Patch Panel: Distributed I/O Management for Ubicomp

Rafael Ballagas
Media Computing Group
RWTH Aachen University
52072, Aachen, Germany
ballagas@cs.rwth-aachen.de

ABSTRACT

The goal of this work is to enable the concept of focus for the ubicomp domain by providing distributed I/O management. This includes developing an infrastructure solution that provides dynamic interoperability, security, and acceptable latency for interaction with ubicomp applications. It also includes the development of new focus models that hold up to the broader characteristics of ubicomp applications. Then using this system we can study the user issues and conceptual difficulties associated with distributed I/O.

INTRODUCTION

In the traditional desktop environment, there is a single user with a single point of focus. The primary output mechanism is a graphical display and input is done primarily with a mouse and keyboard. Human interaction with desktop applications is coordinated through the window system, as shown in Figure 1. One role of window systems is *input management* (e.g. management of event flow and event abstraction). The base window system canonicalizes, time-stamps, and orders events, then it determines in which window the mouse pointer resides. The window manager makes a routing decision based on focus policy (usually, *click-to-type*), event type, and current focus [1]. The UI toolkit may abstract events to widget specific event types before the event arrives at the application.

Using window systems for input management is time tested and works well for desktop environments, but quickly breaks down outside of this realm. For example, in ubicomp environments, there are multiple users, each with their own point of focus interacting with multiple devices, using multiple modalities. Output will not be limited to the window construct or graphical output, but may utilize ambient displays or physical devices. In addtion, ubiquitous computing environments are constantly changing due to evolution of physical spaces, introduction of unanticipated entities (e.g. applications, devices, services), and the transient nature of people moving in and out of these spaces.

Desktop systems are also self-contained entities, making many aspects like security and latency for local I/O less significant. For example, it is acceptable to trust that the input device connected to the computer is being manipulated by the person currently logged in. Additionally, it is reasonable to assume that no unauthorized entity will be able to illegally view the input from your device[1]. However, these assumptions are no longer valid in the distributed environments typical of ubiquitous computing. Sensitive data between devices and applications must be adequately safeguarded from unwanted parties on the network. Additionally, access control mechanisms must be in place to verify that the user manipulating a certain device has permissions to interact with a particular application. Also with distributed systems, lag is inevitable and can have a multiplicative effect on the Fitt's law index of difficulty [2] and greatly diminish human task performance. A distributed I/O management infrastructure must provide information security and privacy as well as a high system performance to maintain fluid interactivity.

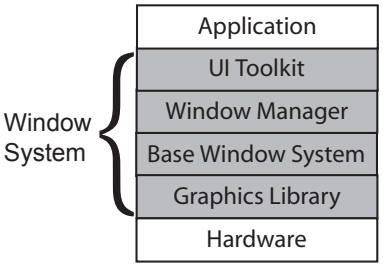

Figure 1. Classic layered model of the window system. All window systems do not explicitly implement the layer architecture, but all do implement the functionality represented by the layers. [1]

INFRASTRUCTURE DESIGN

We propose a new system design based heavily on our experience with the iStuff toolkit [3]. Intermediation will be used as the fundamental mechanism to support ad-hoc interoperation between heterogeneous components as demonstrated in our previous work on the Patch Panel [4]. There are two main drawbacks from using intermediation for distributed I/O management that pose interesting research questions from a systems architecture point of view: security and latency.

[1] assuming the machine is adequately protected from Trojan Horse or Spyware attacks

Security

Intermediation is a mechanism that intercepts and rewrites messages, which is exactly the activity that traditional security measures are designed to prevent. A research question arises: how can we provide secure intermediation? Our design slightly deviates from traditional end-to-end security. It consists of a secure connection between the event origin and the intermediary, and then another secure connection between the intermediary and the destination. In a centralized Patch Panel architecture, this is easily criticized because a single entity knows all secrets. Instead, our system design incorporates one instance of the Patch Panel per user, reducing the impact of a security breach to a single user. This architecture has an advantage in that it guarantees the user source of the data instead of the physical source, facilitating access control mechanisms.

Latency

Intermediation adds a level of indirection between distributed hardware and applications increasing system latency. This latency proved unacceptable for direct manipulation under certain common conditions like network load [4]. We propose to build a transport mechanism that is optimized for the two-hop communication found with intermediation. Optimizations will include:

- *Quality of Service:* We will include event priority mechanisms with three priority levels (from highest to lowest): direct manipulation, command-style interaction, and status updates. We will also explore techniques to prioritize UI network traffic over other data traffic on the network.
- *End-to-end flow control:* Standard TCP flow control can result in undesirable queueing in the intermediary. Higher level flow-control is required to optimize network traffic for the intermediation-centered communication.
- *Minimal network load:* Network transmissions will be limited to events that have outstanding subscriptions.
- *Distributed work load:* With large numbers of users and mappings, a centralized Patch Panel can become a performance bottleneck. The per-user architecture distributes the work load reducing this effect.

Evaluation

The evaluation of the distributed I/O system performance is a research question in itself [5]. Although constant lag is well understood as a factor in determining human performance, the lag commonly found in distributed settings today is random. We propose to perform a Fitt's Law style analysis of human performance under stochastic lag to understand the tradeoff between mean and standard deviation of lag. With this work, we can take measurements of our system performance over time to judge if our system performance affords acceptable human task performance for direct manipulation.

FOCUS

The focus policy is central to achieving distributed I/O management. How will a user associate input with interactive digital elements in a ubiquitous computing environment?

Several distributed focus policies have been attempted in other research projects. For example in the XWand [6] project, a user may point a physical wand at a device and then control it using simple gestures. Alternatively in the PointRight [7] project, a static spatial layout of the room is used to guide the redirection of a pointer and the users focus between different machines. A system might potentially monitor the users gaze as an indicator to redirect focus. We plan to continue to identify and comparatively evaluate the different focus mechanisms by measuring task completion times and success rates for common ubicomp tasks.

CONCEPTUAL MODELS

A system with distributed I/O management will enable advanced study of user centered issues, including:

- *Device ownership:* how does a user associate and relinquish a device with his identity?
- *Troubleshooting:* Can an average desktop computer user deal with the added complexity of distributed I/O? What mental models do they have of the interactions and how do they go about solving problems when things go wrong?
- *Metaphors for Lag:* is it possible to mitigate lag with new feedback mechanisms like a haptic progress bar, or cursor decorators [8]?

RELATED WORK

Many ubicomp middleware infrastructures employ indirect communication mechanisms through publish/subscribe semantics that could facilitate intermedation [9, 10]. However, none of these systems directly support intermediation. Additionally, several systems have worked on input redirection and notions of distributed I/O management [6, 7, 11].

REFERENCES

1. J. Gosling, D. Rosenthal, and M. Arden. *Windows System Architecture: History, Terms and Concepts*, pages 23–52. Springer-Verlag, 1989.
2. I. MacKenzie and C. Ware. Lag as a determinant of human performance in interactive systems. In *Proc. CHI*, pages 488–493. ACM, 1993.
3. R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments. In *Proc. CHI*, pages 537–544, Ft. Lauderdale, FL, USA, April 2003. ACM.
4. R. Ballagas, A. Szybalski, and A. Fox. Patch Panel: Enabling Control-Flow Interoperability in Ubicomp

Environments. In *Proc. PerCom*, Orlando, FL, USA, March 2004. IEEE.

5. W. Keith Edwards, Victoria Bellotti, Anind K. Dey, and Mark W. Newman. The challenges of user-centered design and evaluation for infrastructure. In *Proc. CHI*, pages 297–304. ACM, 2003.
6. A. Wilson and S. Shafer. XWand: UI for intelligent spaces. In *Proc. CHI*, pages 545–552. ACM, 2003.
7. B. Johanson, G. Hutchins, T. Winograd, and M. Stone. PointRight: experience with flexible input redirection in interactive workspaces. In *Proc. UIST*, pages 227–234. ACM, 2002.
8. Carl Gutwin, Steve Benford, Jeff Dyck, Mike Fraser, Ivan Vaghi, and Chris Greenhalgh. Revealing delay in collaborative environments. In *Proc. CHI*, pages 503–510. ACM, 2004.
9. B. Johanson and A. Fox. The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In *Proc. WMCSA*, page 83. IEEE, 2002.
10. M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *Proc. PerCom*, pages 263–273. IEEE, March 2004.
11. M. Newman et. al. Designing for serendipity: supporting end-user configuration of ubiquitous computing environments. In *Proc. DIS*, pages 147–156. ACM, 2002.

BIOGRAPHY

Rafael Ballagas is a computer science doctoral candidate in the Media Computing Group at RWTH Aachen University with an anticipated graduation date of September 2006. His research interests include distributed I/O, tangible interfaces, and infrastructure solutions for ubiquitous computing environments and interactive media applications. Rafael has developed the iStuff toolkit for rapidly prototyping physical user interfaces and the Patch Panel intermediary to support incremental integration and reconfiguration of toolkit components. He has an MS in electrical engineering from Stanford University and a BS in electrical engineering from Georgia Institute of Technology. Upon finishing his degree, Rafael hopes to become a professor in computer science with a research emphasis on Human Computer Interaction and Ubiquitous Computing.

Rafael has been active in research of interactive spaces including two-years of research in the iRoom at Stanford University. Now he is helping to construct a new interactive space envisioned by his supervisor Jan Borchers with a focus on interactions with time-based media at RWTH Aachen University. Other professional activities include co-organizing the workshop *Toolkit Support for Interaction in the Physical World* at Pervasive 2004. Rafael is also active in teaching as a teaching assistant for the computer science courses *Designing Interactive Systems*, an introduction to Human Computer Interaction, and *HCI Design Patterns*, an introduction to how patterns can be applied to the field of HCI, and *The Media Computing Project*, a project based course with a focus on ubiquitous and media computing.