Clicky: User-centric Input for Active Spaces

Christopher R. Andrews, Geetanjali Sampemane, Andrew Weiler, and Roy Campbell
Department of Computer Science, University of Illinois at Urbana-Champaign
{crandrws,geta,aweiler,rhc}@cs.uiuc.edu

Abstract. We present a user-centric input system that is useful for a variety of applications in ubiquitous computing environments. Users in these environments currently switch between different keyboards and mice, or other devices, to interact with all the devices present. Allowing users to select a personal input device for the space and use it to interact with all the devices in the space is a more natural approach. Associating a user with an input device also serves to distinguish between the different users in a space, and creates the notion of a user session that is useful for applications like authenticating user actions on shared public displays or logging a particular user’s activities.

Clicky is our prototype implementation of this idea. It allows a user with a single mouse and keyboard to control multiple computers and displays running standard operating systems in a smart-space environment. It is designed to be a light-weight middleware tool that does not require modifications to the base operating system. Any input device in the room, such as the keyboard and mouse on a personal laptop or handheld computer, can be used as a user’s input device to interact with the space. Clicky allows users to configure a set of displays in the space into a large “virtual display” and use a single mouse pointer to move across it. It also allows multiple users with their own input devices to share a public display. In this paper, we describe the motivation for a user-centric input system, and describe the current status of the implementation.

1 Introduction

Ubiquitous computing environments typically contain a plethora of networked computing devices. Users have to interact with a variety of these devices, and the fact that each of these typically requires their own input device (keyboard, mouse, remote control) makes it that much harder. Computers with large display devices, as well as numerous smaller devices such as hand-held computers or cellphones, location systems such as Active Badges [15] or Ubisense [16] tags are all common. User applications typically span a number of these devices, and users have often to interact with many of these devices. While networked filesystems provide a uniform interface to data, interacting with the displays is still clumsy, requiring users to switch keyboards and mice as they move across displays. This is largely because the displays are connected to computers running
commodity operating systems which expect each display to be associated with its own keyboard and mouse.

For example, our research environment (shown in Figure 1) consists of an Active Space [12] that contains a number of computers running commodity operating systems such as Microsoft Windows and Linux connected to large wall-mounted plasma displays with touchscreen capabilities. The software infrastructure consists of middleware services in the form of a “meta Operating System” that we call Gaia [12]. The objective of Gaia is to provide a uniform interface to all the hardware and software resources within a physical space, much like an operating system at a desktop computer provides a uniform interface to the various input/output devices. Gaia provides a framework that allows application components to be moved between different computers (and displays) in this environment, but it is difficult to mask the fact that they are separate machines when the user has to switch input devices to interact with them.

Another problem in such environments is to identify users and authorize their actions. When multiple users are within a space, sharing access to a common keyboard and mouse when they want to share a display, it is difficult for the system to distinguish between users without cumbersome authentication mechanisms each time the keyboard/mouse change hands.

However, users often already possess a device that identifies them. This may be a laptop or handheld computer, which is typically a personal device, and can potentially be used to interact with the various devices in the space. Ubisense tags are issued to users to track their location within our Active Space and provide location-based services. These tags also have programmable buttons which can provide the functionality of a mouse. Thus, allowing people in the room to use these tags as point-and-click devices for the various displays becomes an attractive idea.

Clicky was developed with the idea of providing a user-centric input mechanism for an Active Space. A user should be able to use any one of a set of input devices, possibly of varying capabilities, to interact with the variety of devices in

![Fig. 1. Users in an Active Space](image)
the environment. Middleware services can translate the “native” input of these devices to a form understood by the target device, much like a Universal Remote Control. An input device associated with a user can be used to identify actions as coming from that user. This can be useful for access control. A user-based input device also provides a point to log user activity in the space, for debugging purposes or as input for learning algorithms. The mouse can also be used to identify the current focus of activity, which helps applications that record such space activity to point the camera at the right displays.

Our current implementation of Clicky allows a person to use any mouse in the room, including a personal laptop, as his “space mouse”. It is being extended to allow the use of hand-held computers and Ubisense tags as input devices as well. Each user in our Active Space now needs only one mouse for their interaction with all the displays in the room. Keyboard input follows mouse focus. This changes the one-mouse-per-machine paradigm to a one-mouse-per-user paradigm. The user effectively has a large “virtual display” comprising the set of displays in the space—when the mouse moves off the edge of a display, it enters the “next” display, based on the space configuration.

Multiple users in the space may each use their own mouse. Each user’s Clicky “session” is independent of others—two or more users can each point their mouse at a common display at the same time and they will effectively “timeshare” the mouse. Clicky will transmit each of their mouse/keyboard events to the windows on which they focus. If they focus on different windows in the display, their activities will not interfere with each other; e.g., they can each type into a separate editor window without conflict. Users can also focus on a common application, such as a collaborative debugging session.

One of the design contraints for Clicky was to keep it lightweight, as we would like to simplify the process for users to be able to use it in any multi-display environment. Thus it was developed as a middleware tool that does not require any OS-level modifications or require much from the environment.

The main contributions of our work are:

- A user-centric approach to input that allows users in an Active Space to use a single input device to interact with all the devices around.
- Applications for ubiquitous computing environments based on the availability of a user-specific input device.
- Implementation of a simple, standalone middleware tool that allows a user to interact with multiple displays and control multiple computers using a single mouse and keyboard, without requiring modifications to the operating system.
- The ability for multiple users, each with their own input device, to share a common display such as a large display wall.
- Support for a variety of heterogeneous input devices, of varying capabilities.

The rest of this paper is organized as follows: Section 2 describes necessary background and related work. We then describe applications for a user-centric input device in Section 3, and use this to characterize the requirements for Clicky.
Section 5 describes the design and status of the current implementation. Finally, we conclude in Section 6.

2 Background and Related Work

In this section, we give a brief overview of the Active Space environment for which we developed Clicky and then review some related research in the area. While Clicky does not depend on the Active Space for any functionality, many of the design decisions are driven by applications for such an environment.

2.1 Gaia

The Active Spaces project [12] at the University of Illinois aims to develop software infrastructure to enhance user interaction with a device-rich physical and computational environment. Computers in the environment run standard operating systems such as Microsoft Windows, and CORBA middleware services provide operating system functionality such as naming, location and a uniform interface to data. An application framework [11] provides the ability to compose applications across the variety of devices available in the space and to move application components around during the application lifetime. This middleware meta-operating system is known as Gaia.

Our prototype Gaia system runs in a room with large touchscreen plasma displays around the walls. As mentioned earlier, these displays are connected as monitors to PCs which each expect a distinct keyboard and mouse for user interaction. In addition to these computer displays, there are a variety of devices in the space—wireless handheld and laptop computers, badges of various kinds to identify users or provide location-tracking for users and devices. To interact with applications running on a display, users have to either walk over to the display and use the touchscreen, or find the keyboard and mouse belonging to the computer connected to this display. Neither of these methods are particularly user-friendly, and this tends to inhibit users from using multiple displays. A user-centric paradigm of one-input-device-per-user rather than one-input-per-computer would be more natural for such environments.

If users could interact with applications using the devices they already hold, their interaction with the space would be greatly simplified. This device can be treated as a personalized “space controller”, and can be used to perform a variety of functions—clicking on menu options on any of the displays in the space, controlling devices such as lights and thermostats, and serving as a user “point-of-presence” in the space. Interesting applications for user identification and authentication also present themselves—if the user identifies himself on starting to use the input device, all further input from that device can be treated as originating from that user. This can help with the problem of providing a secure and convenient channel for users to interact with the space.

Our first attempt at providing this functionality was to develop a Gaia service for input redirection. While this was useful, we found that there was great
demand for such a service during the bootstrap of Gaia (when Gaia services are not yet available), especially during development and debugging of the infrastructure services. This led to a redesign of Clicky as a standalone middleware tool, independent of Gaia. An advantage of this approach is that Clicky can be used in other device-rich environments as well as in our Active Space.

2.2 Related Work

Related work has been in two areas: tools for “smart space” environments that allow users to control multiple devices with single pointers and collaborative environments that allow multiple users to interact with a common application. Clicky addresses both these problems—it allows a user to use a single set of input devices to interact with all the devices in an Active Space, and it allows multiple users to collaborate by sharing a display.

XWand [17] is a specialized hardware device with associated software to control multiple connected devices in an intelligent environment by pointing to them and gesturing. In our work, we try to use input devices that we already have, rather than develop customized hardware. However, we envision that Clicky will eventually support a variety of input devices, including the XWand and other like devices.

The PointRight [6] system from Stanford allows a single input device to interact with multiple displays in an Interactive Workspace [5]. In addition to input redirection, the objective of Clicky is to associate input with a user, thus creating the notion of a user “session” in the Active Space. A user session is helpful for a variety of applications, but difficult to track in multi-user, multi-device environments without a personal input device.

Integrating this input mechanism with location-sensitive devices such as Ubisense [16] tags provides some additional context-sensitivity—based on the location of the user, the input device can be bound to the most likely available device for interaction. For example, a user pointing the device at the lamp and clicking may indicate that he wants to toggle the light switch, but pointing the same device at a display may indicate a wish to use it as a mouse. Work is currently in progress on developing location-based services for Gaia and we expect to integrate Clicky with these.

Single Display Groupware [13] refers to mechanisms that allow co-present users to share a common display; the SDG toolkit [14] provides a library to aid the development of applications that can have multiple mice controlling a single application, but requires the application to be built using this library. The Pebbles [7] project allows the use of PDAs to control a single shared desktop. However, multiple simultaneous input requires application support. Clicky, however, can be completely transparent to applications and allow a shared focus over and input to any applications on the display, though it is possible to develop Clicky-aware applications.

There has also been work on devices for controlling ubiquitous computing environments using a pointer. The NCSA CAVE [2] environment allows multiple
users with their own “wands” to interact with a virtual 3D space. Xwand\(^1\) [8] is a program that uses the Diverse Tookkit [3] to allow the use of an input device such as a wand to manipulate all XWindow applications in an immersive environment. The Pick-and-Drop [9] system allows the use of a pen to “pick up” and move information between displays, which may be separate machines.

The “mighty mouse” [1] is a collaborative tool that uses the VNC [10] protocol to allow multiple users to interact with a set of applications on a variety of machines, but it preserves a one-to-one relationship between the controlled and controlling machines, so only one mouse can control a display at any time. Similarly, x2vnc [4] uses the VNC protocol to allow a machine running X windows to control another machine running the VNC server.

Clicky’s ability to merge multiple displays into a “virtual display” is similar to the functionality provided by various X [19] window managers [18] that support virtual desktops. VNC [10] allows you to view an entire remote desktop on a different machine. Unlike most of the virtual desktops, all the Clicky desktops are visible at once, since they are separate physical displays. Only the focus moves to the currently active desktop. X servers can be run in the multi-headed mode, where multiple display monitors are connected to a single machine, and the system mouse can move across the multiple monitors. In this case, all displays are visible at once; however, all these displays are connected to a single machine. Clicky allows a single input device to control a set of displays connected to separate, independent machines.

We now proceed to describe the applications of Clicky in our Active Space.

3 Applications for a user-centric input device

In this section, we describe applications that a user-centric input system would enable in an Active Space. We use these applications to guide our design for Clicky.

Users in Active Spaces interact with a variety of devices. This creates two problems: for the user, switching between input devices to interact with various devices in the space is cumbersome; and for the system, it is difficult to associate input as coming from a particular user. Providing a user-centric input mechanism can help with both these. We now describe how some applications in the Active Space would benefit.

3.1 User sessions

On conventional distributed systems, a user interacts with a machine (either remote or local) by logging in. The operating system creates a login session for the user, and all input provided to this session is treated as coming from this user. This allows the operating system to use the appropriate user credentials to authorize actions. In an Active Space, the equivalent notion of a user session

\(^1\) Unrelated to the Xwand in [17]
is harder to create, because users may interact with different devices by walking around to them and typing into the variety of keyboards and mice around the space. In single-user spaces, this is not a big problem, since it can be assumed that all input comes from the sole user, but especially when there are multiple concurrent users, separating input from different users is a problem. Users may wear identifying badges, and tracking cameras or other location-tracking systems can identify where they are in the space, but these are not currently precise enough to reliably identify which user is using a particular tablet or touchscreen, especially when users or input devices are close to each other. Having a user authenticate himself each time he picks up a new device is not practical.

One approach to solving this problem is to provide users with authorized input devices that they can use to interact with the space. A user can either use a personal input device (such as a hand-held or laptop computer) or be issued an input device (such as a wireless keyboard/mouse in the room) after being authenticated by the space. If the keyboard/mouse has a fingerprint reader or other authentication mechanism attached, a user who picks one up can use it to authenticate himself to the space. In either of these schemes, the space now associates a particular input device with a user. Input from this device may be targeted at different machines in the space, but it is all treated as part of this user’s session in the space, and authorized accordingly.

This mechanism can be subverted by a user “lending” an input device, but this is the same risk as letting someone use your login session and we will not consider it further.

3.2 Heterogeneity

An Active Space is a device-rich environment, and users may often wish to interact with devices other than computers. Some of these devices require physical interaction for control, e.g., thermostats, projectors, or may be controllable with a remote control device. Many of these devices also provide a software interface by which they may be controlled. Our objective is to develop middleware that supports heterogeneity both in the devices that can be used as input devices and the target devices that they provide input to. For example, we would like users to be able to use their personal laptops or hand-held computers to interact with the computers in the space. We are also investigating the use of location-tracking devices as limited input devices (the devices we have contain a pair of programmable buttons that can be programmed to act like mice). On the target side, other than the various computers in the space, users most often interact with the sound and light systems in the room, and these are controllable programmatically as well.

This is similar to the idea of Universal Remote Controllers for home entertainment systems. However, integrating the input device with the location-tracking device has some additional advantages. First, the user could point at the target device and click, and the location system can identify the device he wishes to control. Also, a graphical user interface for controlling this particular device can be shown on a nearby display. The user need not remember the various things
that this button maps to for controlling different devices—the GUI can label the controls according to the context.

3.3 Clicky as a user-proxy

There are various applications for which it is useful to log a user session in an Active Space. Clicky provides a single point from which a user’s activity can be observed. Apart from helping with debugging, it provides useful input for profiling user behavior, which helps with developing adaptive mechanisms to tailor the space to an individual user’s preferences, based on his or her past behavior.

Tracking a user’s activity in the space can be helpful for usability studies and for studying the effectiveness of new techniques of user interaction. This tracking requires the logging of events on multiple separate machines and then sorting and correlating the data into user sessions. Using Clicky allows us to log the user sessions directly.

Another common activity in our space is to videotape activities (such as seminars or presentations) that occur, so as to allow remote users to participate. Given the number of display devices in the space, it is not always clear what the focus of the current activity is, and where the camera should be aimed. Clicky can act as an indicator of focus for the speaker—the camera can be set to follow the pointer.

3.4 Collaboration

Many applications in our Active Space involve a group of users collaborating to perform some task. Allowing multiple users to simultaneously interact with an application on a large shared display is useful. Operating systems typically expect each display connected to a computer to have only one source of input; however, a middleware approach to share the display between different input devices in the room would be useful. We have multiple users in the space, each with their own input device, interacting with a set of display devices. Users should be able to seamlessly switch the displays they want to send input to, and multiple users should be able to use the same display at once.

3.5 Application interaction

Gaia provides an application framework that allows the composition of applications across the various devices in an Active Space. Gaia middleware services also allow application components to be moved across different devices. Currently, the target display to move a component to can be selected by means of a pull-down menu that lists the available active displays. Integrating Clicky with this application would allow users to drag-and-drop applications (or pieces of applications) across displays, which are actually different machines, thus providing a more intuitive interface to this functionality.
3.6 Security Considerations

While Clicky is extremely useful in shared, collaborative environments, there are various issues of privacy and security that arise. Our approach to these issues reflects the nature of our environment; different situations may require different approaches.

If multiple users are allowed to share a display, and interact with the same windows, authorization issues have to be considered. Our implementation allows sharing of public displays, i.e., all users who are allowed into the room are automatically authorized to use these displays, so this is not an issue. Other situations may require authorization for display use. If input devices are associated with authenticated users, it is trivial to disallow input from unauthorized users to a particular display.

One option could be to allow users to request “exclusive” ownership of a display and prevent other users from issuing commands to it for the duration. Finer granularity locking, such as of a window or application, rather than an entire display, may also be useful.

A “personalized” input device may also provide a single point to launch attacks on a user. A denial-of-service attack on a particular user’s input device would leave them unable to access anything in the space. Similarly, since Clicky provides a single point to view user activity in the space, unauthorized access to this information would be a severe breach of user privacy. Another concern is the ability of an attacker to inject fake input messages into the space. In our environment, this problem is addressed by using Gaia’s mechanisms for secure communication. However, similar tools for using Clicky in other environments may require further consideration.

4 Design Objectives

From the above applications, we identify the following requirements for Clicky.

− One input device per user: The main objective of Clicky is to allow a user to interact with all the devices in the space using a single input device. Clicky must at least support the ability to start new applications, click on menus displayed on wall displays, or enter text into windows. It must also provide the ability to interact with non-computer devices in the space, e.g., lights, projectors, cameras, that can be controlled remotely.
− Support a variety of input devices: Since finding the “right” keyboard and mouse had proved a problem, we want Clicky to be usable with any available input device. A user entering the space should be able to use either publicly available keyboards and mice or their own personal devices (such as a laptop or hand-held computer) to interact with all the computers in the space. In addition to regular mice, we would like to allow the use of devices such as Ubisense tags or bluetooth cellphones as point-and-click devices.
− Location-tracking: Providing services to users based on their location is an interesting idea for ubiquitous computing environments. Integrating the user
identification and location services with the input system may allow for easier binding of the input device to the desired target device.

– Usability: Active Space users have varying technical backgrounds. Therefore, Clicky must be intuitive and easy to use. We did obtain feedback from users of our Active Space while building Clicky, though we have not yet conducted any formal user studies.

– Performance: Low latency is important for user-perceived events such as responses to keyboard and mouse events. Clicky must not introduce perceivable delays between user input and its realization on a remote display.

– Independence of Active Space: Basic input redirection (i.e., mouse movements, clicks and key presses) should require minimal “system” support. This allows Clicky to be used during the Gaia boot process, as well as for debugging the system infrastructure. It also allows Clicky to be used in other device-rich environments. Integrating Clicky into the Active Space may provide some interesting applications, but it should not be a requirement for Clicky operation.

– Transparency: It is not practical to require application or OS modifications to provide this functionality in our environment, hence a middleware approach was the most suitable.

– Effectiveness for collaboration: Many of the applications in our Active Space involve groups of collaborating users, such as meetings and classes. Clicky is designed to foster collaboration by allowing maximal participation and limiting users as little as possible.

We are in the process of implementing Clicky in the light of these objectives; we now describe the current state of the implementation and the directions of our work.

5 Implementation

We have a preliminary prototype of Clicky implemented and used in our Active Space. While it does not yet provide all the functionality described above, we found that having a usable version running in our Active Space is useful—users provide feedback on features they like or dislike or would like to have. The current version has proved popular with users in our Active Space.

The current implementation of Clicky only supports Microsoft Windows platforms, but we have kept the system-dependent functions generic, so porting Clicky to other platforms such as Unix should be straightforward. Clicky is being ported to an iPaq running WinCE, to allow users to interact fully with the space using only a hand-held computer for input. We are also in the process of equipping our Active Space with location-tracking hardware, which will allow us to use the Ubisense tags as pointer input devices.

5.1 Architecture

Clicky consists of two main components: an endpoint runs on each display that forms part of the Clicky virtual display and each user runs a redirector. Broadly,
each redirector captures mouse and keyboard events from its host system, figures out which display in the room to send the event to, and sends it to the endpoint on that display. The endpoint on each display receives input events and realizes the event on its display as if initiated from its own input devices. Thus, each active redirector represents a user session in the Active Space. The endpoint is a CORBA service running on a display host that accepts keyboard and mouse requests from redirectors.

The public displays in an Active Space make up a virtual display. Edges of these displays are “linked” together so that when a user moves the mouse off the boundary of one display, it moves seamlessly to the “next” display. The space administrator provides a suggested space layout in a publicly available file found on a well-known space file server. This layout file provides a list of available displays and the neighbor information for each of them. Redirectors use this information to map mouse movements to the appropriate display.

Endpoints are started automatically on public displays at system bootup. On startup, an endpoint registers the display into Clicky, which updates the list of active displays maintained in the layout file. When a user starts a redirector, it queries the layout file to find the list of active displays and configures itself to use the available ones. User mouse events are mapped to an appropriate display based on the layout geometry and forwarded to the appropriate endpoint for that display. We refer to the endpoint that input events are being forwarded to at any moment as the “active endpoint”.

5.2 Choice of Redirector

A user can either use a mouse and keyboard provided by the space, or a personal laptop—in each of these, the redirector runs on the host computer controlling these input devices. The details differ slightly.

Laptop as Redirector When a redirector first starts on a laptop, the user is prompted for the location of the space layout file. The user can modify (a local copy of) this file to use only a subset of the available displays. Figure 2 shows a graphical view of the room layout.

Redirectors on laptops have two modes of operation: local display and virtual display. The redirector has a status icon on the system tray, which changes color depending on the mode. In local display mode, mouse and keyboard events pass uninterrupted to the host operating system. This allows a user to use her laptop normally without having to shut down the redirector. In virtual display mode, however, all mouse and keyboard events are sent to the active endpoint (shown as the gray-shaded display in the figure) and mouse clicks and key presses are suppressed on the local system to prevent actions from occurring on the local display host. Users can switch between these two modes by triple-rightclicking the mouse.

2 A triple-rightclick was chosen as being easy to remember and not likely to clash with other uses by applications.
Laptop users can customize the links between the public displays (i.e. the space layout) for their own redirector. This is particularly handy for users who are only using a subset of the displays and find it annoying to have to traverse displays they do not use, or who get confused after accidentally moving across an edge to the other side of the room. They can reconfigure their space layout to skip over displays or end at an edge by adding, removing, or changing the arrows in the layout GUI while in local display mode. Users also have the option to save their customized layout for future use, as long as the default layout of the room has not changed in the meantime.

Public Redirector The space can provide mice and keyboards for users. In our space, these input devices are either connected to machines with public displays or no displays at all. Therefore, while these redirectors can still manipulate the virtual display of the Space, there is no concept of a local display. So these redirectors do not have the capability to customize their layout or toggle between local and virtual mode. Of course, if other environments wanted their public redirectors to have the layout GUI then these endpoints can just run the laptop version of the redirector. Users may be required to authenticate themselves to the space before they are allowed to use a public redirector.

5.3 Multiple Mice

Clicky allows the concurrent use of multiple redirectors in the space and on individual displays. This means that multiple users in the space can each move a mouse around the same display, with little effect on the others. Two different users can also type concurrently into two different windows on the same display without losing any keystrokes. The endpoint accomplishes this by sending keypress events from a given redirector to the last window that it clicked on in the
current display, regardless of which window is currently in focus. Figure 3 shows a screenshot with two users using editors simultaneously.

Fig. 3. Multiple Mice on a Display. Two users with separate editing sessions on the same display.

Multiple redirectors on the same display works quite well for most uses, but there are a few artifacts that remain from our implementation. For example, a pop-up menu that one user opens may prematurely disappear if another user clicks on something else. This happens because the underlying host operating system knows nothing about multiple mice; this is all handled by Clicky “time-sharing” the mouse between multiple users. A redirector’s mouse as seen on the display is not really the system mouse, but a “fake” mouse. The system mouse is hidden until it is briefly needed to enact a click. There are two minor problems with this approach: the “fake” mouse is really a graphic that Clicky positions across the screen, and does not change shape to reflect system events (such as turning into an hourglass while the system is busy). Also, events that require knowledge of mouse entry, such as scrolling over entries in a popup menu, do not work entirely as expected (they can still be used by clicking).

Another artifact occurs when the title bars of multiple windows feverishly toggle color when being typed into concurrently. This results from the endpoint’s need to give temporary focus to the window for which a key event is bound. This only becomes a real distraction when the windows in question overlap.

The inconveniences caused by using multiple redirectors with Clicky are relatively small and can be solved with guidelines for good social conduct in an environment like our Active Space. More stringent controls will be required for more public environments. The fact that there are multiple mice in the space, however, can be confusing, especially at first. When there are multiple identical mice moving around the space, it becomes difficult to tell them apart, so we have some mechanisms to help with this.

The cursor displayed for each mouse can be colored to help users discern their mouse from others. We use only a few colors (i.e., white, yellow, red, blue, and green), so the mice are easily identifiable. On laptops, the redirector can
be used to select a color. On public redirects, this color is hard-coded and we put a matching color sticker on the input devices. Another possibility for mouse identification is to allow users to request that their mouse perform some eye-catching activity such as blink or “explode”.

5.4 Fault Tolerance

Developing and testing Clicky involved numerous machines communicating in an experimental network. The inherent instability of such configurations encouraged us to address the problem of fault tolerance at an early stage. The basic design decision was to make the comings and goings of redirectors and endpoints as independent of each other as possible. Clicky should continue operation even if individual components such as endpoints or redirectors fail. As mentioned earlier, endpoints write their handle to the layout file at startup. Similarly on startup, redirectors add themselves to a spacewide list. These lists can be used by restarting endpoints and redirectors to synchronize themselves with other running components.

When an endpoint comes online, it notifies the current redirectors of its presence, and they can then add it into their layout. When an endpoint goes down, either by a user action or by crashing, the redirector detects this and removes it from the space layout. Once this is done, the mouse skips over this display. The redirector can reconfigure the virtual display to exclude this dead display. For simple configurations, this can be handled automatically, but may sometimes require user input to obtain a desirable reconfiguration. Handling redirector failure is easier—a redirector can go down and come back up without affecting the endpoints. Thus, the order in which endpoints and redirectors come up is unimportant.

5.5 Dynamic Reconfigurability

The space layout used by the redirector reflects the configuration of displays in the room and are not expected to change often. However, it is possible to change the layout, for example, to add a laptop and projector that should be part of the room temporarily. This uses the same mechanisms that provide fault-tolerance: the ability of the redirector to detect new endpoints when they come up. Currently, this is implemented by manually editing the layout file to specify the new configuration and starting up the new endpoints. The redirectors can re-read the configuration file and update the room layout accordingly.

5.6 Evaluation

Clicky is popular amongst users in our Active Space for interacting with the displays that are around the walls in the room. New users find it easy to start using Clicky. Performance is good enough that no lag is noticed between mouse movements and response.
We designed Clicky to be independent of Gaia, so that it would be easy to use in general environments. We did find additional applications that benefit from integrating it with Gaia, but the standalone design has proved a good decision. The middleware approach makes it easy to port to the variety of heterogeneous devices that are likely to be used as pointing devices in an Active Space. While Clicky is currently most often used with one of a set of wireless mice we have in the room, we plan to use other devices in the room, such as UbiSense badges [16], as location-sensitive input devices.

Clicky has been useful in collaborative applications in our Active Space; however, more user studies on human-computer interaction could be useful.

Dynamic reconfiguration of the space layout was easy to support, since redirectors can re-read the configuration file and do not need to be restarted if the room configuration changes. While our space configuration does not change frequently, this feature is useful for allowing a user to construct a virtual display consisting of a subset of the space displays.

Multiple mice on a display is very useful for collaborative applications, but can be a problem with non-cooperating users stealing focus from each other. This is not a problem in our environment, since we expect users sharing the room will be able to work out a sharing protocol, but would need to be addressed if we wished to use this in a more public environment.

Mouse movements have to be scaled between the redirector and the target display if they do not have the same resolution. This is especially true as we integrate PDAs as redirectors in the space, since they have much lower resolution than the target displays. The Pebbles [7] project has addressed many of the issues with using PDAs as input redirectors.

5.7 Ongoing work

The current implementation satisfactorily handles user input from a mouse and keyboard (whether wireless, or connected to a desktop or laptop computer) to control a set of displays. We are working on extending both the set of devices that can be used as input devices (to include PDAs and UbiSense tags) and the set of devices that can be controlled (such as lights, audio volume, projectors) using Clicky.

To address the problem of identifying users in the space, we are considering issuing users who enter the space a badge that can be used as an input device. Obtaining the badge thus authorizes the user in the space and allows them permissions to perform some activities. For example, “visitor badges” would allow their holders limited permissions within the space. Another option is for users to be allowed to select an input device after authenticating themselves to the space. Gaia can then associate all input from that device as coming from this user, and perform access control accordingly.
6 Conclusion

We have presented a user-centric model for input into a smart space. Apart from allowing users to conveniently access multiple displays around the space, we provide the notion of a user session in the space by tagging all input from a given source with an authenticated user. The main idea is to allow an individual user to interact with all the devices in an Active Space by means of a single input device of his choice.

A single input device per user helps with the problem of authorized input to public displays. Authenticated users can be associated with an input device, and all input from that device forms the user “session” in the space. Another application uses the input device as a point of focus in the space for activity-tracking applications. A further application integrates the input device with location-tracking systems, allowing the user to point at devices in the space to indicate interest in interacting with them.

We are in the process of developing Clicky as a middleware tool to provide this functionality. It currently allows users to interact with multiple displays using a single mouse and keyboard. Key features that Clicky provides are the ability to use any available mouse and keyboard as a personal input device, support for multiple such devices in the space, including concurrently on individual displays, support for dynamically reconfiguring displays in the space into a “virtual display”, and achieving this without requiring modifications to the operating system or applications.

Clicky has been developed and is being used in our Active Space. Lightweight middleware tools improve the usability of such environments. We envision future work with user studies and applications that use Clicky. Tools for automatically configuring the virtual displays may also be useful.

References


