# Designing for Latency: In Search of the Balance Between System Flexibility and Responsiveness

**Brian Lee, Rafael Ballagas**
Stanford University
{balee, ballagas}@stanford.edu

**Maureen Stone**
StoneSoup Consulting
stone@stonesc.com

**ABSTRACT**
This paper outlines experiences with latency and responsiveness in a distributed application. We analyze latency from the perspectives of the user and the system. From these experiences and analyses, we suggest questions and highlight tradeoffs that pertain to both HCI and systems in the realm of ubiquitous computing.

**INTRODUCTION**
In traditional computer systems, which heavily utilize the desktop metaphor and WIMP-style interface, there exists a tightly coupled user interaction loop. Pointer inputs are specially integrated into the operating system, bypassing normal system event queues. Thus, the pointer is exceptionally responsive to user input; device movements are echoed almost instantaneously.

For the Interactive Workspaces project [2], we have created a loosely coupled, room-based interactive space, using a system infrastructure, the Interactive Room Operating System (iROS), which offers inherent decoupling to better support heterogeneous environments. iROS provides a network abstraction consisting of an event server (the Event Heap) and a translation intermediary (the Patch Panel) to handle communications between workspace entities.

In ubiquitous computing systems, there are several advantages to loose coupling, including dynamic configurability and ease of integration for heterogeneous environments with multiple users, computers, and displays. Over three years of experience in the iRoom, we have found that the iROS system provides excellent and useful coordination between applications and users.

Given our experiences at the application level, we would like to apply the above principles to input. However, there are several drawbacks to loose coupling, the most prominent of which is latency. A common example of latency from the traditional computing realm may be found in VNC [5], a remote display system that allows a user to view his or her computer desktop from another machine over the Internet. Though handy, the VNC user experience is often frustrating: no longer is there a single machine, tightly coupled pointer input loop. Instead, a stream of mouse event messages travels over the network from the local computer to the remote computer, and a stream of output bitmaps travels back along the same path. This adds a significant delay (latency) between the time the user performs a mouse action and the time he or she sees the result of the action. Added flexibility comes at the expense of perceived responsiveness, which is noticeably worse than the single machine case.

We examine some of the tradeoffs between system flexibility and responsiveness in the design of this and other distributed ubiquitous computing systems.

**LATENCY**
To understand responsiveness, we must look at latency from the perspectives of both the user and the system, prompting two questions. First, what are the current bottlenecks of the system? This question can be answered with benchmark timing measurements, to be provided. Second, how much system latency is acceptable for the required user interactions? On the surface, this is a well-researched topic that should provide us with empirical data indicating some thresholds. Where they have been applied to computer science, however, they have mostly been applied to traditional computing systems. It is an open question as to how existing results can be applied to ubiquitous computing spaces, and where new analyses are needed. Ultimately, the goal is to develop theories for design in the presence of latency, which will always be a factor in loosely coupled, highly flexible systems.

**Sources of Latency**
Figure 1 outlines the possible sources of latency in one example system interaction. The layout illustrates the distributed nature of the overall system, in which input, system, and output are not necessarily tightly coupled. They may comprise several different devices or computers.

We have identified three general types of latency in ubiquitous computing systems:

- *Internal* latency – latencies internal to a single system device or computer, e.g. program processing times
- *External* latency – latencies in communication between system devices or computers, e.g., networking
- *Human* latency – latencies arising from human factors, including physical issues of input and output, as well as psychological issues such as attention and perception

Note that the arrows themselves may encompass several smaller latencies. For example, the latency of the input

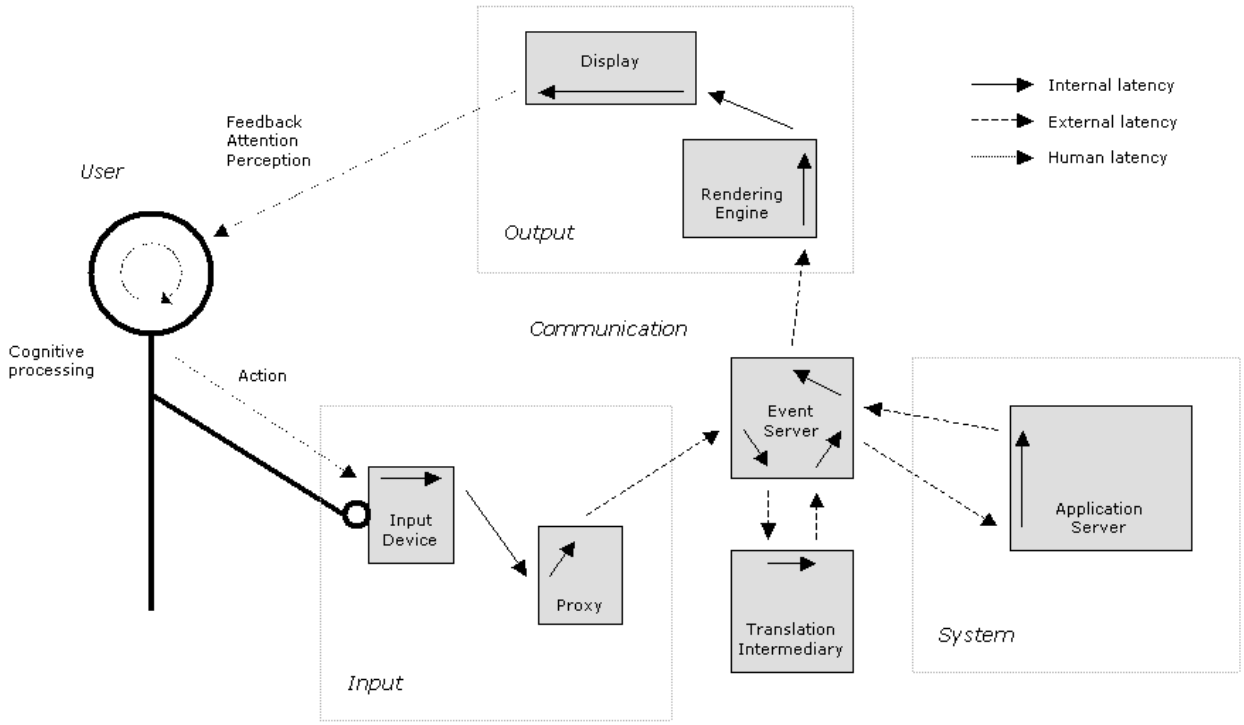


**Figure 1:** Diagram showing possible sources of latency in an example distributed system.

device arrow may include factors such as hardware sensor sampling rate, wireless transmission time, or device driver processing time. Most of these factors are outside the control of system designers; we exclude them from the diagram for clarity.

Taken together, these arrows form the overall latency of the system. We seek not only to make measurements of each of these factors, but also to understand the deeper implications and tradeoffs, and in turn to develop general theories and guidelines for HCI and systems designers.

### A User's Perspective

From a human-computer interaction perspective, it is important to understand how users perceive latency. The arrows representing human factors in Figure 1 describe some of the effects that add to the perception of latency, but the issues underlying them are complex and not well understood. For instance, in existing systems, users tolerate much more latency in the execution of complex commands, such as recursive file deletion, than they do in the movement of mouse cursors. What levels of responsiveness do users expect and consider acceptable? How do these expectations vary with modality or task?

A key point in thinking about expectations of responsiveness is causality, the feeling that user actions directly correspond to system reactions. Different modalities appear to have different performance and feedback requirements for maintaining the impression of causality; this is the motivation behind the special integration of mouse and keyboard input in traditional desktop operating systems, for instance. Other issues that may affect user expectations include preconceptions from existing systems (e.g., mouse performance in traditional computers) and new factors introduced by the distributed nature of the system (e.g., physical distances from users to inputs and outputs).

Another interesting question is: how do users cope in the presence latency? Much of how users deal with latency has to do with predictability: if delay is predictable, users can develop strategies that account for known delays, but if delay is erratic or unknown, they cannot do anything about it in advance. This may have larger implications in the underlying design of ubiquitous computing systems, as minor variances in performance become highlighted and magnified in user interactions.

## EXPERIENCE

Our own experience with system latency stems from prototyping user interactions in the iRoom using the iWall application. We developed iWall as a distributed blackboard system to experiment with moving items around a room. Using a traditional Model/View/Controller design philosophy, we separated the blackboard model containing knowledge of object location in this abstract

space from the multiple views around the room, each displaying a small subset of the space. The implementation is such that the iWall model is an application that runs on a central server and remotely controls each viewer application running on separate machines.

This distributed implementation implies at least one network hop is added to the latency of the system. iWall takes advantage of the Event Heap, which serves as a useful network abstraction that has many benefits over a direct socket connection, despite the fact that it was not originally designed for streaming user input events. The Event Heap is a centralized tuplespace with publish / subscribe semantics. Each event posted to the Event Heap must first go to the centralized server before it reaches the subscribing client, effectively adding an additional network hop to the iWall system.

The goal of the iWall experiment is to understand which modalities or combination of modalities were most effective for moving information around the room. We are interesting in building a multi-user, post-desktop environment, requiring a facility for multi-cursor interaction. We used the iStuff [1] toolkit and architecture to provide the multimodal, multi-cursor interaction. The iStuff architecture fundamentally includes a level of indirection known as the Patch Panel to quickly map devices to applications. The resulting system (shown in Figure 1) includes three Event Heap hops, for a total of six network hops. We realize that building so much flexibility into our system will inevitably degrade responsiveness; what we seek is a balance of these two goals.

## ANALYSIS

The iWall provided a sluggish, though usable, user experience for moving cursors or objects around the room. We made some measurements to get a rough estimate of the bottlenecks of the system:

- Event Heap ping – Measured the time it took for a client to receive an event that it sent out. 5-10ms
- Patch Panel ping – Measured the time it took for a client to receive the result of a translation corresponding to the event that it sent out. 12-17ms
- Standard network ping – For wired network traffic on the same subnet, 0.1-0.5 ms; for wireless network traffic, 2-3ms

The insignificance of the network delay in our system was somewhat surprising. The internal latency of the multiple system components, including the Event Heap and Patch Panel, made up the bulk of the system delay.

We began to consider certain optimizations to decrease system latency, but each optimization has significant consequences for the system architecture. For example, the iWall could eliminate the need for the Patch Panel by hard coding interfaces for each anticipated device or modality, but a consequence of this is an undesirable requirement that the iWall code be changed each time a new device or modality is introduced. Alternatively, the Event Heap could be modified to include the Patch Panel functionality. However, the Event Heap is a well-tested communications channel that would ideally be left unmodified, and any integration task would require a significant coding effort. Lastly, the centralized iWall model could be reimplemented as a decentralized model with each viewer maintaining a copy of the shared distributed state, but such models are generally harder to implement correctly.

Before putting the effort into any of these paths, we want to understand whether these optimizations would provide sufficient improvement to system latency. We anticipate using several different interaction models in our application, such as direct physical pointers, indirect virtual cursors, and voice command. Each of these interaction models is likely to have different responsiveness requirements. We analyze some of the issues brought up by the system interactions under consideration.

### Pointer Manipulation

With respect to pointer manipulation, Fitts' law is the most commonly referenced model for determining acceptable input lag. Several [3,4] have proposed a slightly modified version of Fitts' law that incorporates lag as shown in Equation 1:

$$\text{Mean Time} = C_1 + C_2(C_3 + \text{MachineLag})\text{ID} \qquad \text{(Eq. 1)}$$

$C_3$ represents the human processing time required to make a corrective movement, MachineLag represents the system processing time, $C_2$ID represents the average number of iterations of the control loop, and $C_1$ represents the sum of the initial response time and the time required to confirm the acquisition of the target. Ware et al [4] applied this modified Fitts' Law to 3D virtual reality systems and based on their task performance measurements define a lag target of less than 50ms for input devices.

The implications of this result raise several questions for ubiquitous computing environs. Does this version of Fitts' law hold in ubiquitous computing applications? If so, how do the parameters change for this type of environment? What latency thresholds do these parameters suggest? Additionally, noting that forearm movement has a physiological upper bound on rate of control at 3Hz and also noting that Ware's results are based on forearm controlled devices, do other modalities like leg controlled, or gaze controlled interfaces require different latency thresholds?

### Command Interfaces

Command interfaces, such as spoken instructions, do not have the same performance requirements as pointer manipulations, but do need to maintain causality: the user must perceive that each action causes the appropriate reaction in the system. It is well documented that the latency threshold for weblinks is approximately 1 second.

Does this result apply in some fashion to ubicomp spaces? How do properties of modalities affect the maximum allowable latency to preserve perception of causality?

### Constant vs. Variable Latency

Loosely coupled systems with many sources of latency will present variable amounts of latency at different times. Our system suffers from highly variable latency. This is partly due to the use of an Ethernet network, which uses statistical multiplexing as policy for sharing the network. Thus, network latency is variable and increases as network traffic increases. In addition to network variability, each system component is implemented as a Java application. Each Java Virtual Machine adds variability due to the unpredictable nature of the garbage collection process.

There are several real world examples where users can be trained to deal with constant latency. Shooting a gun at a moving target requires that the marksman lead the target to account for the time it takes for the bullet to travel through the air. As noted above, however, users cannot use the same tactics in coping with variable latency. At what point does the user feel the variance of the system? Does this variability make the certain interactions inappropriate for our system? Should system designers use networks and programming languages that provide more consistent delays or execution times?

### Flow Control

Our experiments and measurements highlighted other system design decisions that have direct impact on the user experience. The Event Heap has no mechanism for flow control, so input devices may produce I/O events at a rate faster than the iWall application can handle. If the sampling rate of the device were 20Hz, and the iWall application were only able to process events at 10Hz, when the user interacts with the device for 1 sec, the last event would have an additional 1 sec of queue latency in addition to the existing system latency. In practice, we have found that event queueing increases latency. Where possible, flow control should be used to avoid queue latency.

## FUTURE WORK

As computing and network performance increase, so do the goals and ambitions of computing systems. Human performance, meanwhile, remains relatively constant. Thus, we believe that this issue of responsiveness, from the perspective of both systems and HCI, will remain an interesting one for the foreseeable future. In addition, there are several research directions to be pursued:

*Multimodal interfaces*: We are investigating multimodal interactions, combining pointers with voice commands. In a distributed system, such interactions will include latencies from several different system components, and thus possibly introduce new critical paths. How does mixing modes affect latency, from both the user and system points of view?

*Precision*: Another question is the ability of distributed applications to support spatial and temporal precision, especially in light of the latency issues already raised. Can distributed applications be designed such that users can successfully hit a small button (spatial precision) at a precise instant (temporal precision)? One central tradeoff: sampling rate vs. latency, as faster sampling rates generally lead to larger latencies, due to system bottlenecks.

*Memory and attention*: Interactive spaces and other ubiquitous computing environments, with multiple users, input modalities, output modalities, etc., often tax users' attention and memory in ways that traditional computing systems do not. The resulting distractions or interruptions may have implications for perceived responsiveness.

## CONCLUSION

As ubiquitous computing systems become both more prevalent and more interactive, we need to discover the designs and paradigms that will allow users to use these spaces as comfortably as they now use workstations. Responsiveness is a key element in proper HCI design, while flexibility is a key element in proper systems design; these two ideals often make conflicting demands when designing actual systems. In this paper, we have identified some of the tradeoffs involved. This workshop presents an excellent opportunity to discuss the experiences of designers in the ubiquitous computing community.


## ACKNOWLEDGMENTS

Terry Winograd has been one of the principle advisors and sources of wisdom for this research. Also, several students have worked on components mentioned herein: Jeff Raymakers was the author of the iWall rendering engine; Andy Szybalski worked extensively with the Patch Panel; Susumu Harada and Jennifer Hwang worked on speech interfaces for iWall applications.


## REFERENCES


1. Ballagas, R., Ringel M., Stone M., Borchers, J. iStuff: a Physical User Interface Toolkit for Ubiquitous Computing Environments. *Proc. Of CHI '03* (Ft. Lauderdale FL, April 2003).
2. Johanson B., Fox A., and Winograd T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing Magazine* 1(2), April-June 2002.
3. MacKenzie, S., and Ware, C. Lag as a Determinant of Human Performance in Interactive Systems. *Proc. Of INTERCHI '93*, 488-493. New York: ACM.
4. Ware, C., and Balakrishnan, R. Reaching for Objects in VR Displays: Lag and Frame Rate. April 20, 2000.
5. Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood & Andy Hopper, "Virtual Network Computing", *IEEE Internet Computing*, Vol.2 No.1, Jan/Feb 1998 pp33-38.