tranSticks: Physically Manipulatable Virtual Connections

Yuji Ayatsuka
Interaction Laboratory,
Sony Computer Science Laboratories, Inc.
3–14–13 Higashi-Gotanda, Shinagawa-ku,
Tokyo, 141–0022, Japan
aya@csl.sony.co.jp

Jun Rekimoto
Interaction Laboratory,
Sony Computer Science Laboratories, Inc.
3–14–13 Higashi-Gotanda, Shinagawa-ku,
Tokyo, 141–0022, Japan
rekimoto@csl.sony.co.jp

ABSTRACT
A virtually connected medium called *tranStick* is described that functions both as a “virtual wire” and as a “memory card” containing a shared space. A user can connect two networked devices by simply placing one of a pair of tranSticks with the same identifier into each device. The tranSticks provide feedback indicating that the devices are connected; the connection to be closed or changed in the same way it would be if the devices were connected by a physical cable. A user can also access to a shared space on a network as if the space were in the tranStick. Since tranSticks contain long secret keys, the process of finding another tranStick with the same identifier can be encrypted. The tranStick approach differs from other approaches in that it provides feedback from the connection as well as serving as a medium for establishing a connection, and it enables disconnection and switchover to be done intuitively because the operations are reversible.

Author Keywords
tangible user interface, connection control

ACM Classification Keywords
H.5.2 User Interfaces (Interaction styles), I.3.6 Methodology and Techniques (Interaction techniques)

INTRODUCTION
Mark Weiser’s concept of ubiquitous computing envisions computers seamlessly integrated into our everyday environment[18]. In a ubiquitous computing environment, “invisible” computers interact with us, supporting our daily activities. While small computing devices have become common, they are neither invisible nor seamless. Moreover, we still have to directly or indirectly manipulate them. Although the number of computing devices around us is expanding, it will be a long time before they become “invisible.” In addition, even users in a ubiquitous computing environment may still find some visible directions and feedback helpful.

In today’s computing environment, users have to control the connections between devices, which can be cumbersome. To give a presentation, for example, a user has to search for the other end of the cable connected to the projector and connect it to his/her PC while being careful not to knock anything over. Even a networked device often forces us to find it in a list (that is often long with many similar names) of available devices discovered by a service discovery protocol. A user has to know the name of the device to be connected on the network, even if it is immediately in front of him/her. As the number of networked devices increases, giving them easily recognizable names will get harder and harder. In addition, this approach to finding a device works only for a devices with a display.

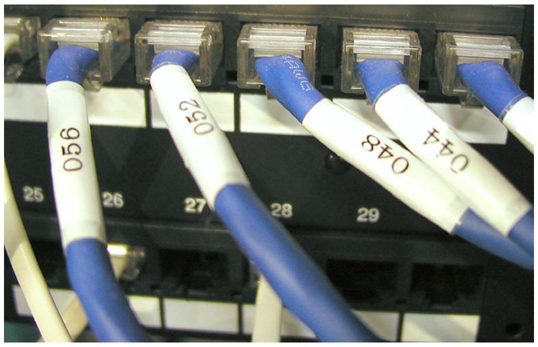

Figure 1. Cables Labeled by Numbers

There has been work on connection establishment using intuitive means[1, 9, 8, 15, 6, 5]. However, from a connection management viewpoint, a physical cable still has several advantages. It not only establishes a connection directly but also indicates the connection directly, so that a user can easily tell which devices are connected. Closing or changing a connection is also done by an intuitive physical action, the result of which is clearly evident. In addition, as in the case of a presentation, it inherently provides and represents an exclusive connection so that it can be used as a token.

Some of these advantages of a physical cable are also useful in a networked environment. One of the important features of a cable is that its ends are clearly distinguishable when they are labeled appropriately by letters, colors, and so on (see Figure 1). The actual physical route of the cable usually does not matter. We therefore use the idea of “labeled ends” to manage connections in a networked environment.

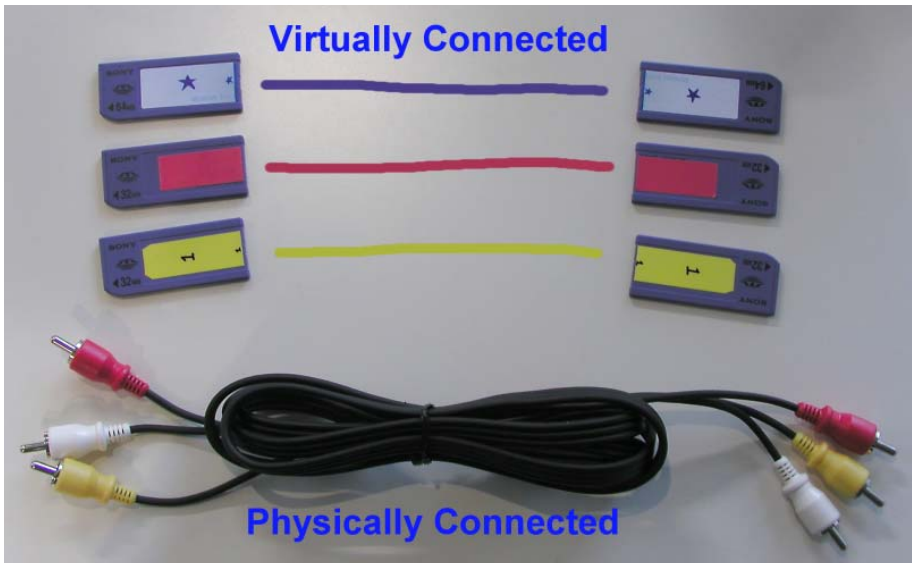

Figure 2. tranSticks and Physical Cables

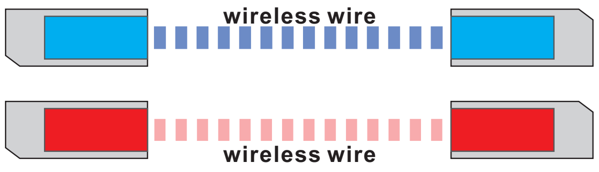

Figure 3. Function of tranSticks (1): connected by wireless wires

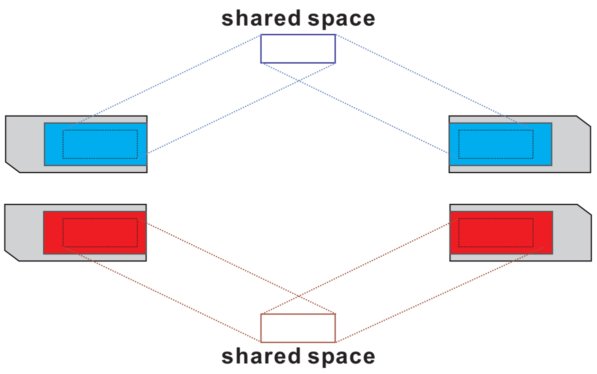

Figure 4. Function of tranSticks (2): containing shared space

We propose that such labeled ends be thought of as *media physically separated but virtually connected to each other*. Virtually connected media can not only replace a physical cable but also provide a connection between remote place and serve as virtual shared storage.

We implemented a virtually connected medium, which we call "*tranStick*," by using small memory cards. A tranStick contains an ID and a secret key that are used to identify a "buddy" tranStick. The key can also be used for encryption to establish secure connections. This paper describes the tranStick system and its applications.

TRANSTICK

A pair of tranSticks are virtually connected; that is, they correspond to the two ends of a physical cable. (Figure 2). From an implementation viewpoint, a tranStick is a medium that has an identifier enabling both people and computers to easily identify its buddy (i.e., another tranStick with the same identifier). For computers, the ID is composed of bits stored in the tranStick; for people, the ID is shown on a physical label affixed to the tranStick. When a tranStick is placed in a device, the label is better to be clearly visible for easy recognition.

A tranStick has two basic functions: one is to establish over some network a (possibly secure) connection between devices; and the other is to provide access over some network to a data space shared with buddies. To make use of these functions, a user need not know the names or addresses of the devices or shared space. The connection function enables a pair of tranSticks to be used as if there were an invisible wire between them. We call this style of use a "wireless wire" (Figure 3). The access function makes it seem as if the tranStick contains shared storage (Figure 4). The shared storage can be on a network storage server or in the tranStick itself. These two functions can be used simultaneously in some applications.

The tranSticks operations are reversible. A connection established by placing a pair of tranSticks in two devices is closed by removing either of them. A connection between devices A and B can be changed to one between A and C simply by moving the tranStick from B to C; moving it back to B re-establishes the connection between A and B. A shared space made available when a tranStick is placed in a device becomes unavailable when it is removed.

Each tranStick has an ID so that it can be found by its buddy. A secret key in each tranStick, generated at the initialization of a pair, provides a mean of authentication for the buddy (which has the same secret key), thereby keeping communication between devices secure. Since the user does not need to input the key or even see it, it can be much longer than a conventional key. In the traditional approach to selecting a connection target, i.e., by inputting an address or selecting an icon, the user may have to input a key on both devices to establish a secure connection.

Suitable Applications

tranSticks are not suitable for all connections among networked devices. They are suitable, for example, for

- a connection that is often intentionally changed by the user, rather than by an application program (e.g., between a PC and a projector used for presentation);
- a transient (secure) connection between devices in a transient network environment like a conference LAN or an ad hoc network; and
- a peer-to-peer connection between terminals of a remote family. Easy setup and visible/tangible feedback for access control is important, especially for novice users. tranSticks also provide a sense of security.

An example of an unsuitable application is a connection between a file server on an office LAN and a user terminal because it is almost always a fixed connection and the physical entity of the server is often hidden from users. Multiple connections from one (especially mobile) device is another example. Many tranSticks on one device may be cumbersome and confusing. Connections to be represented and manipulated by tranSticks may be the part of connections.

IMPLEMENTATION

A tranStick system includes the tranStick media, slots for inserting the media into devices, and software to search for and connect to the target device. Also needed is a tool to create a pair of tranSticks. In this section, we describe each part of the system we implemented.

Media

We selected the Memory Stick, a small memory card developed by Sony, as the tranStick medium. Using an existing memory card has advantages. One is that many PCs, especially notebook PCs, have a slot for a memory card, and memory card reader/writers connected via USB are widely available. Another is that it can easily store data including a long key. A third advantage is that their small size makes them easy for a person to handle.

For computers, the various kinds of memory cards are almost the same with regard to the storage function; for users, however, they differ. The reader/writer for a Memory Stick is often designed to show the label of the medium when it is in the slot, as shown in Figure 5. This feature is quite suitable for tranStick. From this point of view, a small USB storage device is another candidate for the tranStick medium.

The storage area in the memory card required for tranStick management is not large. Even if the tranStick contains a long key, the rest of the area can be used for data or caching when the tranStick provides access to virtual shared storage. In this sense, a tranStick is an expanded memory card.

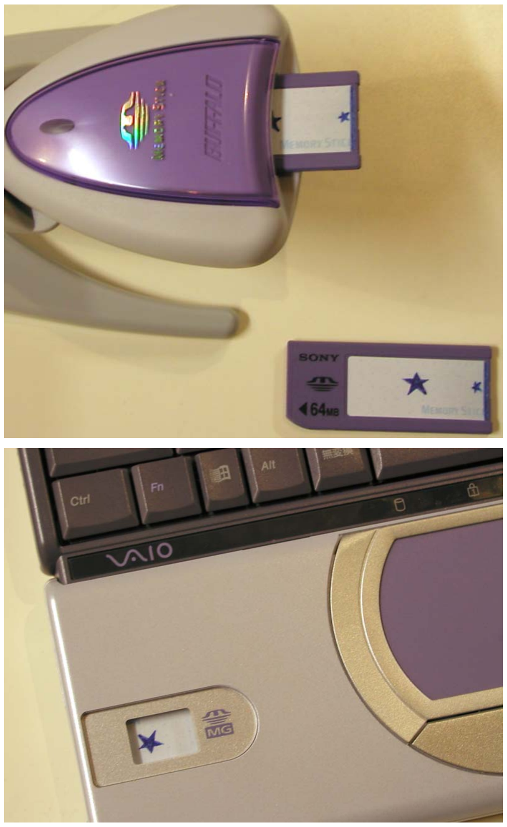

Figure 5. tranSticks in Slots

Initialization of tranSticks

We developed a tranStick initialization tool with two slots. A pair of memory cards are simultaneously placed in the slots, and when the user pushes the “register” button, they become buddy tranSticks. The tool generates a random 128-bit ID and a random 1024-1280 bit secret key (the exact length is determined randomly). A key is generated on a trusted device and written into a pair of tranSticks by memory card writers connected to the device so that the key need not to be transferred over a network. For higher security, the key should be protected using some technique, like those used for digital copyright management. Setting up a pair of tranSticks simultaneously avoids the accidental creation of other buddies of the tranSticks. Unfortunately, in the current implementation, the physical labels attached to each pair must be prepared and affixed manually. In a practical use, ready-made pairs of tranSticks will be convenient for users. Such ready-made pairs should have globally unique human-readable identifier. In the case, a part of its ID is represented by colors or simple marks for human to distinguish tranSticks at a glance, and the full ID may be printed as a number or string.

Some uses of tranSticks require the media to contain additional information. Such information is also written during

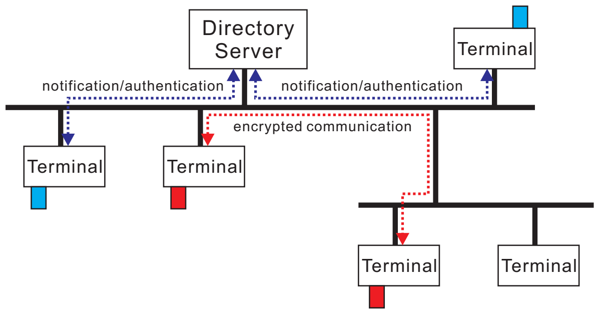

Figure 6. tranStick Using a Directory Server

the initialization. For example, a tranStick used to access shared storage server on the Internet (an example application will be described in the Applications section) needs the URL of the storage. The address of the directory server to use helps a tranStick find its buddy. Applications to handle tranSticks can also be stored in a tranStick because they have mega bytes of storage. This enables tranSticks to be easily used on devices in which tranStick applications have not yet been installed.

Searching for a Buddy

A device into which a tranStick to be used for a wireless wire has been placed has to look for the tranStick's buddy. A buddy can be searched for by asking the directory server or by issuing/listening broadcast or multicast messages. Here we describe using the directory server (Figure 6).

When device A detects a tranStick in its slot, it sends the ID of the tranStick and its own network address to a directory server (the address of which is stored in the tranStick). The server creates a random string as a challenge and sends it to device A. Device A calculates an MD5 digest for the string consisting of the received challenge followed by the secret key in the tranStick, then sends it to the server.

If device B sends the same ID to the server, the server creates another random string and sends it to device B. When the server receives the digest calculated by device B, the server sends it and the challenge string for device B to device A. Device A calculates a digest for the challenge and the secret key, then checks whether the calculated and received digests are identical. Device B similarly checks the digest calculated by device A. Both send their results to the server.

Only if both devices A and B accept each other's digest does the server send them each other's addresses. Each device then sets the other's address as the connection target (this is described in the next section) and, if needed, tells appropriate local applications on them to start a session.

Removing a tranStick from a device aborts the process, causing the device to return to its initial state. If a session is already open, the tranStick manager in the device instructs the application to disconnect it.

The secret keys in the tranSticks need not be sent over the network in this process. The server also need not know each secret keys. The process is mediated by the server, but the devices authenticate each other themselves. Broadcast or multicast messages can also be used to find buddy, instead of using the directory server. In this case, challenge strings are generated by each device, and devices receiving the string must check each other's digest directly.

Our directory server is written in Java, so it runs on any platform that has a Java environment. A client tranStick manager now runs on Windows. The protocol between them is an original one but can be embedded into HTTP requests and responses in order to be available over a firewall.

When a server is not available, both device A and B issue muticast message with the ID and different random strings. Each device picks up a message containing the ID that is identical to device's own ID, and a cluculated digest is sent back to the sender. A connection is established only if device A and B accept calculated digests each other.

Setting up a Connection Target for an Application

An application can include the tranStick manager function, and in that case it is easy to set up and change a connection target. Some applications allow themselves (and the establishment and closing of their connection) to be controlled externally by means of messages, and a tranStick manager can issue such messages. The connections of the other applications are managed by a proxy controlled by the tranStick manager.

A proxy runs on each tranStick-enabled device. An application's target address has to be set to that of the local host (127.0.0.1). The proxy checks the current target address set by the tranStick manager for each connection request from applications and then bridges communications between the applications and the current target. Our proxy, which can handle TCP connections, is written in Java.

The proxy and an application that supports tranStick management by itself provide encrypted communication using the keys stored in the tranSticks. We introduce an example in the next section.

APPLICATIONS

In this section we describe some prototype applications of tranSticks. Some are completely newly programmed; the others use existing applications without modification. Alternative implementations, for example, using a directory server and using broadcast messages, are also described.

tranSticks as Wireless Wires

Three applications that use tranSticks as wireless wires are a native application supporting tranSticks, an application controlled by proxies, and an externally controlled application. Each is managed by the tranStick management program called transConnect, which uses the directory server.

tranScreen

This application includes a presentation tool and a display server. The presentation tool sends the current display im-

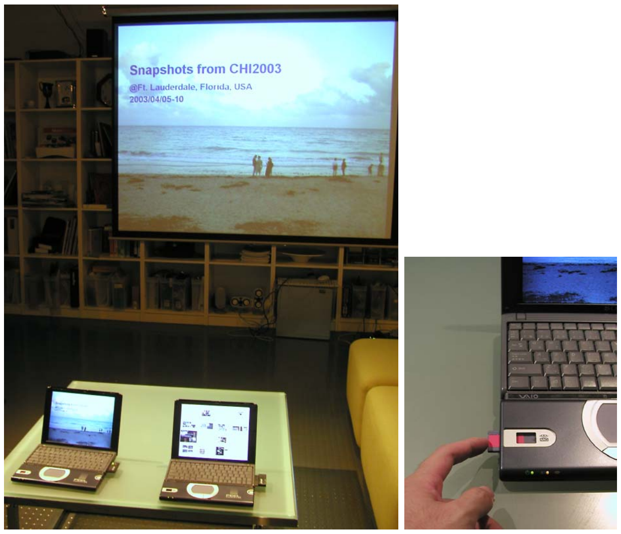

Figure 7. tranScreen: the left PC is connected to a projector via network

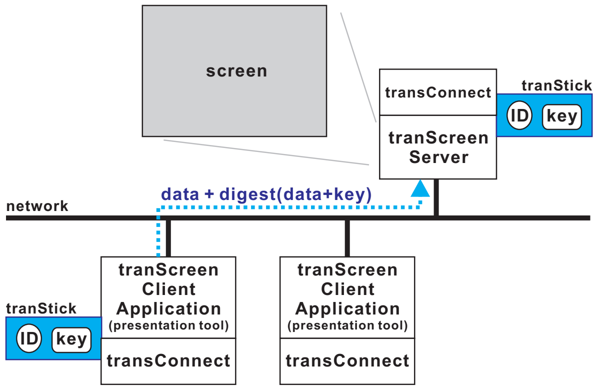

Figure 8. Architecture of tranScreen

age to another host as a JPEG image, and the display server shows JPEG images at the specified positions (this tool and server are both written in Java). Slides are projected on a screen not by using a physical cable to connect a notebook PC running a presentation tool to a projector but by placing buddy tranSticks in the PC and the projector (in the current implementation, actually a desktop PC physically connected to the projector). Figure 7 shows a scene of a presentation, he/she removes the tranStick from his/her PC of one speaker ends, the speaker takes (a "No Input" message appears on the screen) and passes it to the next speaker (needless to say, their PCs are connected to a network). This action is quite similar to what is done when physical cables are used, but the speakers do not have to worry about knocking anything over wit or about cables getting tangled. A pair of tranSticks could of course be placed in two PCs or in two projectors, but nothing would happen, the same as with physical cables.

The architecture of tranScreen is shown in Figure 8. The presentation tool sends images and a digest calculated from

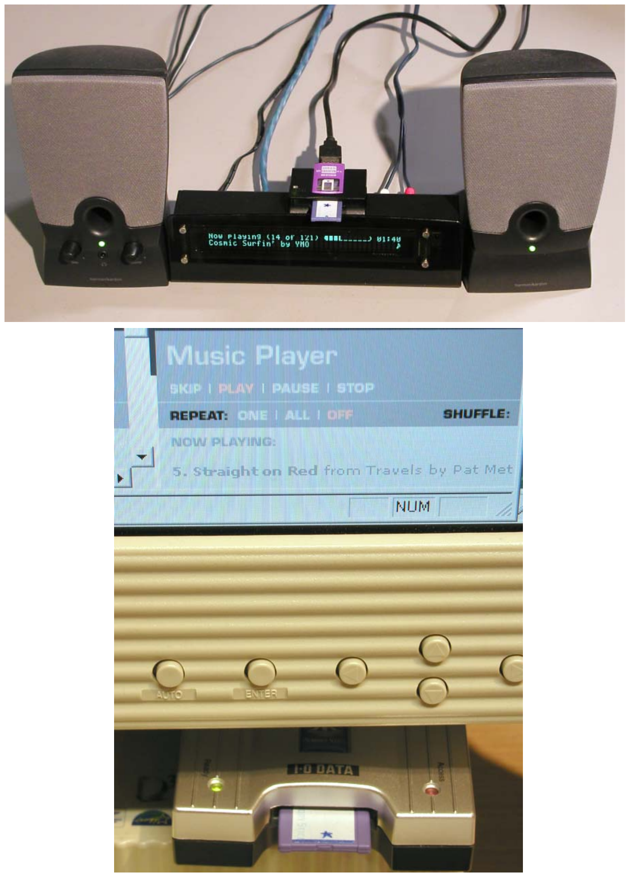

Figure 9. tranSound: Speakers and Server

the image data and the key[1] in the installed tranStick to the address set by transConnect. The display server shows only the received image data; the digest is verified against the key in the installed tranStick. Image data are not encrypted in the current version. We are going to extend this system to use other display transfer protocols like VNC[10].

tranSound

This application is made of music servers that provide MP3 data and music players that decode MP3 streams (Figure 9). We use is Slim Device's SLIMP3 server and client. Normaly a SLIMP3 client connects to a server directly, it connects to a client-side proxy in this application. The proxy connects to a server-side proxy that is denoted by a transConnect (Figure 10). The proxies mediate UDP packets from/to the SLIMP3 client. Communication between client-side and server-side proxies are encrypted by a shared key in a pair of tranSticks. Both SLIMP3 server and client are used unmodified.

A user listening to music can move to another room by transferring the tranStick from one speaker to another. A user can also change the music source by transferring the tranStick from, for example, a CD player to a radio.

[1]Such a digest is called as a Message Authentication Code (MAC).

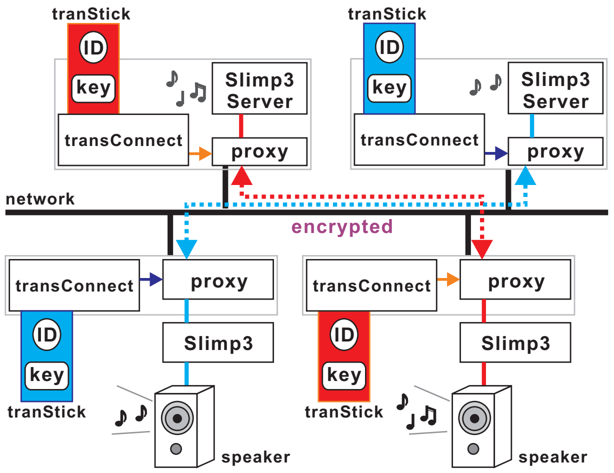

Figure 10. Architecture of tranSound

NetMeeting with tranSticks
NetMeeting is Microsoft's video conferencing application. An external application can make a NetMeeting call to a specified address, enabling to be controlled by transConnect (Figure 11). Users can start a video conferencing session, without typing or selecting a name or an address, by simply placing buddy tranSticks into their terminals.

Some experienced users may think that typing a name or selecting an address is easier than placing a tranStick and is available without handing the buddy tranStick to someone to talk. It can be true in many cases. However, tangibleness greatly helps novice users. In addition, it will also help experienced users when they want to change connections, for example, from a mobile terminal to a fixed terminal.

In tranScreen and tranSound, there is a server-client relation between devices to be connected, while in NetMeeting all conferencing devices have equal status. A connection between devices A and B can therefore be changed to one between A and C by transferring the tranStick from B to C. Then that connection can be changed to one between devices B and C by transferring the tranStick from A to B, and so on (Figure 12). This feature illustrates how a wireless wire implementation using tranSticks acts exactly like a physical cable. This simplicity will also help a user of multi-player games on mobile game terminals with an ad hoc network to select the others who play with him/her.

Access to Shared Storage
We developed two shared-storage applications: one uses a shared storage server on a network, and the other synchronizes contents in a pair of tranSticks by peer-to-peer communication.

tranSpace
tranSpace works on a device (a Windows PC) as a proxy for WebDAV servers (written in Visual C++). It refuses any connection unless there is a tranStick in the device requesting the connection. When it detects a tranStick, it invokes

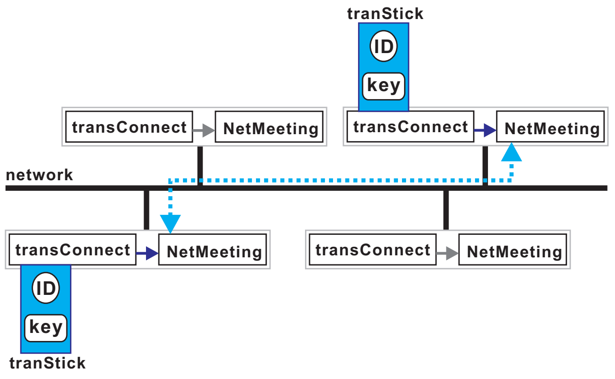

Figure 11. NetMeeting Controlled by tranSticks

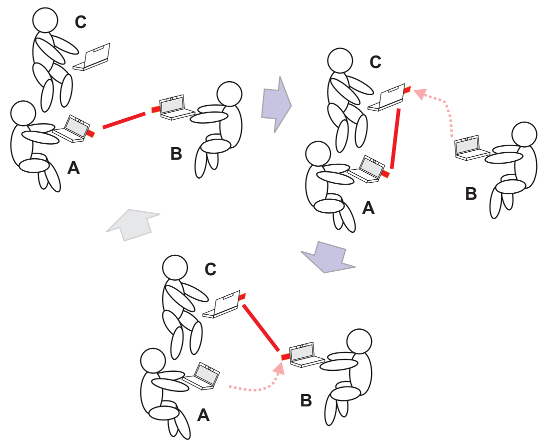

Figure 12. Changing Connection among Three Devices Using tranSticks

a web folder (a WebDAV client) targeted at the local proxy (i.e., http://localhost/) and starts to work as a proxy for the server of the URL stored in the tranStick. The storage area denoted by the URL is allocated when that tranStick is initialized. tranSpace encrypts and decrypts files in the storage space by using the secret key in the tranStick. Because tranSpace runs on a device, only encrypted files are transferred over the network and stored in the server. The server (we used an Apache 1.3 with mod_dav) need not to handle encryption at all.

With tranSpace, a user can access shared network storage in the same way a conventional memory card is accessed. Data can be easily and safely shared with another user who has a buddy tranStick. Passing a tranStick to share data is similar to passing a memory card containing data to share, but tranSticks have advantages. One is that users need not care about the free area in the media. Another is that a user can add data to share after passing a tranStick to another user. A third is that once a tranStick is passed, data can be shared bidirectionally.

One tranStick, not a pair, is also useful with tranSpace. In this case, the network storage is a private storage that can be accessed from anywhere by means of that tranStick. This

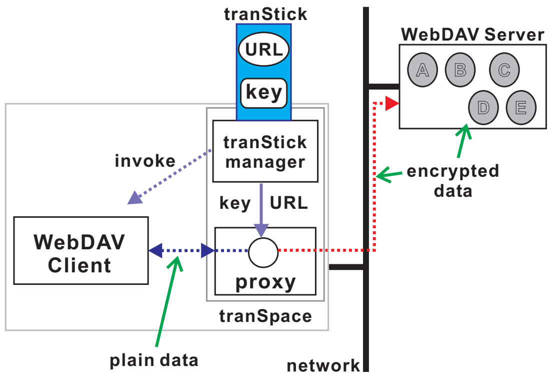

Figure 13. Architecture of tranSpace

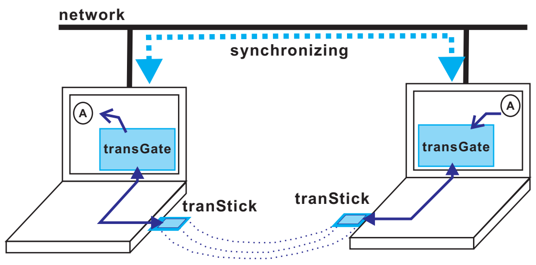

Figure 14. Concept of transGate

style of use of tranSpace can be considered an extension of mediaBlocks[16] or Informative Things[2].

transGate

While tranSpace provides access to a network storage server, transGate lets devices with a pair of tranSticks directly communicate with each other and synchronize the data in their tranSticks (Figure 14). A tranStick can be used as a normal memory card when its buddy is not found on the network. If the buddy appears on the network, transGate detect it by using multicast messages (without a server), then transGates on the devices start communicating with each other. After authentication, they exchange the files that one has but the other does not have. To the users, the pair of tranSticks appear to have the same storage area.

Figure 15 shows two screens displaying transGate browsers that are connected. A user can drag-and-drop files onto and from the browsers. When communication is active, a "connected to..." message appears in the browser window, as shown in Figure 16. File icons or thumbnails transferred from another device are faded in, and deleted items are faded out. The positions of the items are synchronized during the connection (in the current implementation, off-line movements are not reflected). The window of a transGate browser works as a gateway to another device. Note that the shared files are not stored in a server but in the tranSticks themselves. This results in secure file sharing even in an ad hoc network environment (one in which the network is not connected to the Internet), without managing user accounts or access permissions.

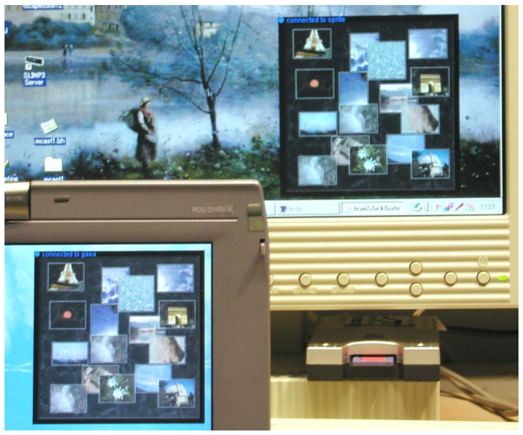

Figure 15. transGate: These two PCs access to the "same" shared space in tranSticks (not in a server)

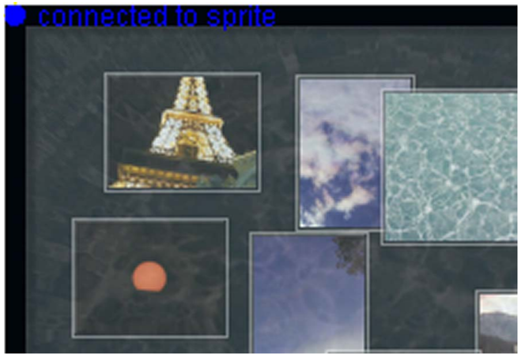

Figure 16. transGate Browser (part)

The transGate browser is written in Java. A search and authentication protocol is built on top of UDP/IP multicasting, and a file synchronization protocol is built on top of TCP/IP. Different files with the same name are given additional different names.

We plan to combine tranSpace and transGate into an application that, without a network connection, will enable a tranStick to be used as a normal memory card and, when a network is available, share with a buddy a larger data space than tat of the tranStick itself. A tranStick would thus work as a cache of a shared network storage area.

PRELIMINARY USER STUDIES

We have performed a preliminary user studies to ensure that a concept of tranStick is acceptable for users. The study is divided into two parts: the first one is for revealing first impressions on paired media without any information, and the second one is for investigating whether the functions of tranSticks can be recognized by first-time users.

First Impression of Paired Media

We showed three pairs of tranSticks to a subject who had not seen or heard about tranSticks before. 4 subjects (1 novice PC user and, 3 experienced users, two female and two male) participated in the experiment.

First, without informing them about the functions of tranStick, we asked subjects to imagine the meaning of the pairs indicated by colors and/or marks. The novice user could not imagine any meaning. The experienced users answered; "each color or mark represents a function," "each color or mark represents its content," "those may be a kind of key," and "each pair contains same data" (we allowed multiple answers). These answers indicates that first-time users do not notice a concept of "virtually connected" without an explanation, but recognize that a pair shares something in common.

Second, we explained that "a pair is virtually connected to each other," and asked the subjects to imagine its function again. All subjects answered that contents in a tranStick will be shared with its buddy (the function of shared space). Because a tranStick medium itself is a memory card, it is natural that tranSticks are recognized as an extension of them.

About Wireless Wires and Shared Storages

The next expriment is about the functions of tranSticks. Five subjects participated. Two of them had not seen or heard of tranSticks, and the other three had already known something about them. We explained the concept and the functions of tranSticks before the experiment.

First, while showing them two ends of a tranStick, we asked, "could these represent the ends of a virtual wire to you?" All the subjects understood the function and thought the idea was a reasonable one, but one subject who had not known tranSticks before said "they seem to be memory cards, so it feels strange, but I understand."

We also asked when tranSticks are preferable to having a list of connectable end points to chose from. The subjects answered: "when there are many devices," "it is hard to recognize what is connected in wireless LAN environments," "when I want to change or duplicate outputs," "if a tranStick is clearly assigned a function and destination (cf. output a PC screen to a projector)" (not all the subjects answered). Their expectations seem match our original intentions very closely.

Second, we asked about their thoughts on function of shared storage. The subjects could more easily understand this function than the wireless wire. We explained that each pair of tranSticks has a key and shared data are transferred with encryption by the key. We then asked whether he/she think it is a safe way to share data. All the subjects said "yes."

We also asked whether he/she prefer data transfer using tranSticks to transfer by email (as attachment files) or using a web/ftp server. Four of the five subjects agreed with these comments: "it is easy to understand," "it is comfortable to use physical representations," "it is easy and safe, without any complicated settings," and "for myself, any method is ok, but when I want to ask a novice user to send data, tranSticks will help a lot." The other one said "it depends on the situation."

From the results, we expect that concept and functions of tranSticks will be accepted by users. However, as a wireless wire, tranStick medium using a memory card can cause confusion to users unless appropriate information is provided.

DISCUSSION

tranStick vs. Other Objects with IDs

Using objects with IDs to make virtual data tangible is not a new idea. For example, mediaBlocks[16] provides blocks with IDs to which a user can bind some data. Informative Things[2] uses floppy disks containing IDs to retrieve information on a network. IconStickers[13] are printed icons with barcodes bound on virtual data. Applications introduced in [17] augment things with RFID tags. The pick-and-drop[7] technique enables a user to "pick up" data on the screen using a pen with an ID.

These approaches focus on a data object or a group of data objects, rather than connections between devices. The tranSticks approach is different – the focus is on paired media, which enables connections to be managed flexibly. A tranStick buddy on a local or ad hoc network can be found by broadcasting or multicasting message, so that the system can work without a server, whereas previous approaches need a server to retrieve data corresponding to an ID. When a directory server is available, a tranStick buddy that is located in a different network can be found over an internet.

Wireless Wire vs. Physical Cable

Not only tranSticks have the attractive properties of physical cables, as described above, they are also free from some of their limitations. First, a connection established using tranSticks can extend over any distance (even over the ocean), while physical cables are naturally restricted by their length. From this point of view, tranStick is also quite different from wireless communication media. Second, they never become tangled, one of the biggest annoyances with physical wires. These features enable free placement and movement of connected devices.

One disadvantage of tranSticks, compared with physical cables, is that a user might lose one of a pair of tranSticks. While the other end of a physical cable can easily be found, a tranStick can be hard to find if it falls into the space between cabinets or if someone forgets to remove the tranStick before carrying the device away.

A pair of tranSticks used as a wireless wire can easily be replaced by another pair, but losing a tranStick used for shared storage causes the same kind of data-leak problems that occur when a conventional memory card is lost.

Selecting an Application

As described above, there are many potential applications of tranSticks. If a tranStick-enabled device has a simple function, an application to be invoked might be selected automatically. For example, a speaker could automatically output sound from a source device to which the speaker is connected by tranSticks. On the other hand, it is hard to select one application when, for example, two PCs are connected.

A transConnect has a list of applications available on the device, indicating the priority of each application. When a connection target is found, the list is compared to the target's list, and a matched one is selected. If there are two or more matched ones, the one with the highest priority is selected.

There are other ways to select an available application. One is to have the device show a list of applications to the user and wait for the user to select one. An easier way is to have a tranStick manager that does not handle connection establishment but rather handles only the target address. A connection is established when the user issues an "open connection" command to an application and selects "tranStick" as the target. A third way is to provide application-specific tranSticks. A tranStick for NetMeeting, for example, is used only for NetMeeting. A tranStick for two or three specified applications is also possible. To keep users from becoming confused, tranSticks for wireless wires should be clearly distinguished from tranSticks for virtual shared storage.

Three or More Buddies

So far we have talked over about using a pair of tranSticks, and the current implementation of a directory server and several applications handle only pairs. However, three or more tranSticks containing the same ID and key are also possible. Some applications could easily be extended to use a multi-device connection. tranScreen, for example, can simultaneously have two screens for one client. On the other hand, it is hard to determine an action for the case of two clients for one screen.

Instead of one-to-many connections, serial connections of three or more devices can be used in some cases. For example, a user who puts a set of three tranSticks in a music player, a sound effector, and a pair of speakers obviously wants to connect the player to the effector and the effector to the speakers. Identifying such a relationship is problematic. In addition, we have to consider feedback that represents a connection order among multiple devices.

A tranStick for each member of some small group makes it easy to share data (ex. photographs) among the members of the group. In this case, a tranStick is a kind of tokens, which contain address and permission to access.

Comparison to Our Other Approaches

While investigating intuitive ways to manage connections between devices by physical actions under real-world conditions, we developed interaction techniques other than the ones that use tranSticks and have been described here. Gaze-Link[1], which uses a camera installed on a device, enables a user to connect the device to a target on which is attached a visual code that is identified by using the camera. FEEL[9] is an interaction technique that establishes connections by bringing devices to be connected physically closer (detected by an RF tag system), or aiming one device at the other device (using infrared communication). SyncTap[8] connects devices when a user simultaneously pushes the same buttons on each devices.

A significant difference between tranSticks and these other approaches is that a tranStick provides feedback from the connection as well as serving as a medium for establishing a connection. tranSticks also enable disconnection and switch over to be done intuitively because tranStick operations are reversible.

Gaze-Link provides connection feedback as photographs of targets, and the user can easily change connections by clicking on a photograph. They are shown only on the source device, however; at the target device, there is no way to know where the connection originated. In addition, it is hard to change from one source device to another. Gaze-Link also requires that at least one terminal be portable, so that it is not suitable for managing a connection between fixed devices.

The operations of tranSticks are more indirect than the FEEL technique, in which devices to be connected are brought close to each other. However an obvious advantage of using tranSticks is that they can be used to connect devices that are hard to bring together, or are separated by long distance. These techniques thus complement each other.

RELATED WORK

There has been other work on connection management using intuitive means, rather than using names or icons on a display, as mentioned. gesturePen[15] uses infrared communication to detect a target to connect. Iwasaki and his colleagues introduced a technique, called Touch-and-Connect[6] for establishing connections by physical actions. Each Touch-and-Connect device has two buttons (1 and 2). A user who wants to connect devices A and B pushes button 1 on device A and button 2 on device B (or button 1 of device B and button 2 on device A) within a short interval. These techniques support only connection establishment; they do not support feedback. They also support only connection between devices in a room, or a small space. On the other hand, tranSticks can connect devices to each other over any distance, and provide feedback.

Synchronous Gestures[5] uses "bumping" to connect two tablet PCs to each other. When the PCs are detached, the connection is broken. This technique requires that at least one device be portable, and remote devices cannot be connected. Moreover, the positions of the devices are quite restricted. tranSticks can be used for connecting two remote and fixed devices to each other.

Putting a "same label" on devices to be connected is a traditional technique. For example, some wireless mice can select an ID (made of a few bits) so as to avoid crosstalk or conflicts. The IEEE802.11 standard uses an ESSID, a kind of software label, to distinguish an access point to be connected. These labels, however, cannot be moved or manipulated physically.

The concept of virtually connected devices appears in research on computer-aided and network-aided remote communication. InTouch[3], RobotoPHONE[11], Meeting Pot[12], and Peek-A-Drawer[14] are examples. Each of them is an independent application. tranSticks, on the other hand, are not virtually connected devices but virtually connected media, so that they can provide infrastructure for those applications.

Giving IDs to physical objects to make virtual data tangible was mentioned above. There are many systems and techniques for doing this[16, 2, 13, 17, 7], but they focus on handling a data object or a group of data objects, rather than on managing connections between devices or sharing data.

Using music CDs to control access to a privileged web site[4] is an application similar to providing virtual shared storage with tranSticks. In this service, a commercial music CD works as a ticket to contents provided by a service provider, rather than as data shared between users.

CONCLUSION

We have described a new type of medium, called “tranStick,” that is labeled to make it easily identifiable by both people and computers and that is connected virtually. One function of tranStick is providing virtual cables (wireless wires). A pair of tranSticks enables a user to manage (establish, close, change) a connection between devices intuitively and to receive connection feedback. Another function is providing access to virtual shared storage. A tranStick contains a long secret key shared with a buddy; it is used to ensure the security of the process of searching for the buddy and of the communication between them. We plan to investigate the usability of tranSticks in daily applications.

REFERENCES

1. Yuji Ayatsuka, Nobuyuki Matsushita, and Jun Rekimoto. Gaze-link: A new metaphor of real-world oriented user interface. *IPSJ Journal*, 42(6):1330–1337, June 2001.
2. Rob Barrett and Paul P. Maglio. Informative Things: How to attach information to the real world. In *UIST ’98*, pages 81–88, November 1998.
3. Scott Brave, Hiroshi Ishii, and Andrew Dahley. Tangible interfaces for remote collaboration and communication. In *Computer Supported Cooperative Work (CSCW’98)*, pages 169–178, 1998.
4. Sony Music Entertainment. ConnecteD. *http://www.sonymusic.com/thelab/ConnecteD*.
5. Ken Hinckley. Synchronous gestures for multiple persons and computers. In *UIST 2003*, pages 149–158, November 2003.
6. Yohei Iwasaki, Nobuo Kawaguchi, and Yasuyoshi Inagaki. Touch-and-Connect: A connection request framework for ad-hoc networks and the pervasive computing environment. In *First IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 20–29. IEEE, Mar 2003.
7. Jun Rekimoto. Pick-and-drop: A direct manipulation technique for multiple computer environments. In *UIST ’97*, pages 31–39, October 1997.
8. Jun Rekimoto, Yuji Ayatsuka, and Michimune Kohno. SyncTap: An interaction technique for mobile networking. In *Mobile HCI 2003*, pages 104–115, September 2003.
9. Jun Rekimoto, Yuji Ayatsuka, Michimune Kohno, and Haruo Oba. Proximal interactions: A direct manipulation technique for wireless networking. In *Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003)*, pages 511–518. IFIP, September 2003.
10. Tristan Richardson, Questin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.
11. Dairoku Sekiguchi, Masahiko Inami, and Susumu Tachi. RobotPHONE: Rui for interpersonal communication. In *Conference Extended Abstracts on Human Factors in Computer Systems (ACM CHI 2001)*, pages 277–278. ACM, April 2001.
12. Itiro Siio and Noyuri Mima. Meeting Pot: Coffee aroma transmitter (poster/demonstration). In *ACM UbiComp 2001*. ACM, ACM, Sep 2001.
13. Itiro Siio and Yoshiaki Mima. IconStickers: Converting computer icons into real paper icons. In *Human-Computer Interaction, Ergonomics and User Interfaces, Volume 1 (HCI International ’99)*, pages 271–275. Lawrence Erlbaum Associates, August 1999.
14. Itiro Siio, James Rawan, and Elizabeth Mynatt. Peek-A-Drawer: communication by furniture. In *Conference Extended Abstracts on Human Factors in Computer Systems (ACM CHI 2002)*, pages 582–583. ACM, April 2002.
15. Colin Swindells, Kori M. Inkpen, John C. Dill, and Melanie Tory. That one there! pointing to establish device identity. In *UIST 2002*, pages 151–160, October 2002.
16. Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. mediaBlocks: Physical Containers,Transports, and Controls for Online Media. In *SIGGRAPH ’98 Proceedings*, pages 379–386, 1998.
17. Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging Physical and Virtual World with Electronic Tags. In *Proceedings of CHI ’99 Conference on Human Factors in Computing Systems*, pages 370–377. ACM, May 1999.
18. Mark Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1991.