releasing buttons, both the devices multicast user datagram protocol (UDP) packets that contain:

- Time interval between button press and release,
- Sender's IP address, and
- (Optional) public key information for secure connection.

Since these are multicast packets, all the devices (not limited to devices with which the user is interacting) that listen to a specific UDP port receive them (Figure 2). Both the devices also receive the packets, and checks if this connection request if for them, by comparing:

- the (locally recorded) button release time and the packet arrival time, and
- the (locally recorded) interval between button press-release, and the corresponding time contained in the packet.

Accounting for the fact that the local button release time differs from the packet arrival time due to network delay, and the accuracy of human performance, we set a fixed error limit to C1 and C2. In our current implementation, this is ranging between 100-200ms. When values are within these limits, the device recognizes that the other end is requesting a connection. Note that this check does not directly compare timestamps on the device A and device B, thus synchronized clocks are not necessary.

2.2 Selection of SyncTap Buttons

SyncTap assumes devices have network connectivity and have at least one button (we refer to this as the "SyncTap" button) for operation. The button can be newly installed or an existing one such as a keyboard key of a PC. It can also be implemented as GUI (on screen) buttons.

Our current prototype uses the Escape key or the Shift keys as SyncTap buttons for PCs. These keys also act as normal keyboard keys. For example, when the Escape key is pressed and immediately released (e.g., within 500 ms), the operation is treated as a normal use. When the press-release interval exceeds a predefined time, the operation is handled as a SyncTap operation. The Shift keys are handled similarly. When the Shift key is pressed and released without any other keystroke in-between, the operation is treated as SyncTap; otherwise, it is treated as a normal shift. These techniques greatly reduce the number of unnecessary SyncTap packets and, thus, reduce the probability of collision (described in the next section).

2.3 Collision Detection

When other sets of devices also try to establish another network connection, other SyncTap multicast packets might be transmitted. The system can detect this "collision" situation by collecting all the multicast packets that arrive within a certain time interval around the local button release time. If two or more packets arrive, the device regards it as a collision, and asks a user to press SyncTap button again. The device also records the IP addresses of these multicast sources. In the next trial, the device only accepts multicast packets from these recorded IP addresses (Figure 4).

By using this simple scheme, even though the first SyncTap operation fails because of colliding with other operations, the second try is almost collision free

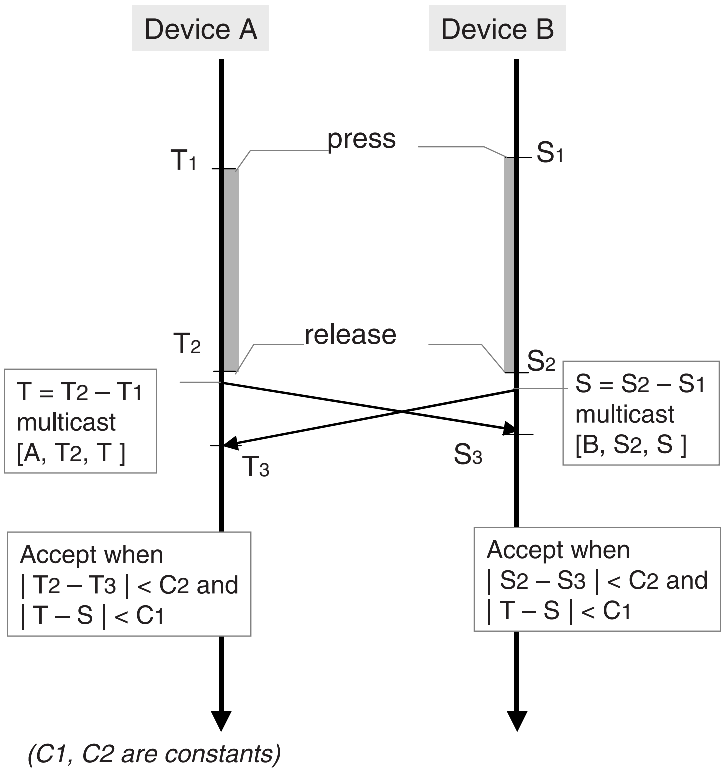

Fig. 2. SyncTap packet exchange protocol

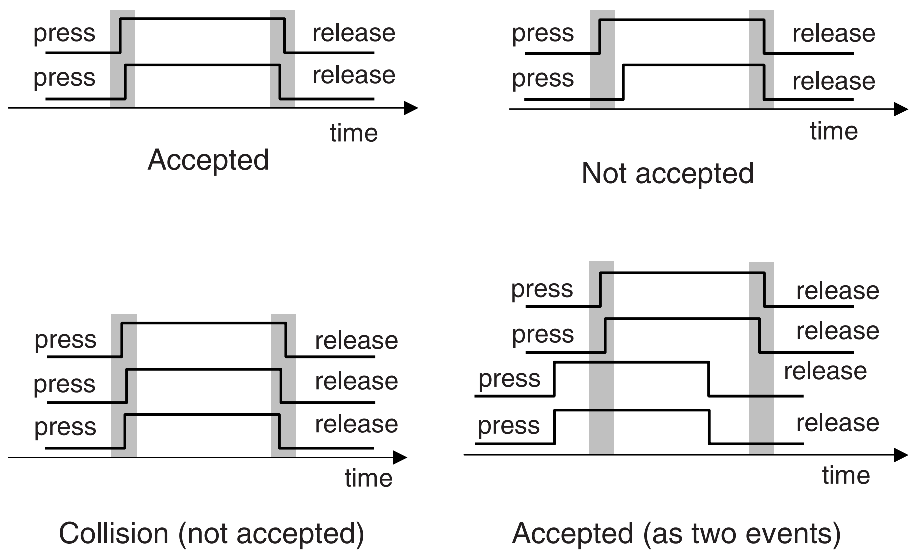

Fig. 3. Packets timings are used to distinguish SyncTap events from other collisions

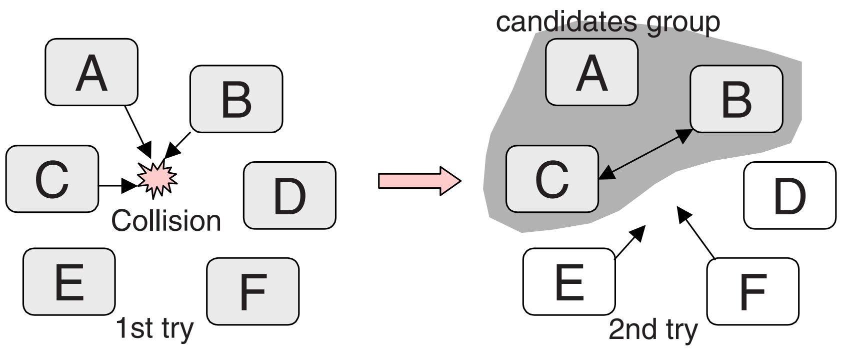

Fig. 4. Multiple overlapping SyncTap operations can be detected and handled

because the number of connection candidates is greatly reduced (limited to the message senders of the first trial). This feature makes SyncTap scalable, and usable in an environment where many (typically several hundreds) devices are on the same LAN segment.

2.4 Secure Communication

Protecting wireless communication is important, especially when using a public wireless service (i.e., *Hotspot*). SyncTap can also be used for creating a shared session key for secure communication by piggybacking Diffie-Hellman public keys on multicast packets (Figure 5). Each device generates and exchanges public keys (Ya, Yb) by using multicast packets. These public keys are used to calculate a shared secret session key for encrypted communication.

Normally, the Diffie-Hellman algorithm is subject to the ”man-in-the-middle” problem and requires an additional method to authenticate the end points. Using SyncTap, however, becoming the ”man-in-the-middle” is very difficult because it requires interception of all the multicast packets and transmission of faked packets substituted for them. Since SyncTap is used for connecting nearby devices, devices can easily give immediate connection feedback (e.g., showing a message on a screen, blinking LEDs, etc.); thus, hidden man-in-the-middle hosts can be easily detected. As a result, a simple public key exchange scheme is reliable enough in practical situations.

3 Application Examples

As described in the previous section, the SyncTap is an easy and intuitive way to establish a network connection between various types of digital devices. This section explains how this technique is used in realistic contexts with several examples.

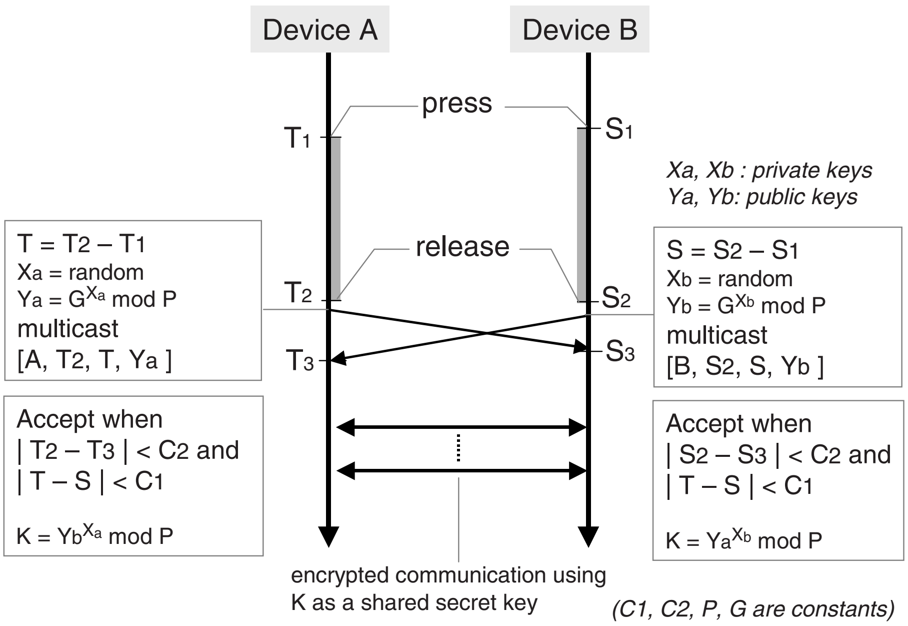

Fig. 5. Establishing secure communication: The Diffie-Hellman key exchange protocol is used to create a shared secret key K

3.1 Instant Connection between PCs and Appliances

Suppose one has a digital camera that is capable of connecting to a wireless network. After taking photographs, one would like to transfer pictures to one's notebook PC. Then, one presses the shutter button and the Shift-key (of the notebook PC) synchronously, and wireless communication between the PC and the camera is established (Figure 6 (a)). Then, a window corresponding to the camera appears on the computer screen and one can drag picture files from this window to one's PC's document folder.

3.2 Ad-Hoc Connection at a Hotspot, or a Meeting Room

Suppose that one is at a public lounge and using a hotspot service. One would like to exchange a file with one's colleague. One can both presses both SyncTab buttons on devices to create a connection (Figure 6 (b)). Then both devices exchange Diffie-Hellman public keys and secure communication starts.

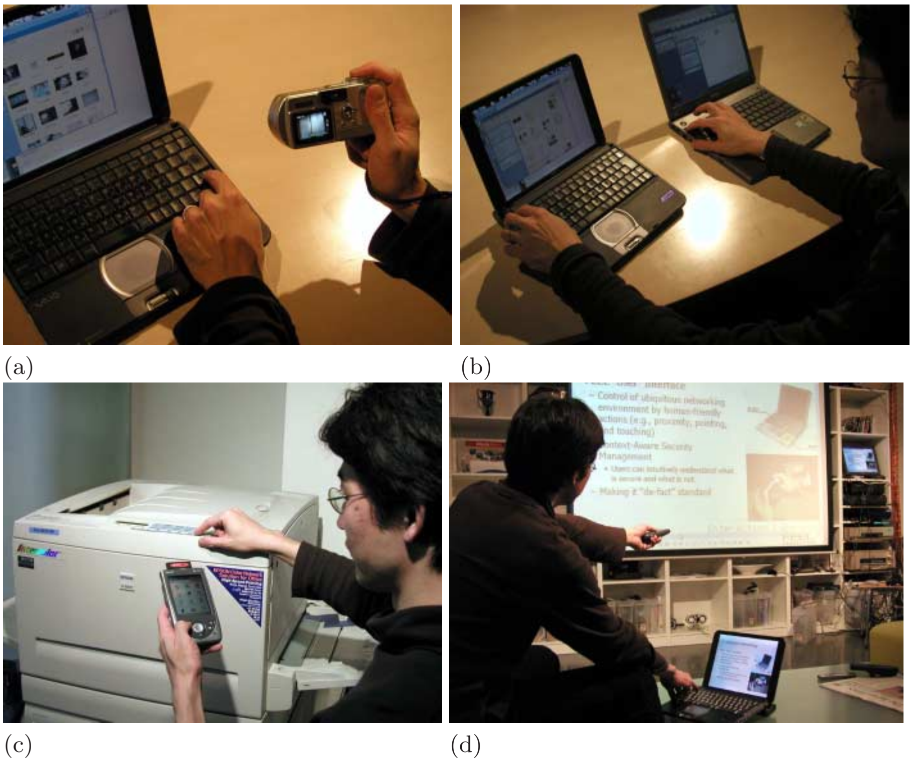

Fig. 6. SyncTap operations can be used in various settings. (a) A Digital camera and a PC. (b) Connecting two PCs. (c) Printing a document from a PDA to a nearby printer. (d) Presentation using wireless connection between a notebook PC and a presentation screen

3.3 Printing

When one wishes to print a document that is listed on one's PDA to a nearby printer, one presses the PDA's and the printer's SyncTap buttons. Then a printer icon appears on ones PDA screen, and one can drag a document icon to the printer icon (Figure 6 (c)). Note that the contents of the documents might be contained in the PDA's flash memory, or the PDA may only manage links (e.g,, URLs) to documents.

3.4 Presentation

When presenting a slide show in a meeting room, by using a wireless connection, a user transfers slide data from his or her computer to the presentation computer. To do this, one simultaneously presses and releases the remote controller's SyncTap button and the PC's SyncTap button (Figure 6 (d)). The presentation computer receives an IR beam from the remote controller, and a network connection between the two (notebook and presentation computer) is established.

This example demonstrates how SyncTap can be used when two devices are not within an arm's length. A simple intermediate device, such as an IR remote controller, can be used as a remote SyncTap button (Figure 7). In this

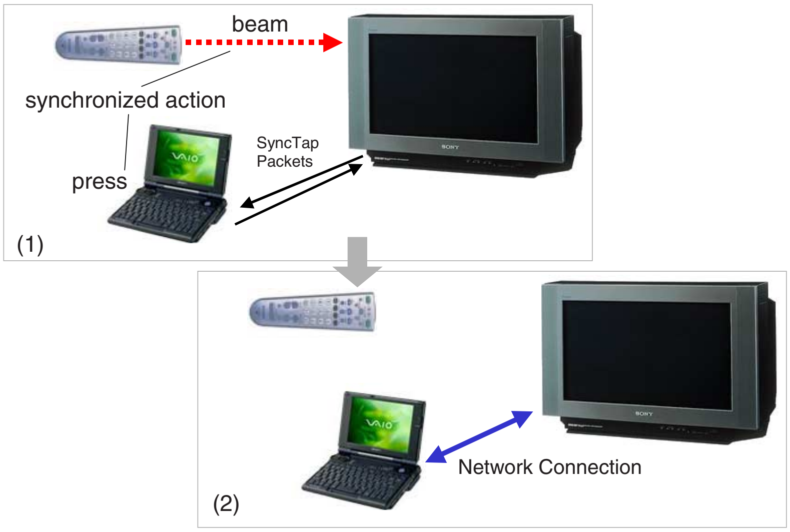

Fig. 7. Combination of infrared beaming and SyncTap. The IR commander is just used as a "remote" SyncTap button. The time of the beaming and button press on the PC are synchronized (1). Then a network connection between TV and note PC is established (2). Note that no previous settings are required for this combination

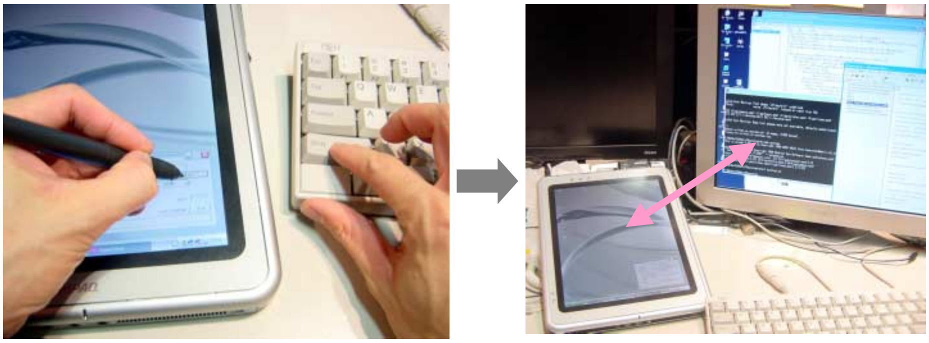

Fig. 8. SyncTap for setting up the HyperCursor connection

case, the controller is only used to transmit press and release timings, and thus, transmitting any complicated data, such as the address of the target device, by using the IR beaming, is unnecessary.

3.5 Using SyncTap for HyperCursor[3] Communication Setup

As a part of our "Augmented Surfaces" system, we previously implemented a migrateable cursor system called HyperCursor [3]. Using HyperCursor, a user can control two computers by using a single mouse and keyboard. When the cursor reaches the edge of one computer, the cursor automatically "jumps" to the next computer. Keyboard inputs are also delivered to the second computer. A user can also drag an object from one computer to another across the boundary of their screens.

The original HyperCurosr relies on a camera sensor that recognizes computer positions. The sensor enables spatially continuous operation because logical mouse movements reflect the physical positional layouts of computers. For example, when a user places the second computer on the first computer's left, the cursor is configured to jump through the left edge of the first computer's screen. However, without such sensors, users had to manually configure the environment. This is cumbersome especially when mobile computers are using DHCP.

A combination of HyperCursor and SyncTap addresses the problem. For example, when a user brings a tablet PC (without keyboard) to his or her office's desk, and wants to manipulate the tablet PC with his or her desktop PC's keyboard and mouse. To set this, a user simply presses the PC's SyncTap button (e.g., Escape key) and synchronously taps the tablet PC's screen[2]. Next, one controls the mouse to hit the screen edge of the desktop computer screen. This operation tells the system about the relative location of the tablet PC, and the cursor automatically jumps to the tablet PC's screen. We are also trying another

[2] Some of the tablet PC have physical "hot" buttons, and these buttons are often act as the "Escape" key. In this case, these hot buttons can also be used as a SyncTap button.

method in which a user can specify physical layout by choosing SyncTap buttons. For example, pressing the left Shift key on a device tells the system that the other device is placed on the left.)

4 Discussions

4.1 Related Work

The work described in this paper is inspired by a series of previous systems that tried to connect nearby devices using physical actions [4, 3, 2, 1]. Those systems relied on additional sensors, such as radio frequency identification (RFID) tags, infrared beaming, or barcodes. These systems become ineffective when some of the devices do not have these sensors.

Some recent work on network services try to provide a method for accessing network resources by using understandable names such as ”Kate's PC” or ”the printer in the copier room” [5]. Users could choose the target device by selecting an item from a menu. However, maintaining the long list of such names still requires considerable effort. Some digital devices, such as wireless headsets, do not have a screen and thus GUI-based selection is not available. SyncTap can coexists with these technologies and act as a ”greatest common denominator” because of its minimum hardware requirements.

In the presentation example we showed how a simple intermediate device, such as an infrared controller, could be used as a ”remote” SyncTap button. We are also considering other types of intermediate devices, such as similar to the Pick-and-Drop pen[6]. While the pick-and-drop technique mainly handles data, this intermediate device also handles network connection.

Although this paper mainly focuses on network connection, user interfaces *after* the connection is established are equally important. Holland et al. proposed a method called ”Dynamic Combination”, that effectively chooses available operations based on the combination of selected devices [7]. For example, when selecting two devices, such as a PDA and a Printer, the number of the possible operations would be greatly reduced. Thus, the user interface could be simplified by first connecting two devices, then choosing a command. We consider that similar techniques can be used with SyncTap.

4.2 Human Operation Accuracy

SyncTap relies on humans to perform synchronous operations by using both hands. To effectively distinguish SyncTap events from other coincidental operations, a selected threshold (maximum allowable time lapse as SyncTap event pairs) is important. When this value is too small, some SyncTap actions are not correctly recognized. On the other hand, when this value is too large, the probability of collisions increases.

We actually measured human performance accuracy by providing SyncTap software to several users. The average time lapse between two SyncTap actions is 27ms. Based on this measurement, we currently use +- 50ms as a threshold. Figure 9 shows an actual timing chart when SyncTap pairs are being established.

We also installed a HyperCursor (a remote cursor) system that uses SyncTap on 15 PCs in our laboratory, and five people actively used this system. During the one-month trial, no collisions were detected. This is mainly because the number of simultaneous users was small. We are currently planning to distribute this system throughout the entire laboratory and investigate the probability of collisions and their effects on usability.

4.3 Other Synchronous User Operations Possibilities

Although this paper mainly treats button pressing, mouse clicking, and IR beaming as methods for initiating SyncTap, many other user actions could be used. Two interesting examples are shown in Figure 10. The first one is to use a button of the first device is used to press the button on another. For example, if the tip of the cellular phone antenna was a button, it could be used to "press" other devices' SyncTap buttons. This style might be more natural than using both hands, and offer a metaphor, which is similar to real world actions such as connecting a plug to the socket, or inserting a key into the keyhole.

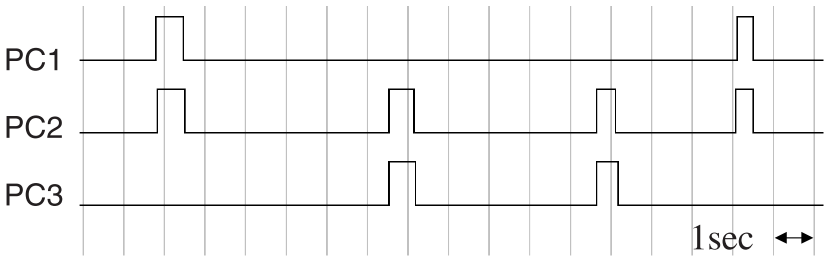

Fig. 9. Timing records when establishing SyncTap pairs

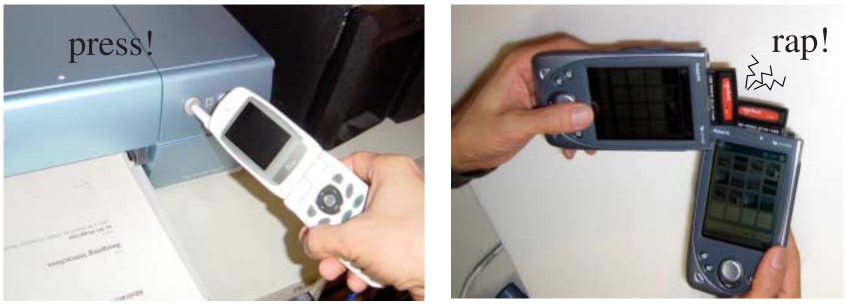

Fig. 10. Variations of SyncTap operations (left: a button of one device is used to "press" the other device's SyncTap button, Right: knocking on the device with the other device, causes synchronously generated sound)

Another possibility is to detect synchronous sensor values, such as sound. Figure 10 right shows how one device is used to ”knock” on the other device, the resulting sound captured by the both would be similar and synchronized. By comparing this similarity, creating a SyncTap pair between these two devices should be possible.

5 Conclusion

This paper presents the SyncTap method, a simple user interface for making network connections between digital devices. Unlike previous systems that assume various sensors for device identification, SyncTap only assumes both devices have human-controllable buttons, and uses synchronous timing as an identification method.

References

[1] Rekimoto, J., Ayatsuka, Y., Kohno, M., Oba, H.: Proximal Interactions : A direct manipulation technique for wireless networking. (In: to appear in proc. INTERACT 2003) 105, 113

[2] Swindells, C., Inkpen, K. M., Dill, J. C., Tory, M.: That one there! pointing to establish device identity. In: Symposium on User Interface Software and Technology (UIST’02). (2002) 151–160 105, 113

[3] Rekimoto, J., Saitoh, M.: Augmented Surfaces: A spatially continuous workspace for hybrid computing environments. In: Proceedings of ACM CHI’99. (1999) 378–385 112, 113

[4] Want, R., Fishkin, K. P., Gujar, A., Harrison, B. L.: Bridging physical and virtual worlds with electronic tags. In: CHI’99 Proceedings. (1999) 370–377 113

[5] Zero Configuration Networking: (`http://www.zeroconf.org`) 113

[6] Rekimoto, J.: Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In: Proceedings of UIST’97. (1997) 31–39 113

[7] Holland, S., Oppenheim, D.: Direct combination. In: Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems. (1999) 262–269 113