Pers Ubiquit Comput (2004) 8: 126–134
DOI 10.1007/s00779-004-0262-2

ORIGINAL ARTICLE

Jun Rekimoto

SyncTap: synchronous user operation for spontaneous network connection

Received: 5 December 2003 / Accepted: 10 March 2004 / Published online: 1 May 2004

Abstract This paper introduces the concept of synchronous user operation, a user interface technique for establishing spontaneous network connections between digital devices. This concept has been implemented in the ''SyncTap system'', which allows a user to establish device connections through synchronous button operations. When the user wants to connect two devices, she synchronously presses and releases the ''connection'' buttons on both devices. Then, multicast packets containing button press and release timing information are sent through the network. By comparing this timing information with locally recorded information, the devices can 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 to establish secure connections by exchanging public keys. This paper describes the principle, the protocol, and various applications in the domain of ubiquitous computing.

Keywords Synchronous operation · Resource discovery · Spontaneous networking · Wireless communication · Mobile computing · Security · SyncTap

1 Introduction

In ubiquitous computing environments, many networked devices—ranging from personal computers to various digital appliances—are used in combination. For example, users may need to frequently create network connections for various purposes:

- To print a hard copy of a document from a PDA on a nearby available printer.
- To show presentation data on a meeting room screen. The user's notebook PC can transmit data to the presentation computer by using wireless networking.
- To use a PDA as a remote controller for a nearby television.
- To transfer files to a colleague's computer at a public wireless LAN hotspot. In this case, the user would like to ensure that the transmission is secure.

All of these instances of network connections are different from traditional network communication in that they are inherently *spontaneous*. That is, these types of connections are frequently made and broken, according to users' real-world activities, and the durations of the connections are generally short. In addition, the necessity of connecting to nearby devices is increasing, as illustrated by the above example scenarios.

Traditionally, symbolic identifiers, such as IP addresses or machine names, are used to specify devices. As network configurations become more complex and dynamic, however, such address-based targeting becomes ineffective. Inspecting IP addresses or machine names is often a tedious task. In particular, finding the addresses of digital appliances with limited input–output capability is not easy. For example, getting the IP address of a printer often requires using unfamiliar maintenance commands. In addition, as the dynamic host configuration protocol (DHCP) becomes more popular, many devices are using dynamically assigned network addresses, making the situation even more difficult for users.

To address these issues, more direct and intuitive ways to specify target devices are needed. For example, an infrared beam containing an IP address could be used: a user could ''beam'' a target device with a mobile device in order to trigger wireless communication between the two [1, 2]. To create a secure connection, the

J. Rekimoto (✉)
Interaction Laboratory,
Sony Computer Science Laboratories,
Inc., 3-14-13 Higashigotanda,
Shinagawa-ku, Tokyo, 141-0022, Japan
E-mail: rekimoto@acm.org
Tel.: +81-3-54484380
Fax: +81-3-54484273

infrared beam might contain a one-time session key. This solution works if all of the devices have the same sensors, but in reality, this is not always the case.

This paper proposes a simple but effective way to deal with such problems, which is called *synchronous operation*, with only minimal hardware and sensor requirements [3]. This refers to the idea that, when two networked devices are operated with synchronized timing, based on operations such as button presses, keystrokes, mouse movements, pen gestures, light changes, device motions, or even voice inputs, the devices should be able to identify each other by multicasting the received operation information on the network and finding each other by matching the timing information (Fig. 1).

This paper also focuses on one of the simplest cases of synchronous operation, in which we only assume that each device has at least one button, called the ''SyncTap'' button. When a user wants to establish a network connection, she synchronously presses and releases the SyncTap buttons on both devices. By checking this press–release synchronicity, the devices can correctly identify each other and establish a network connection. As we explain in more detail later, the SyncTap button can also be used for other purposes (e.g., a key such as the ''Escape'' key on a keyboard can be used as a SyncTap button without eliminating its original function).

2 Related work

The work described in this paper is inspired by a series of previous systems that tried to enable connection to nearby devices by using physical actions on the part of the user [1, 2, 4–6]. Those systems relied on additional sensor techniques, such as radio frequency identification (RFID) tags, infrared beaming, or barcodes. They become ineffective with devices that do not have the required sensors.

Some recent work on network services has tried to provide a method for accessing network resources by using understandable names, such as ''Kate's PC'' or ''the printer in the copier room'' [7]. With this approach, users could choose the target device by selecting an item from a menu. Maintaining the long list of such names, however, would still require considerable effort. Some digital devices, such as wireless headsets, do not have displays and, thus, GUI-based selection is not available. On the other hand, our group's proposed synchronous operation method can coexist with these technologies and act as a ''greatest common denominator'' because of its minimal hardware requirements.

Some of the recently proposed systems have also used synchronous events on a network to achieve spontaneous device connection. ''Smart-Its Friends'' [8] uses a motion sensor to connect two devices when they are held together and moved (so that the motion sensors in each device report similar sensor value sequences). After our group's initial publication on SyncTap [3], Hinckley et al. also reported a system using motion sensors, so that when one computer bumps into another, they are connected [9]. Whereas these motion-based systems require special sensors that might not be installed in today's digital appliances, our proposed synchronous operation concept does not require any special sensing hardware. The proposed method instead relies on the synchronicity of a user's operations rather than on sensor values. In addition, motion-sensing methods implicitly assume that the devices to be connected are all mobile devices, whilst our method works with both mobile and static devices.

3 SyncTap: synchronous user actions for creating a device association

In this section, I explain the actual mechanism of SyncTap, which handles two types of network connections. The first type, described in Sect. 3.1, assumes that the devices already have network access and that ''connection'' means one device finding another device and establishing a service connection. For example, when printing a document from a PDA to a nearby printer, these two devices must find each other. The second type of connection, described in Sect. 3.4, means the creation of a network connection itself. For example, when people want to exchange data between two wireless devices without having any wireless infrastructure, they have to create an ad-hoc wireless connection between the devices.

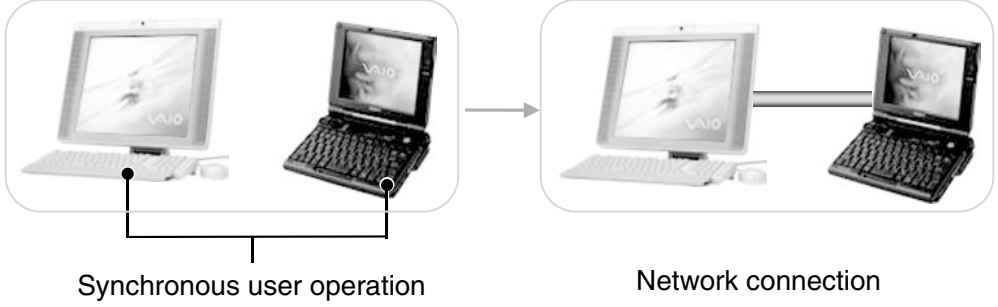

Fig. 1 Synchronous user operation: the user synchronously manipulates both devices, such as by pressing and releasing keys on each device. The devices can then find each other and establish network communication

3.1 Principle

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

- The time interval between the button press and release
- The sender's IP address, and
- Public key information for establishing a secure connection (optional)

Since these are multicast packets, all of the devices on the same network segment (i.e., not limited to devices with which the user is interacting) that are listening to a specific multicast port receive them (Fig. 2). This includes both of the devices that the user is attempting to connect. The two devices check whether this connection request is for them by comparing each of the following:

- The (locally recorded) button release time, and the packet arrival time, and
- The (locally recorded) interval between the button press and 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, as well as the accuracy of human performance, we currently apply fixed error limits to detect synchronicity. The first limit (C1 in Fig. 2) is the difference in the intervals between the button press and release. This difference depends on the accuracy of the human user. We currently use a value of 100 ms for C1. The other limit (C2 in Fig. 2) is the difference between the locally recorded button release time and the arrival time of the packet from the other device. This difference is caused by the multicast packet transmission delay. We currently use a value of 200 ms for C2.

When the comparison values are within these limits, each device recognizes that the other one is requesting a connection; otherwise, the devices simply ignore the

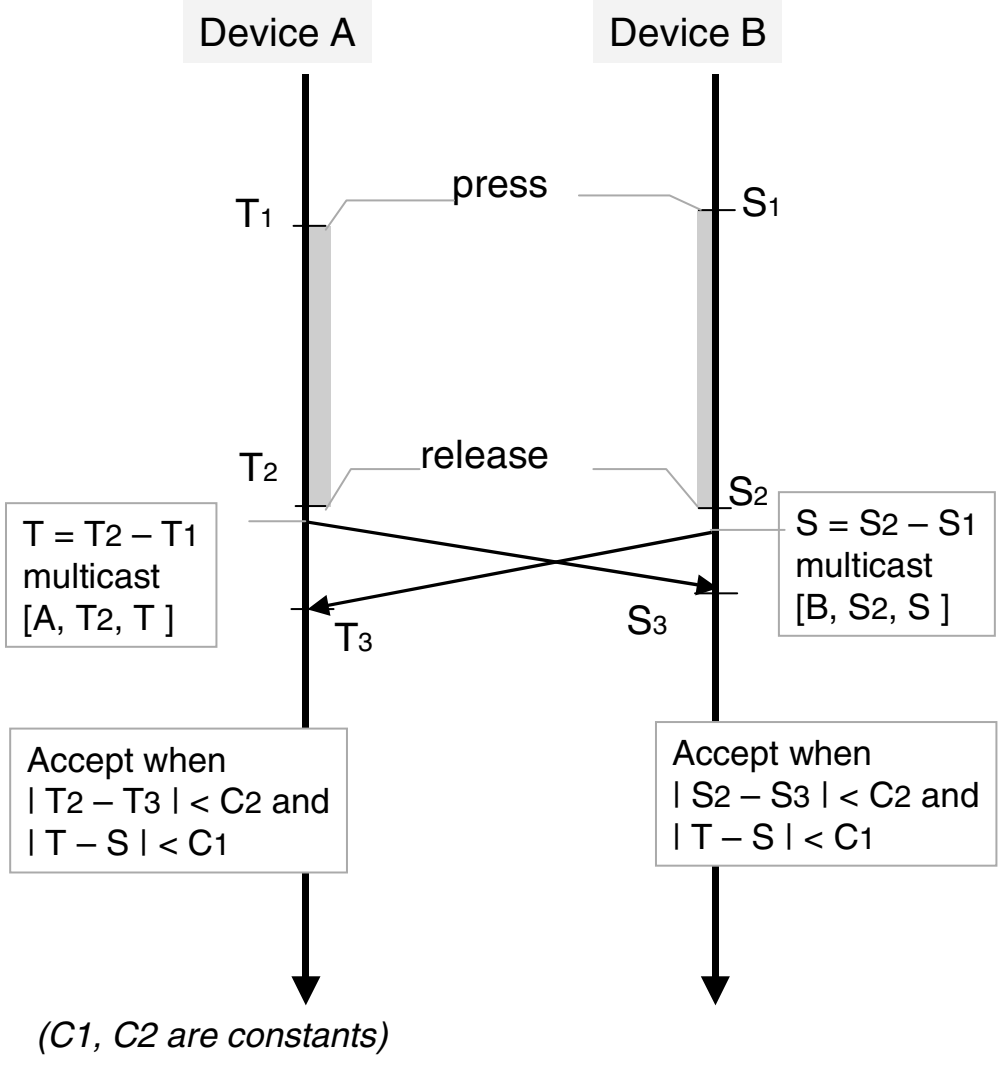

Fig. 2 SyncTap packet exchange protocol

received SyncTap packets because they must be a connection request for another device pair. Note that this check does not directly compare timestamps from each device. Thus, this protocol works even when the clocks on the connecting devices are not synchronized.

3.2 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 having a connecting device collect all of the multicast packets that arrive within a certain time interval around the local button release time (Fig. 3). If two or more packets arrive within this window, the device regards it as a collision and asks the user to press the SyncTap buttons again. The device also records the IP addresses of these multicast sources. Then, in the next attempt, it only accepts multicast packets from these recorded IP addresses (Fig. 4).

By using this simple scheme, even if the first SyncTap operation fails due to a collision with other operations, the second try will almost always be free of collisions because the number of connection candidates is greatly reduced (i.e., it is limited to the message senders from the first attempt). This feature makes SyncTap scalable and still usable in an environment where many (typically, several hundred) devices are on the same LAN segment, with only one SyncTap operation, or possibly a few extra in case of a collision.

3.3 Secure communication

Protecting wireless communication is important, especially when using a public wireless service (i.e., a hotspot). SyncTap can also be used to create a shared session key for secure communication by piggybacking Diffie–Hellman public keys [10] on multicast packets (Fig. 5). Each device generates and exchanges public keys (Ya, Yb) by using multicast packets. These public keys are then 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. With SyncTap, however, it is very difficult to create a man-in-the-middle situation, because it would require the intercepting of all the multicast packets and transmitting fake packets in their place. Since SyncTap is used for connecting nearby devices, the devices can easily give immediate connection feedback (e.g., by 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.

3.4 Establishing direct wireless connection

In the previous sections, I assumed that the devices already had network access and that the user performed a synchronous operation to establish a device association. For instance, we could assume that the wireless devices already have network access through a wireless access point. In addition to this application of SyncTap, by slightly modifying the synchronous operations, it also becomes possible to establish the wireless connection itself.

Wireless ad-hoc networks, such as the 802.11 b ad-hoc mode, create direct device-to-device wireless connections without using an access point. This mode is suitable for connecting devices where no wireless infrastructure is available. To establish such a connection, both devices must identify each other and agree on

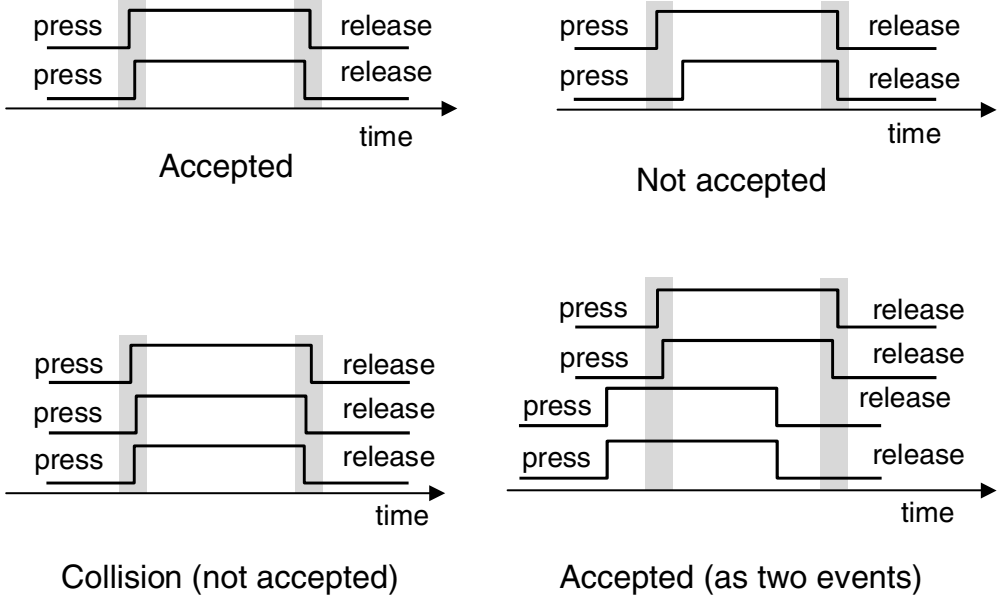

Fig. 3 Packets timing is used to distinguish SyncTap events from other collisions

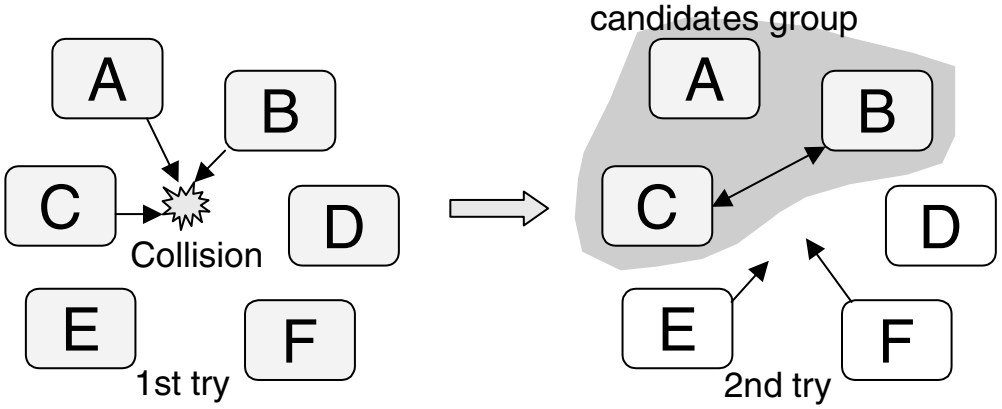

Fig. 4 Multiple overlapping SyncTap operations can be detected and handled as a "collision"

shared session information (such as an extended service set identifier (ESSID) for service identification and a wired equivalent privacy (WEP) key for encryption). This means that, when people try to connect two devices in ad-hoc mode, they typically need to exchange ESSID and WEP key information beforehand. This exchange typically involves a manual operation.

The synchronous operation approach can also be applied to this situation. We have implemented a variation of SyncTap for ad-hoc mode wireless communication. In this case, when a user wants to create a (ad-hoc mode) connection between two wireless devices, he first synchronously presses and releases the SyncTap buttons on both devices. Instead of transmitting multicast packets as described in the previous subsections, however, the devices transmit the synchronous information as link-layer wireless broadcast packets. Any wireless device within wireless transmission range can receive these broadcast packets. Then, the two connecting devices check the synchronicity of the received broadcast packet and the local SyncTap information, and then create an ad-hoc wireless connection between themselves[1].

3.5 Selection of SyncTap buttons

SyncTap assumes that devices have network connectivity and have at least one button (we refer to this as the "SyncTap" button) for operation. The button can be specially installed for this purpose, or it can be an existing button, such as a keyboard key on a PC. It can also be implemented as a GUI (on-screen) button. Our current prototype uses the "Escape" key as a SyncTap button for PCs. This key can also act as a normal keyboard key by using a timeout. That is, the SyncTap action is not invoked unless the press–release interval is more than a predefined threshold (e.g., 500 ms). The only drawback of this technique is that if a user performs an auto-repeat of the Escape key, an unnecessary SyncTap packet is transmitted upon releasing the key. This might lead to an unexpected device connection if another auto-repeat operation occurs simultaneously on another device. In this case, the user would have to cancel the connection.

[1]Normally, wireless broadcast packets are processed in the wireless device driver and it is not possible for applications to send and receive these packets. We modified the existing wireless device driver [11] to allow applications to send and receive broadcast packets without establishing a wireless connection.

4 Application examples

This section explains how the technique can be used in realistic contexts, with working example applications.

4.1 Instant connection among mobile computers and devices

When a user wants to print a document from her PDA to a nearby printer, she begins by pressing the SyncTap buttons on the PDA and the printer. Then, a printer icon appears on the PDA screen, and she can drag a document icon onto the printer icon (Fig. 6a). Note that the actual contents of documents might be contained in the PDA's flash memory, or it might only manage links (e.g., URLs) to documents. Our current implementation uses a web server corresponding to a printer to handle the actual printing task. The PDA knows the URL of the web server as a result of the SyncTap operation. Then, the user operation for printing invokes the uploading of a document file to be printed to the web server, which handles the actual printing. We also expect that a similar interaction can be applied to connect various types of wireless appliances, as in the case of connecting a digital camera (with wireless communication capability) to a printer or a computer.

4.2 Ad-hoc connection at a hotspot or in a meeting room

Suppose that a user is meeting other people at a public hotspot and would like to exchange a file with one of his colleagues. He can press the SyncTap buttons on both

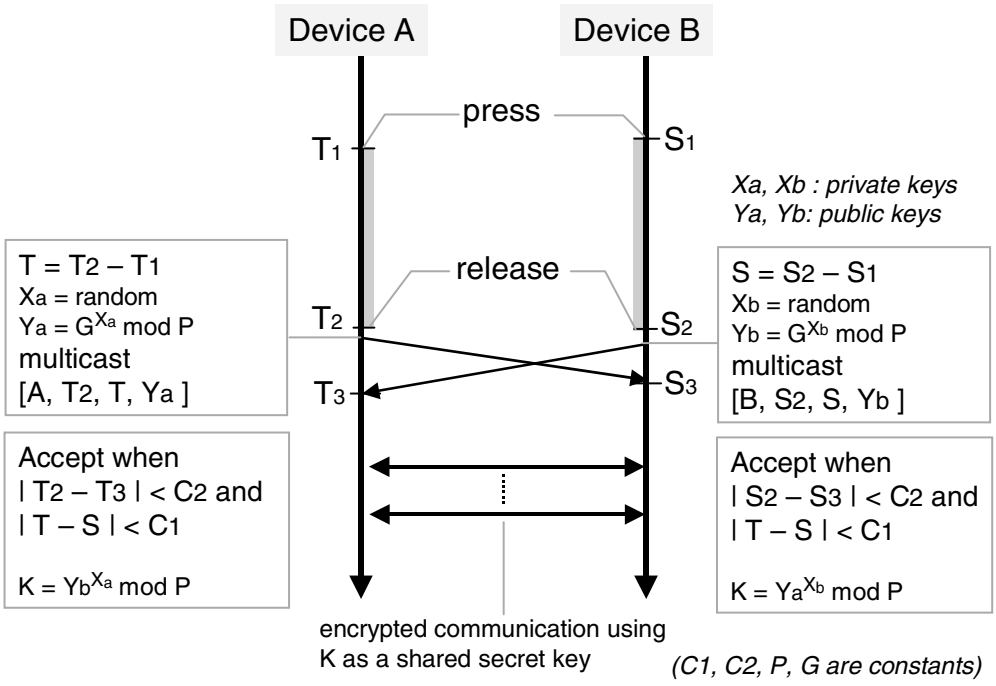

Fig. 5 Establishing secure communication: the Diffie–Hellman key exchange protocol is used to create a shared secret key, K

devices to create a connection (Fig. 6b). This operation creates an ad-hoc mode wireless connection between the two devices, which exchange Diffie–Hellman public keys and then begin secure communication.

4.3 Connecting devices that are not within a user's reach

When presenting a slide show in a meeting room, by using a wireless connection, a user can transfer the slide data from her notebook computer to the presentation computer. To do this, she simultaneously presses and releases the SyncTap buttons of the remote controller (i.e., the notebook computer) and the presentation PC (Fig. 7). The presentation computer receives an IR beam from the remote controller, and a network connection between the two is established. This example demonstrates how SyncTap can be used when two devices are not within arm's reach. A simple intermediate device, such as an IR remote controller, can be used as a "remote" SyncTap button. That is, the controller is only used to transmit press and release timings, making it unnecessary to transmit any complicated data, such as the address of the target device, by IR beaming.

4.4 Use of SyncTap for HyperCursor [3] communication setup

As part of our "Augmented Surfaces" system, we previously implemented a cursor migration system called HyperCursor [3]. Using HyperCursor, a user can control two computers with a single mouse and keyboard. When the cursor reaches the edge of one computer's screen, it automatically jumps to the second computer's screen. Keyboard inputs are then also delivered to the second computer. The user can also drag an object from one computer to the other across the boundaries of their screens.

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

A combination of HyperCursor and SyncTap can be used to address this problem (Fig. 8). For example, suppose that a user brings a tablet PC (without a keyboard) to his office desk and wants to operate it with his desktop PC's keyboard and mouse. To achieve this, he synchronously presses the PC's SyncTap button (e.g., the Escape key) and the tablet PC's screen[2]. Next, he manipulates the mouse to cross the edge of the desktop PC's 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 to develop another method in which a user can specify a physical layout by choosing one of multiple SyncTap buttons. For example, pressing the left Shift key on a device as a SyncTap button would tell the system that the other device is placed to the left.

[2]Some tablet PCs have physical "hot" buttons, which often act as an "Escape" key. In this case, these hot buttons can also be used as a SyncTap button.

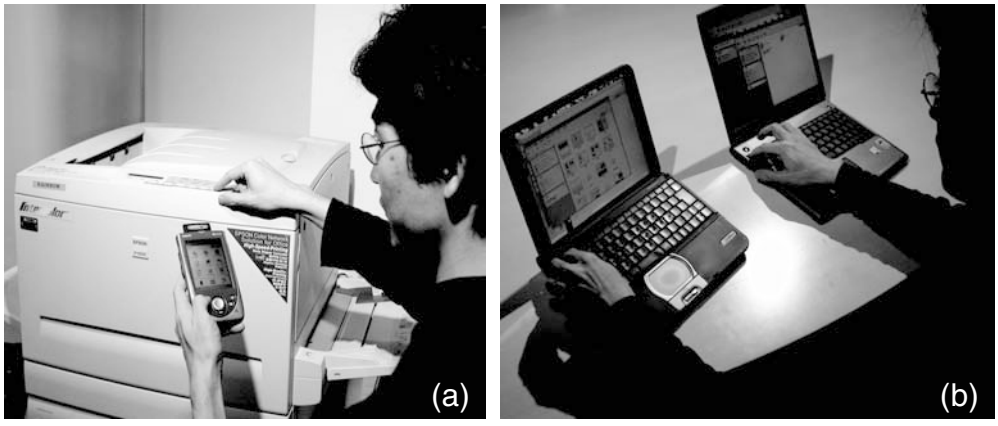

Fig. 6a, b SyncTap operations can be used in various settings. **a** Printing a document from a PDA to a nearby printer. **b** Connecting two PCs

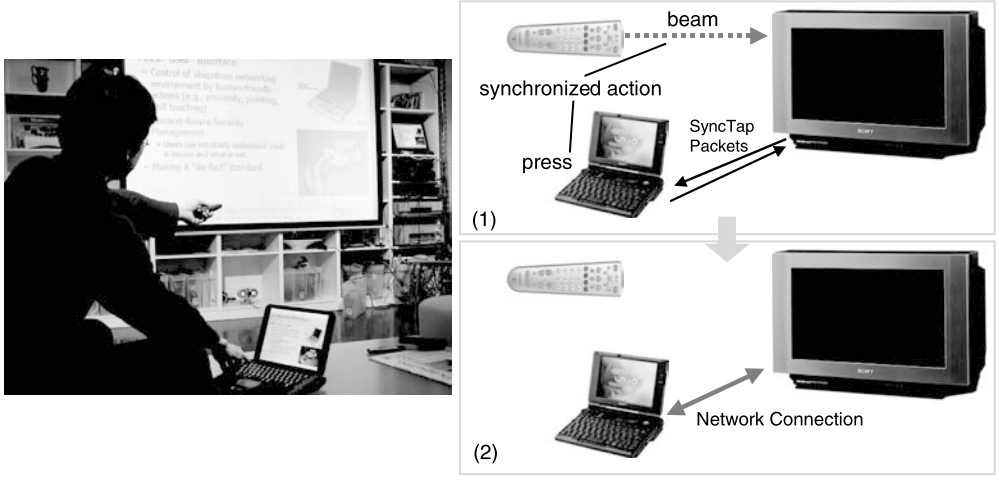

Fig. 7a, b Combination of infrared beaming and SyncTap. The IR remote controller is simply used as a "remote" SyncTap button. **1** The times of the beaming and the button press on the PC are synchronized. **2** Then, a network connection between the display device and the notebook PC is established. Note that no previous setup is required for this connection

5 Discussion

5.1 Human operation accuracy

SyncTap relies on humans to perform synchronous operations with both hands. To effectively distinguish SyncTap events from other coincidental operations, the selected threshold (i.e., the maximum allowable time lapse between SyncTap event pairs) is important. If this value is too small, some SyncTap actions will not be correctly recognized, while if it is too large, the probability of collisions will increase.

We actually measured the human operation accuracy by providing the SyncTap software to several users. Five subjects (all were male computer engineers) were asked to perform SyncTap operations 50 times by pressing the Escape keys on two computers. The average button press–release interval was 976 ms (with standard deviation of 172.22 ms), and the average time lapse between the button press–release intervals of SyncTap pairs was 35.37 ms (with standard deviation of 11.75 ms). Based

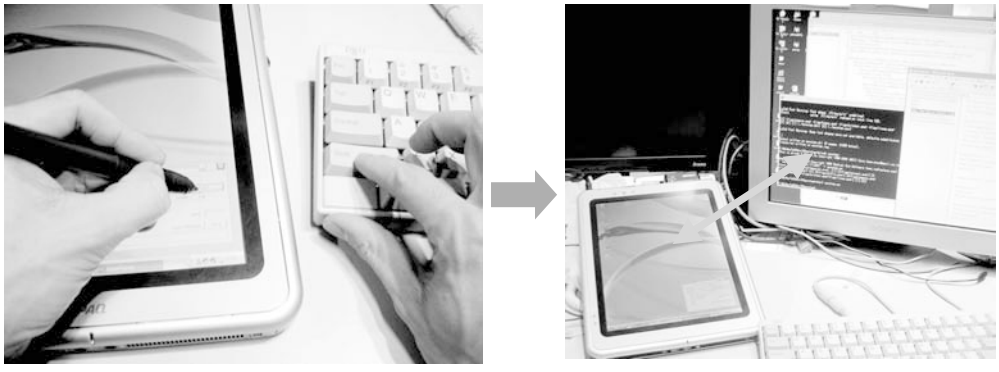

Fig. 8 Using SyncTap to set up a HyperCursor connection

on these measurements, we currently specify 100 ms as the threshold value (C1 in Sect. 2).

Figure 9 shows an actual timing chart for the establishment of SyncTap pairs. We also installed a HyperCursor system using SyncTap on 15 PCs in our laboratory, and five people used this system in their regular work. During a 1-month trial, no collisions were detected. This was 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 effect on usability.

Currently, the threshold values are based on our initial experience with SyncTap. Needless to say, larger threshold values would allow ''rougher'' user operation synchronicity and more tolerance for network delay, but they would also increase the probability of collisions, in which independent pairs of SyncTap operations occur. As described in the previous sections, collisions can be handled, and users can eventually establish a correct connection with a few extra SyncTap operations. To balance usability and collision avoidance, we might be able to introduce adaptive threshold values, for which the devices would observe the network traffic and the frequency of SyncTap packets.

5.2 Combination of device proximity and time synchronicity

In this paper, I have assumed that SyncTap packets are distributed to the network by using multicast packets. This means that devices connected on the same network segment are candidates for connection. Although this scheme works with a practical number of connected devices, we can also combine timing synchronicity with other device selection schemes, such as device proximity [9, 12], or a device's absolute locations [13]. Wireless devices for IEEE 802.11 wireless networking are normally capable of controlling the transmission power and measuring the signal strength of incoming wireless packets. Using these parameters, they can either restrict the transmission range by reducing the transmission power or ignore packets from distant nodes by limiting the signal strength threshold. Although these parameters are also affected by device orientation or wireless reflection, and even though it is difficult to measure distance accurately and reliably with only these parameters, it is still possible to define a ''loose'' proximity area. By combining this proximity definition and synchronous operation, the candidates for connection can be effectively limited to devices near the user, and the chances of collisions can thus be greatly reduced. This feature allows a more lenient synchronicity check. For instance, a user can establish a connection between devices by pressing the SyncTap button on the first device and then quickly (i.e., within a few seconds) pressing the SyncTap button on the second device. In this case, the user does not have to use both hands simultaneously for SyncTap operation.

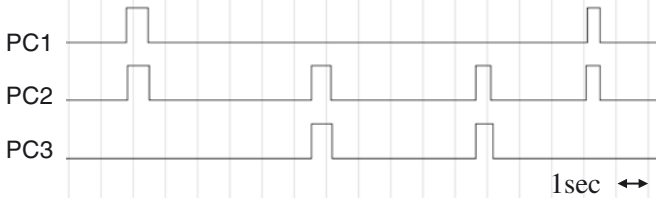

Fig. 9 Timing chart for establishing a SyncTap pair

5.3 Other synchronous user operation possibilities

Although this paper mainly treats synchronous user operations such as button pressing, mouse clicking, and IR beaming as methods of synchronous operation, many other user actions could potentially be used. One possible variation is shown in Fig. 10. This example shows the use of a button on one device to press a button on another. For example, if the tip of a cellular phone antenna were a button, it could be used to ''press'' another device's SyncTap button. This style might feel more natural than using both hands, as well as providing a metaphor for similar, real-world actions, such as connecting a plug to a socket or inserting a key into a keyhole.

In the presentation example in the previous section, 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 a device similar to the Pick-and-Drop pen [14]. For example, consider a wireless device with a SyncTap button, which we call the ''relay'' device. This device could be used to connect devices A and B even if they were not simultaneously within the user's reach. For example, the user might first connect the relay device to device A, then walk over to device B and

Fig. 10 A variation for SyncTap operations: a button on one device is used to ''press'' another device's SyncTap button

connect it to the relay device, thus, connecting devices A and B. While the Pick-and-Drop technique mainly handles one-shot data transmission, this approach with an intermediate device would establish a network connection that could be used continuously to transmit data.

5.4 User interfaces before and after connection

Although this paper mainly focuses on creating a network connection, the user operations after establishing the connection are equally important. For example, if more than one service connection type is available, the system must provide a user interface for choosing one of them. Holland et al. [15] proposed a method, called "Dynamic Combination" that effectively chooses the available operations based on the combination of selected devices. For example, by selecting two devices, such as a PDA and a printer, the number of possible operations would be greatly reduced. Thus, the user interface could be simplified by first connecting two devices, and then choosing a command. Similar techniques could be used with SyncTap.

Our current system supports both device association establishment, where devices already have network access, and ad-hoc mode wireless connection creation, where no network infrastructure is available. The current implementation requires users to specify which connection type is required. We are currently considering an improvement that would automatically sense the network accessibility and automatically choose an appropriate network connection style. This would hide the underlying network issues from users, enabling them to perform the same SyncTap operations in different situations.

6 Conclusions

This paper has proposed a notion of synchronous user operation and described the SyncTap method, a user interface technique for spontaneously establishing network connections between digital devices. This method can deal with multiple overlapping connection requests by detecting "collision" situations, and can also ensure secure network communication by exchanging public key information upon establishing a connection. Unlike previous systems requiring various sensors for device identification, the proposed method only assumes that both devices have human-controllable buttons, and it utilizes synchronous timing as an identification method. Our group believes that this method will find a wide range of applications in mobile computing environments in the near future.

Acknowledgements I thank Michimune Kohno and Yuji Ayatsuka for their thoughtful discussions with me on the idea of synchronous operations.

References

1. Rekimoto J, Ayatsuka Y, Kohno M, Oba H (2003) Proximal interactions: a direct manipulation technique for wireless networking. In: Proceedings of the 9th IFIP TC13 international conference on human-computer interaction (INTERACT 2003), Zurich, Switzerland, September 2003
2. Swindells C, Inkpen KM, Dill JC, Tory M (2003) That one there! Pointing to establish device identity. In: Proceedings of the 15th annual ACM symposium on user interface software and technology (UIST 2002), Paris, France. ACM Press, pp 151–160
3. Rekimoto J, Ayatsuka Y, Kohno M (2003) SyncTap: an interaction technique for mobile networking. In: Chittaro L (ed) Human-computer interaction with mobile devices and services (Mobile HCI 2003), Springer, Berlin Heidelberg New York, LNCS 2795, pp 104–115
4. Rekimoto J, Saitoh M (1999) Augmented surfaces: a spatially continuous workspace for hybrid computing environments. In: Proceedings of the ACM conference on human factors in computing systems (CHI '99), Pittsburgh, Pennsylvania, May 1999, pp 378–385
5. Want R, Fishkin KP, Gujar A, Harrison BL (1999) Bridging physical and virtual worlds with electronic tags. In: Proceedings of the ACM conference on human factors in computing systems (CHI '99), Pittsburgh, Pennsylvania, May 1999, pp 370–377
6. Tandler P, Prante T, Müller-Tomfelde C, Streitz N, Steinmetz R (2001) ConnecTables: dynamic coupling of displays for the flexible creation of shared workspaces. In: Proceedings of the 14th annual ACM symposium on user interface software and technology (UIST 2001), Orlando, Florida, November 2001
7. Zero Configuration Networking: http://www.zeroconf.org
8. Holmquist L, Mattern F, Schiele B, Alahuhta P, Beigl M, Gellersen H (2001) Smart-Its Friends: a technique for users to easily establish connections between smart artefacts. In: Ubicomp 2001, Atlanta, Georgia, September/October 2001. Springer, Berlin Heidelberg New York, pp 116–122
9. Hinckley K (2003) Synchronous gestures for multiple users and computers. In: Proceedings of the 16th annual ACM symposium on user interface software and technology (UIST 2003), Vancouver, Canada, November 2003
10. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE T Inform Theory 22:644–654
11. Jouni Malinen. Host ap driver for intersil prism2/2.5/3 and WPA suppicant: http://hostap.epitest.fi/
12. Schilit B, Adams N, Want R (1994) Context-aware computing applications. In: Proceedings of the IEEE workshop on mobile computing systems and applications, December 1994, pp 85–90
13. Bahl P, Padmanabhan V (2000) RADAR: an in-building RF-based user location and tracking system. In: Proceedings of IEEE INFOCOM 2000, Tel-Aviv, Israel, March 2000, pp 775–784
14. Rekimoto J (1997) Pick-and-Drop: a direct manipulation technique for multiple computer environments. In: Proceedings of the 10th annual ACM symposium on user interface software and technology (UIST '97), Banff, Alberta, Canada, October 1997, pp 31–39
15. Holland S, Oppenheim D (1999) Direct combination. In: Proceedings of the ACM conference on human factors in computing systems (CHI '99), Pittsburgh, Pennsylvania, May 1999, pp 262–269