Virtual Machines for Remote Computing: Measuring the User Experience

Brandon Taylor, Yoshihisa Abe,
Anind Dey, Mahadev Satyanarayanan,
Dan Siewiorek, Asim Smailagic

July 2015
CMU-CS-15-118
CMU-HCII-15-101

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Virtual Machine (VM) encapsulation provides a way to provision existing software over the Internet without any code modifications. However, distributing software introduces performance costs that vary in complex ways. Our study provides empirical evidence of how network conditions, software interfaces and VM provisioning techniques impact user experience. Using these results, designers seeking to provision existing applications can better understand the usability impacts of their design choices. Additionally, our study highlights the limitations of existing system response time guidelines derived from HCI research conducted using text-based terminal applications. In particular, we were able to identify how various, common aspects of GUI application interactions are impacted by system delays and relate these effects to decreases in user satisfaction.

This work was supported by the Alfred P. Sloan Foundation, the Quality of Life Technologies ERC (NSF-EEC-0540865), the Vodafone Group, and by the National Science Foundation under grant number IIS-1065336. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and should not be attributed to Carnegie Mellon University or the funding sources.

Keywords: user experience, virtual machines, demand paging, caching, prefetching

1 Introduction

Recently, Forbes Magazine reported that cloud computing is already a $100 Billion dollar market and is poised to grow even larger in the near future [6]. It is not unrealistic to imagine that users will soon expect to be able to access all of their data and software regardless of where they are. Finding ways to allow users to do so while maintaining crisp interactions on resource-poor mobile devices is a key challenge for cloud-based computing [31].

While many popular applications are actively being re-developed for cloud integration (e.g., Microsoft Office 365), there is a wide range of legacy software that simply cannot be redeveloped. One solution that would let users still access such software in a mobile-cloud environment is to use Virtual Machine (VM) encapsulation to pre-package and pre-configure legacy applications. Doing so would allow the large, existing base of PC applications to be used without necessitating any modifications to the software. However, VM encapsulation does not ensure satisfactory responsiveness for the user. The user experience will be dictated by the complex interactions between (1) the VM provisioning strategy, (2) the network conditions, and (3) the specific application interface.

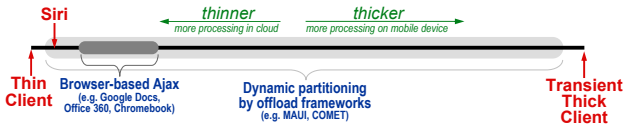

Figure 1: Approaches to Leveraging Cloud Services

As illustrated by Figure 1, there is a wide range of approaches that can be used by mobile devices to leverage cloud resources. In this paper, we will focus on the approaches at the extreme ends, neither of which require modifications to source code. A *thick client* delivery model requires the user to wait while the VM image is demand-paged and prefetched from the server, thus preventing any interaction for a time dictated by the state size and the network conditions between them and the server. Alternatively, using a *thin client* approach, a VM instance can be launched in the cloud and controlled over a wide area network (WAN) by sending user interaction information (e.g., mouse movements and key presses) to the cloud where it is processed, and output information (e.g., screen updates) are shipped back to the client machine. In this case, delays between interactions and subsequent responses are increased by the network's latency, often resulting in sluggish behavior. While systematic explorations of response delays across network conditions have been performed [18, 33], there has not been an appropriate exploration of how those delays interact with application interfaces to truly impact the user experience.

In fact, there has been very little research into the effects of system response delays on modern GUI interfaces. Most of the experiments that did explore the impact of system response times on usability were centered around text manipulation on terminal systems [4, 8, 12, 13, 17]. Over the years, these findings have become codified in textbook heuristics (see Table 1) without regard to specific application interfaces.

It is not difficult to see that such simplistic guidelines cannot possibly apply across all application types. Compare for example, the lengthy delays that users will tolerate in web browsers [14, 22] to the effects of even minor delays in virtual reality [9, 20]. While these two cases represent extreme examples, there is no more reason to assume that delays encountered during photo-editing

Table 1: Response Time Limit Guidelines Adapted from Nielsen [24]

Response Time	Effect on User
<.1 second	Imperceptible
.1 - 1 second	Maintains Train of Thought
>10 seconds	Maintains Attention

tasks will have the same effect as impact on usability as delays encountered while typing [33].

Given these unanswered questions, we take heed of Robert Miller's advice about his estimates of appropriate system response time: they "should, indeed, be verified by extended systems studies [...] in carefully designed, real life task environments and problems" [21].

Following this advice, we ran a user study to examine the impacts of system response times on user experience using a sample of modern computing applications and VM provisioning techniques. From this study, we were able to explore how a variety of common GUI application interfaces are impacted by system delays and provide empirical evidence relating these delays to user satisfaction. We also examined how different Virtual Machine provisioning techniques (thick client, thin client) interplay with typical real-world network conditions to impact the user experience. With these results, designers can consider the existing network conditions and properly evaluate the usability trade-offs of thick or thin client VM provisioning strategies for a particular piece of software.

2 Background

In this section we will overview previous research examining the user experience impacts of system response times. In doing so, we will highlight discrepancies between the empirical studies that inform the generally accepted response time guidelines and the use of modern application interfaces we are interested in. We will also review more recent systems research into remote access techniques that highlights the need for our research. Lastly, we will provide an overview of some of the remote access techniques employed in our study.

2.1 Usability Impact of System Response Time

Decades of human factors research have examined how delays in system response impact user experience. Surveys have demonstrated that system response time is the most important factor related to user satisfaction with computer systems [29]. Work stemming from mainframe time-sharing studies has shown that performance decreases as system response time increases [4, 8, 12]. Stress has also been shown to increase along with system response time [17]. More recent studies have demonstrated a negative correlation between response time and satisfaction with web pages and browser-based applications [14, 22].

In 1968, Robert Miller wrote perhaps the most influential work on response time requirements [21]. In his paper, he discussed 17 different contexts in which "different kinds of response and response delay will be appropriate". The contexts discussed ranged from the response to a

key being struck ("no more than .1 to .2 seconds") to the execution of a program script (within 15 seconds to keep "in the problem-solving frame of mind"). While, he does provide a number of heuristics, Miller cautioned that "the reader should accept the parameters cited as indicative rather than conclusive" and encouraged empirical studies to validate his claims.

A handful of studies conducted in the 1970's and 1980's did seek to lend some empiricism to Miller's guidelines. The vast majority of these studies involved only text entry or editing [4, 8, 12, 13, 17]. A rare exception to this was Goodman's study [11], which had users manipulating a graphical plot using a lightpen. However, to our knowledge, no such studies have examined the impact of response times on user experience in modern GUI applications.

More recent studies using modern interfaces have focused on perceptual thresholds of delays rather than the usability impacts of them [7, 15, 23]. For example, it was found that users perceive delays as low as 2.38ms in touchscreen response during dragging interactions [23]. However, as can be seen by the success of commercially available touchscreens, which often have delays nearing 100 milliseconds [15], perceptual thresholds do not necessarily inform levels of satisfactory performance. Our study is focused not on the limits at which users no longer perceive delays, but instead on the degree to which users will tolerate delays that inevitably occur in mobile-cloud systems.

2.2 System Research: Remote Computing

Recent systems research exploring the response times of remotely accessed Virtual Machines has brought the gaps in usability research for modern GUI interfaces to light. Lai and Nieh [18] set out to compare various thin client models over real-world network conditions. They based their analysis around a series of benchmark measurements collected using eight different thin client platforms. While this provided information about the relative performance of the platforms, no users were involved to verify the systems' effectiveness.

Tolia *et al.* [33] explored the effects of network latency on system response time in a variety of applications using Virtual Network Computing (VNC). Recorded traces of interactions with four applications were replayed over eight different network settings and measurements were made between input commands and output results. Abe *et al.* [1] performed a similar experiment using a thick client approach to remote VM delivery. The study ran traces for six different applications across a set of network conditions and recorded the variety of delays encountered.

Whereas Abe and Lai and Nieh presented the delays encountered as benchmarks, Tolia et al. sought to connect the delays they measured to the ultimate usability of the applications tested as provisioned. They did this by binning the response times according to usability guidelines derived from the human factors research described in the previous section. From these heuristics, the frequency of "unacceptable" delays across the network conditions was estimated. Importantly, no actual user study was conducted to verify these heuristics. Instead, estimates of acceptable delays in GIMP photo editing software were drawn from guidelines that were previously derived from text-entry-based applications [33].

These three studies highlight the need for additional research into the impacts of system response delays on usability. The human factors research from the previous section clearly indicates the importance of system response delays as a strong driver of user satisfaction. However, no work that we are aware of has examined how modern GUI interaction techniques are affected by delays.

In order to make reasonable decisions based on the findings of this type of systems research, the effects of delays need to be explored.

2.3 VM access techniques

How to best provision applications over wide-area networks is an active area of research. For this study, we have focused on two techniques, thick and thin client models, which represent the edge cases for application delivery. In this section, we will discuss the two options we have chosen and why we did not explore other approaches.

2.3.1 Thin Clients

Thin clients are a widely accepted method for accessing remote computing resources. The concept of thin clients has existed since the early days of modern computing, in which computational cycles are provided by powerful computers managed in a centralized fashion. In such environments, client machines on the user end have minimal functionalities necessary for accessing the remote resources, which saves the costs of hardware and maintenance. Today, thin clients play a major role in cloud environments that host VMs, often in the form of client software rather than dedicated hardware consoles.

Virtual Network Computing (VNC [28]) is a popular example of a thin client and the implementation we chose to use. In the case of VNC, the client sends user input such as mouse movements and key presses to the remote VM, and the VM sends back screen updates as a result of processing the received user input. Although various optimizations have been applied, especially for communicating visual output to the client, carrying out most of the computation on the remote site means that the usability of thin clients is primarily affected by the *latency* of communications between the client and remote machine. Each command input by the user needs to be propagated to the VM, and the result needs to be transmitted back. This round-trip communication time, *i.e.*, the latency, is involved in every interactive event handled by the VM.

2.3.2 Thick Clients

VM-based thick clients are another attractive solution to cloud-sourced application delivery. Here, the idea is to transfer software, encapsulated in VMs for universal execution, to client machines. In contrast to thin clients, all computation is performed locally on the client side, while remote servers stream the state of the VMs. Thus, all user interaction is handled locally without involving network communication. Such an approach is becoming increasingly practical, as even mobile devices now have considerable computing power.

For VM-based thick clients, the key constraint is the network bandwidth, which dictates the time needed to transfer the necessary parts of the VM state. Once the VM is ready, local execution ensures interactive usability of the delivered software unless it requires additional state retrieval. This is in clear contrast to thin clients, which typically have moderate demands for bandwidth and instead require low latency for good interactivity. This difference highlights an inherent performance trade-off made by VM-based thick clients. They deliver near-local execution performance for the price of wait time for state caching. Previous work has proposed efficient

solutions for minimizing the cost of VM state retrieval, for example in the forms of VM distribution networks [25, 27, 26] and virtual desktops [5, 16, 30, 32, 31]. In particular, the thick client implementation that we use, vTube [1], focuses on cloud-sourced virtual appliance delivery with the goal of being able to rapidly launch a VM.

2.3.3 Hybrid Systems

Thick and Thin clients represent the two end cases of remote access in which all processing is done remotely (thin client) and all processing is done locally (thick client). Many mobile-cloud applications use a hybrid approach, offloading some computation and processing some locally. There are two main reasons we did not explore a hybrid delivery method. The first is that such methods require modifications to application source code that are not necessary for thick or thin clients. The second is that determining how to distribute processing in hybrid systems is itself a non-trivial optimization problem. Given the number of factors already involved in our study, we leave the exploration of hybrid systems to future work.

3 Experiment Design

While it is not possible to isolate and test every factor that impacts a user experience, our goal in this study was to explore the broad contours of several factors. In particular, we were interested in how network conditions, application level interface designs, and VM provisioning techniques combine to impact software performance. In this section, we will discuss the variables we choose to manipulate to examine these factors as well as the choices we made to limit other factors.

3.1 Experiment Variables

With four controlled variables (application, VM provisioning technique, network bandwidth, and network latency) the number of combinations to test is huge. Based on feedback from pilot studies, we determined that users could adequately gauge the usability of the system and complete a task in 5-10 minutes. We set a time limit of one hour for each user session to avoid fatigue, allowing each user to perform approximately seven trials with time for evaluations. To make the study more tractable, we opted to treat focus on one application at a time, ultimately running three separate studies.

3.2 Application Choices

Our goals in selecting the applications were as follows: 1) We wanted to mitigate learning effects and novelty from the experiment. 2) We wanted a clear method for evaluating user performance across the conditions 3) We wanted discrete tasks that could be effectively but not exactly repeated across conditions. 4) We wanted tasks that could be completed for all conditions in a reasonable amount of time. 5) We wanted a diverse set of UI interactions. 6) We needed applications with a large enough footprint so as not to be trivially fetched in the thick client case.

Given these criteria, the applications and tasks we selected were document-editing using Microsoft Word 2007, a series of photo editing tutorials using Adobe Photoshop CS6, and a simple game, Bejeweled Twist.

3.3 Application Provisioning

For the thin client model we used an implementation of VNC [28] and for the thick client model we used an implementation of vTube [1]. As this is an initial study, we leave an exploration of hybrid delivery models and network fluctuations (dynamic changes in latency and bandwidth) to future work.

3.4 Network Conditions

For each participant, one trial was performed with an application running locally to provide a baseline experience. Thus we had six trials for users to experience different access techniques (thick and thin client), network latencies, and network bandwidths. To fit these constraints, we made a choice to isolate network parameters by access mode. Thus, thick client access trials had a fixed network latency and thin client access trials had a fixed network bandwith.

3.4.1 Latency Invariance of Thick Client Delivery

In the simplest implementation of a thick client, the user would wait until the entire VM application image is downloaded before execution begins. For this approach, the user experience would be an upfront wait dictated primarily by network bandwidth and the size of the VM state. As long as network latency is on the order of a fraction of a second, it will play a negligible role in the delay for any reasonably large application (hundreds of MB to a few GB) even with high bandwidth (tens of Mbps). As discussed previously, the thick client model vTube reduces the up front wait time by beginning execution after a portion of VM state is downloaded. As long as the user's actions are within the scope of vTube's collected traces, chunks of the VM state will be prefetched in an efficient manner and latency will be negligible relative to reasonable bandwidths and image sizes. It is important to note that, as implemented in our study, this assumption only holds when user actions are well-modeled. If user behavior significantly deviates from the model, the VM may begin fetching state in an inefficient manner and network latencies will have a larger impact on the user experience. This case did in fact arise during our study and will be discussed in further detail. However, it can be avoided in principle by better training of the vTube prefetch knowledge.

To illustrate the extent to which latency limitedly impacts vTube, Table 2 shows a summary of VM streaming events in our example traces under varied network conditions. The sessions each correspond to 3 to 4 minutes of Microsoft Word usage, and they retrieved approximately 490 MB of VM state. Also, the application usage behavior remained within the coverage of the traces that generated the algorithm's prefetch knowledge. As table 2 shows, vTube triggers initial buffering lasting several minutes, with minor additional buffering afterwards. During the buffering periods, VM state is pushed to the client exploiting as much bandwidth as available, yet it still takes minutes to prefetch the initial data. Relative to this, network latencies contribute orders of magnitude less to the total experienced delay, as exemplified by the first two cases with varied latency. In addition, VM state misses, which do not exploit bandwidth as efficiently and are therefore more directly

Table 2: Breakdown of vTube events

Network	Event	Statistics
14.4 Mbps 240 ms	Initial Buffering	135.1 sec.
	Additional Buffering	3.0 sec.
	Memory Misses	0.08%
	Disk Misses	0.03%
14.4 Mbps 30 ms	Initial Buffering	134.2 sec.
	Additional Buffering	0.84 sec.
	Memory Misses	0.02%
	Disk Misses	0.03%
7.2 Mbps 240 ms	Initial Buffering	276.6 sec.
	Additional Buffering	3.4 sec.
	Memory Misses	0.22%
	Disk Misses	0.03%

impacted by latency, account for a tiny fraction of all state accesses. These observations indicate that bandwidth is the primary performance factor in vTube (i.e., thick client). The last case with half the bandwidth of the first case confirms this fact, showing initial buffering twice as long. Note that when the user exhibits outlying behavior outside the scope of the algorithm's coverage, VM state misses could increase considerably, with VM performance accordingly degraded. In such a case, the latency involved in each miss does affect the performance.

3.4.2 Bandwidth Invariance of Thin Client Delivery

In a thin client delivery model, user interactions are sent to the host machine and screen updates are returned to the client. While these exchanges do require some amount of bandwidth, the requirements are typically achievable given existing WAN conditions [2]. For example, while monitoring our experimental setup, we typically observed network traffic exceeding a few Mbps only during fullscreen refreshes (e.g. when a new window opened onscreen). Additionally, many approaches have been developed to compress the information being transmitted to and from the host and client.

However, even the smallest amount of information must travel to and from the host computer, thus incurring a full round trip cost of network latency between every input and response. This latency is a persistent feature of the thin client model.

3.4.3 Network Settings

Given these characteristics, we chose to give our thin client mode a high bandwidth of 100 Mbps (to emulate highly efficient compression) and our thick client mode the highest network latency tested in the thin client mode. The precise values used in our user study, listed in Table 3, were selected to be representative of realistic network conditions. For latency, we targeted a range spanning that of wired and wireless connections, having 3G/4G LTE connectivity in mind. For bandwidth, we picked values based on the nationwide average in the U.S. [19]. As we will discuss later, we

ensured during pilot studies that these selected parameters had observably different impact on the user experience.

Table 3: Network Settings per Delivery Mode

Delivery Mode	**Network Latency**	**Bandwidth**
Thick Client	240 ms	7.2/14.4/28.8 Mbps
Thin Client	30/60/120/240 ms	100 Mbps

3.5 Pilot Study

Before conducting the user study, we ran pilot studies with students in our department. The purpose of these pilots was to 1) identify any need to modify or correct user tasks, 2) ensure that network parameters resulted in perceptually different system performances, and 3) test the administration procedure described in the next section 3.6. During the pilot studies, we caught errors and ambiguities in the task instructions, which were corrected in the final versions. We verified that a range of latencies and bandwidth values derived from real-world measurements, as summarized in Table 3, led to visibly different levels of user experience. Finally, we confirmed that we were able to switch between different system settings and complete each study within the targeted time frame. Additional findings relevant to particular applications will be discussed in the corresponding application sections.

3.6 Application Platforms

In order to evaluate the impact of latency and bandwidth on different VM application provisioning techniques, we had to select a set of representative applications and tasks. The need for finely controlled network settings compelled us to implement the study in a LAN-connected desktop computing environment.

3.7 System Set-up

Figure 2 shows the system set-up for Thick Client, Thin Client, and local execution respectively. In each case, the applications are run in a Windows 7 virtual environment executed on a modified version of QEMU-KVM on Ubuntu 12.04. This setup allows the flexibility to run different virtualized operating systems, which is a necessary requirement for distributing unmodified applications, while also allowing us to record low level system data such as keystrokes. In the thick client mode, the local machine starts without the Windows 7 virtual environment, which begins to prefetch necessary VM state upon startup. In the thin client mode, the Windows 7 virtual environment is hosted on the remote machine and the local machine executes a VNC client. This VNC client sends user input to the remote VM, and receives screen updates it generates. For the local execution case, a Windows 7 virtual environment is preloaded onto the local machine eliminating the need for communication with the remote machine.

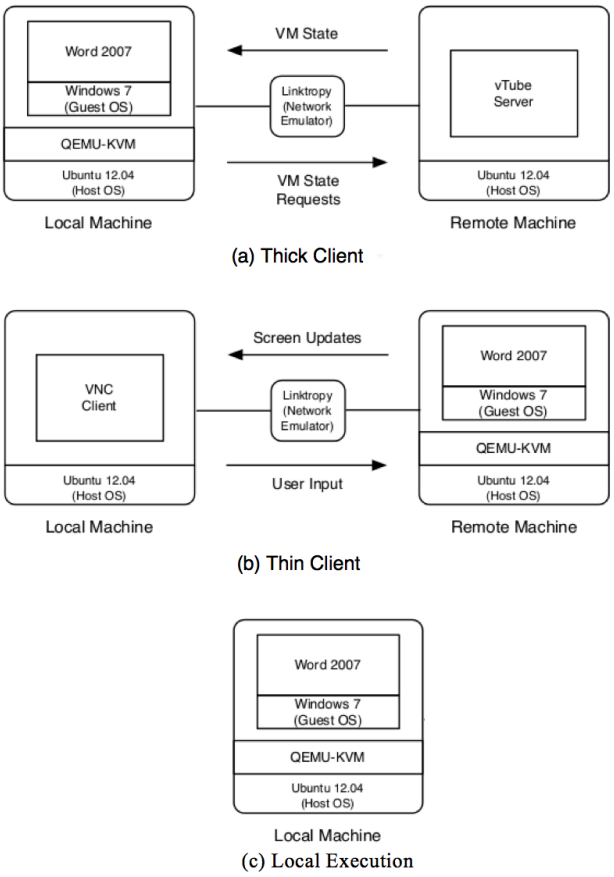

Figure 2: System Setup for (a) Thick Client Provisioning and (b) Thin Client Provisioning (c) Local Execution

The local and remote machines are equipped with an 8-core Intel Core i7 CPU at 3.40 GHz and 32 GB of memory. Network latency and bandwidth are manipulated by the network emulator, Linktropy [3], with dedicated hardware and web interface controls.

3.8 Data Recording

We instrumented the system with various functionalities in order to collect traces for postmortem analyses. These include user input traces, VM streaming logs, and screen captures. User input traces record timestamped mouse button presses, cursor movements, and key presses by the user. They reveal useful information such as the use of short-cut keys. VM streaming logs keep track of detailed low-level information, such as buffering duration, timed VM state accesses, and cache hits and misses. Screen captures dump the current screen of the VM as a JPEG image every 10 seconds. They augment information about user interaction collected by input traces, and help understand user behavior at a high level.

3.9 Administration

Depending on the task, instruction set and study condition being tested with a user, we manage system aspects in the following manner. First, all the VM images are loaded into memory to eliminate overhead associated with undue disk accesses. Then, for each of the seven sessions per subject, network latency and bandwidth are specified with the emulator's user interface. For validation purposes, we record server-side logs of effective bandwidth for VM streaming, and take screenshots of emulator settings for double-checking. Next, we launch a script that starts a VM either locally, by vTube, or via VNC. During VM use, various traces are collected as mentioned above. Finally, we saved edited documents, again for validation purposes.

3.10 User Tasks

For each application, we designed a set of tasks that would keep the users engaged and require them to complete a set of specified interactions. We ran a pilot study for each application to verify the feasibility of our task designs and made modifications where necessary.

3.10.1 Word Document Editing Tasks

The final Word sets of instructions each consisted of a total of 11 distinct steps (e.g., "Use the Find function", "Insert [photo.jpg] into the document"). We created seven sets of instructions, corresponding to seven trial conditions, each with unique documents. The instruction sets each included the same steps, but the ordering was randomized where possible (e.g., adding captions always followed inserting an image, saving the final version of the document was always the last instruction).

We placed a 10-minute time limit on each trial to ensure an endpoint. During our pilot studies this time limit was not a factor as the average task completion time was near five minutes and no one had difficulty completing the set of seven tasks in one hour. However, during the actual study, nearly half of the participants failed to complete at least one trial before the 10-minute limit. We will discuss this in further detail in the results section.

3.10.2 Photoshop Tutorial Tasks

We originally tried to develop a repetitive set of instructions similar to the Microsoft Word tasks for the Photoshop trial. However, during the pilot study, we found that many participants were unfamiliar with Photoshop's functionalities and simply could not complete a varied set of tasks within the allotted time. To accommodate novice users, we created seven tutorials that stepped users through different functions such as the color range tool and motion blur filters.

Other adjustments had to be made for the Photoshop trials as well. Whereas most users were able to complete all seven Word tasks within one hour despite the 10-minute trial time limits, we found we needed to reduce the Photoshop time limits to seven minutes to ensure that all seven trials could be completed. The Photoshop application also required more VM Image state to be prefetched, leaving insufficient time to interact with the application in the 7.2 Mbps condition. Thus, we opted to run the thick client provisioning trials with Bandwidths of 28.8 Mbps and 14.4 Mbps.

3.10.3 Bejeweled Game Tasks

Unlike the other applications, the Bejeweled game did not have a specific task to accomplish. Instead, users were instructed to continue playing and trying to achieve as high a score as possible in the allotted time of seven minutes. If users lost the game, their score was recorded and they were instructed to begin a new game.

Table 4: A summary of Trial Variables

Application:	Microsoft Word	Bejeweled Twist	Adobe Photoshop
Trial Time Limit:	10 minutes	7 minutes	7 minutes
Task:	11 step editing task	Continuous gameplay	Function tutorials
Bandwidths:	7.2 Mbps / 14.4 Mbps	7.2 Mbps / 14.4 Mbps	14.4 Mbps / 28.8 Mbps
Latencies:	30/60/120/240 ms	30/60/120/240 ms	30/60/120/240 ms
Number of Participants:	36	12	12

3.11 Test Procedures

Participants were recruited through an online database for research studies. They needed to be between the ages of 18 and 65 and were requested to have familiarity with the applications being used. Participants were informed of the study objectives and asked to complete a short survey about their experience with relevant software upon arrival.

After reading over and signing consent forms, users were then presented with a local, fully cached version of an application (either Word, Photoshop or Bejeweled). For Word and Photoshop conditions, users were also given a set of instructions. Users were informed that this was a locally executed version to which the following trials' performances should be compared. They were given as much time as necessary and encouraged to ask for explanation if any task was confusing or unfamiliar. Following the completion of the task, participants were asked to complete a survey based on a subset of NASA-TLX dimensions in which they rated the following four questions on a 5-point Likert scale.

- How would you rate your satisfaction with the system performance during this trial?
- To what degree did you experience at least one of these feelings: insecure, discouraged, irritated, stressed or annoyed?
- How mentally demanding was the task during this trial?
- How well were you able to accomplish the task?

There were also two free response questions about the task and system performance, listed below.

- Was there any aspect of the system performance that stood out in your mind? Please explain?

- Was there any aspect of the instructed task that stood out in your mind? Please explain?

After the local execution mode, each user was presented with each of the six test conditions listed in Table 4 in counter-balance order. The order of the instruction sets and accompanying documents was randomized across the users. Users were instructed to associate any glitch, freezing or crashing they experienced with the test condition and fill out the following surveys accordingly. Once all the tasks were completed or users were alerted of the time limit by a pop-up window, users were asked to fill out the survey about that particular trial while the experience was fresh in their minds.

4 Results

This study was designed to achieve two goals. First, we wanted to measure the impact of system response delays on the user experience of applications using modern graphic user interfaces to compare the results with previous human factors research relying solely on text interface applications. Secondly, we wanted to compare the relative level of user satisfaction between thick and thin client provisioning under network conditions currently available to wide area mobile networks.

While the different applications offered different approaches to measuring user performance, the surveys completed after each task were consistent across applications and with prior studies of this kind. In this section, we will present the collective results of these surveys. For all statistical analysis of survey responses, 5-point Likert scale responses were numerically coded such that very low = 1, low = 2, medium = 3, high = 4, and very high = 5. Throughout this paper statistical significance is determined at the 5% significance level.

4.1 User satisfaction

The participants' ratings of their satisfaction with the system performance across all manipulations and applications are shown in Figure 3.

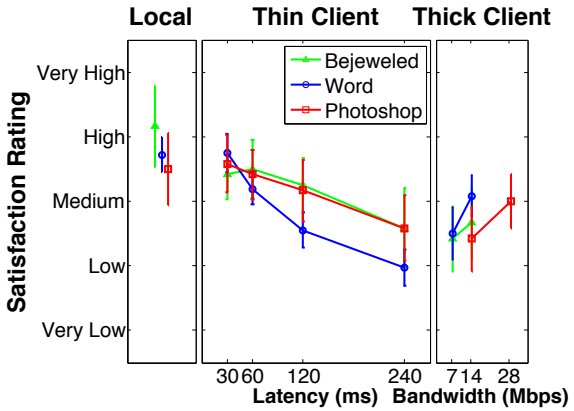

Figure 3: Satisfaction with system performance ratings averaged by trial condition and application with confidence intervals.

The results follow the generally expected trends. Local execution is rated as having a relatively high performance. Thin client trials have higher performance for lower latencies. For thick client delivery, the higher bandwidth setting is always rated as having better performance, despite variations across applications. These trends fit the intuitive expectations. However, variations across applications and settings provide interesting insights. These findings will be highlighted in the following sections.

4.2 Feelings of Annoyance

The participants' ratings of their feelings of annoyance invoked across all manipulations and applications are shown in Figure 4. The results during the local execution seem intuitive, with the game, Bejeweled, resulting in the lowest level of annoyance and the less familiar application, Photoshop, resulting in the highest level of annoyance. However, these trends did not hold across all trials, as Microsoft Word quickly became more frustrating as even relatively low levels of latency were introduced during thin client trials. The relatively high levels of frustration during thick client trials of Bejeweled may be a result of the other applications have task instructions which the participants could read over during the initial buffering.

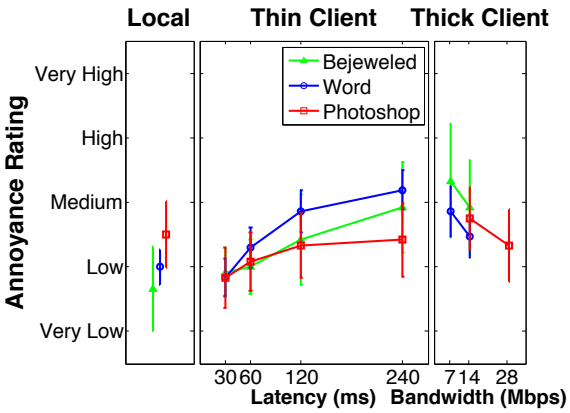

Figure 4: Feelings of Annoyance ratings averaged by trial condition and application with confidence intervals.

4.3 Mental Demands

The participants' ratings of mental demands invoked by the tasks are shown in Figure 5. Only the Microsoft Word trials, which had three times as many participants displayed any clear trend. Even so, the difference between highest and lowest levels of induced mental demand was considerable less than the differences in report performance or frustration.

4.4 Sense of Accomplishment

The participants' ratings of their sense of accomplishment during the tasks are shown in Figure 6. The low sense of accomplishment reported in the Photoshop local execution trial is indicative of a

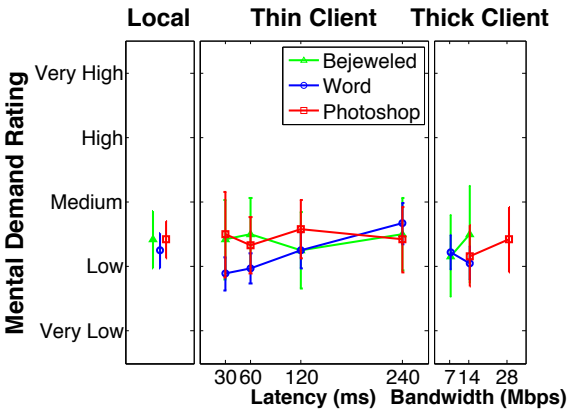

Figure 5: Ratings of Mental Demand averaged by trial condition and application with confidence intervals.

general unfamiliarity with Photoshop amongst the participants.

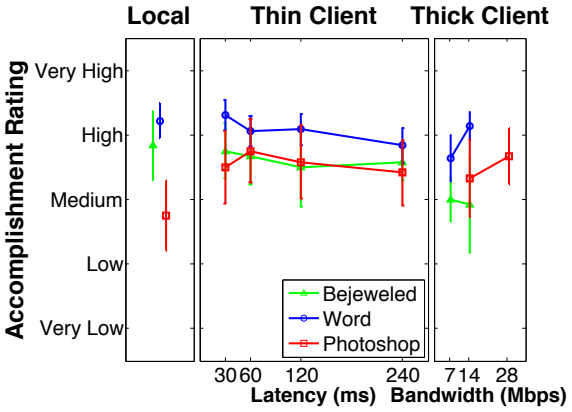

Figure 6: Sense of Accomplishment ratings averaged by trial condition and application with confidence intervals.

5 Discussion

In this section we will focus on variations in responses across applications and settings. We will also present findings from the free response portions of the survey that provide interesting insights into how application and system level parameters impact the user experience.

5.1 Animation is Highly Latency Sensitive

Of the three applications, only Bejeweled was shown to have a significant difference in perceived performance between local execution and 30 ms Thin Client delivery (see Table 5). After review-

ing the free form responses, it became clear that the ever-present animations during Bejeweled's gameplay led to a lower tolerance of relatively low latencies. One user specifically mentioned "visual artifacts degraded [the] overall image for the game", while rating the 30ms trial as having lower than the local execution. For Word and Photoshop, with their default static displays, this was not an issue.

Table 5: Comparison between Local Execution and 30ms Thin Client provisioning across applications. (* Indicates a significant difference.)

	Local Execution	**Thin Client, 30ms**
Bejeweled	4.0667*	3.4
Word	3.722	3.74
Photoshop	3.5	3.483

5.2 Interaction Types are Important

During thin client provisioning trials, no significant difference in perceived performance was observed between Bejeweled and Photoshop for any latency. However, the rated performance of Microsoft Word dropped off significantly more than the other applications as latency increased (see Figure 3, center plot).

As noted previously, participants playing Bejeweled noticed low levels of latency due to visual artifacts that occurred during animations. However, additional increases in latency did not drastically decrease the perceived system performance. As one participant noted after playing Bejeweled with 240ms of lag, "[There was] an increased lag between mouse and pointer, with more intense lag coming from animations, **but [it] had a very low impact on game play**". Similarly, participants using Photoshop tended to make note of latency increases without being bothered by them. Another user reported that during the 240ms lag Photoshop trial, "the mouse cursor kept lagging, but it did not bother me too much." In contrast, participants using Microsoft Word tended to be bothered by increases in latency much more and reported lower levels of system performance accordingly. For example, participant 5 did not appear to notice the 30ms lag, offering no comments and rating the performance as highly as the local execution case. However, at 60ms lag, P5 noted, "it lagged when I was scrolling" and rated the performance lower and annoyance level higher.

The differences indicate the importance of specific interaction types in understanding how the user experience will be impacted by system delays. In Bejeweled, low levels of latency were noticed due to the animation, but the gameplay UI consisted only of clicking relatively large stationary game icons. The Word tasks, on the other hand, required more varied user actions. Typing and menu operations have distinctly different characteristics than scrolling and text selection. Scrolling, which involves rapid, large updates of screen space tended to bring about noticeable visual artifacts and as latency increased, frustration surrounding text selections was commonly noted. Some users went as far as to modify their behavior as lag increased. Participant 2 noted after the Thin Client 240 ms delay trial, that "because of the lag, I found myself changing how I did a task. Rather than scroll to the headers, I was using Ctrl+F".

The Photoshop tasks that users performed were typically whole picture manipulations performed via menu selections. Thus, the users did not need to pan, zoom, or make pixel-level image adjustments. We suspect that if our Photoshop tasks had required such precise manipulations, the reported performance versus latency curve may have dropped off more steeply. This is certainly a point worth examining further with a more advanced group of Photoshop users.

5.3 VM State Size is Important

Thick client provisioning inevitably face some amount of start-up delay as application state is prefetched from the server. This delay is directly related to the size of the relevant sections of the VM state and the network bandwidth available. While approaches such as the vTube implementation we employed can reduce the required level of VM state needed to initially access the application, there is no way to entirely eliminate the delay.

As can be seen in the far right section of Figure 3, this prefetching delay comes at a cost to the user experience. It is important to note that the values we tested were based on the average of measured bandwidth across the US [19]. This choice was made deliberately to allow comparisons between the provisioning techniques for network conditions that are readily available today. An alternative approach could be taken to find the bandwidths at which thick client provisioning performance is rated equivalently to local execution. Given that the performance impact of thick client models is primarily that of a startup delay that can be estimated for known VM sizes and network bandwidths, we left such explorations to future work.

While it is not surprising that the performance ratings for the thick client conditions were significantly lower than the local execution case, we did not anticipate the Thick Client ratings would have much more variance than other conditions. The free form responses after Thick Client trials also differed qualitatively from the Thin Client cases. While the upfront buffer time was nearly universally acknowledged, some participants did not object to the delay. One user, during the 7.2Mbps Photoshop noted, "the slower operating program gives more time [while buffering] to read instructions". This seems to indicate that individual level factors like temperament may play a bigger factor in how users respond to upfront delays than other system response characteristics.

5.4 Task Completion Matters

As mentioned in the 'Word Document Editing Tasks' section, we designed the Word tasks to be wholly completed in the allotted 10 minutes. However, it turned out that our pilot study population (drawn from computer science university students) was likely biased towards expert Word users. As a result, not every participant using Microsoft Word was able to complete every task in the allotted trial time. We took care to ensure that this was not an issue for the other applications.

Overall, during thin client delivery trials, 98% of the participants completed the tasks. For the thick client delivery with 14.4 Mbps bandwidth, which exhibited a typical upfront wait of 2 minutes, 92% of users completed the tasks. For the 7.2 Mbps case, though, with a typical upfront wait of 4 minutes, only 53% of participants were able to fully complete the task. Table 6 lists the completion rate and variance in reported system performance for Microsoft Word trials with different network settings. We can see that the conditions with the lower completion rate had a much higher standard deviation in the performance rating.

Table 6: Task completion rates and Performance Rating variations for participants using Microsoft Word. The lower bandwidth Thick Client condition resulted in a far lower completion rate and much higher variation in Performance Ratings.

	Completion %	**Performance Rating Variance**
Thin Client (all)	98% (141 of 144)	$\leq .91$
Thick Client, 14.4 Mbps	92% (33 of 36)	.94
Thick Clients, 7.2 Mbps	53% (19 of 36)	1.51

While the split between participants who completed the task and those who did not may limit our ability to directly compare between Thick and Thin client delivery methods, it does allow us to examine the impact of task completion upon participants' perception of system performance. In Table 7, we compare the performance ratings of thick client conditions between participants who completed the tasks and those who did not. While the sample size is very limited and the results are not statistically significant, participants who completed the task during the low bandwidth trials rated the system performance higher than users who failed to complete the trial during high bandwidth trials.

Table 7: Reported levels of system performance and sense of accomplishment for Thick Client Microsoft Word trials separated by whether or not the user completed all the steps in the task in the allotted time.

	Performance		**Accomplishment**	
	Complete	Incomplete	Complete	Incomplete
7.2 Mbps	2.84 (N=19)	2.12 (N=17)	4.11 (N=19)	3.12 (N=17)
14.4 Mbps	3.12 (N=33)	2.67 (N=3)	4.21 (N=33)	3.33 (N=3)

5.5 Future Work

There is still substantial work to be done before VM provisioning will truly rival the user experience of local execution. To completely characterize how user experience relates to thin and thick client models, many more factors would need to be considered.

One choice we made was to employ constant latency and bandwidth values. Obviously, this is not representative of real world networks. There is evidence that variations in system response times play a significant role in the perceived satisfaction. A constant delay that is longer than the average of a random delay has been shown to be preferable in some circumstances [10]. The degree to which this might impact the perceived performance of thin client models for the variations typically experienced in real-world network conditions is an open question.

While these parameters could each be systematically explored through user studies similar to ours, it's difficult to truly understand how users will respond to a system without a large-scale deployment. Robert Miller aptly described this problem nearly 50 years ago when he wrote that "different human purposes and actions will have different acceptable or useful response times" [21].

This sentiment applies well to the difficulty in trying to precisely translate a participant's tolerance for poor performance when they are committed to an hour long lab study into how they will respond to the same system performance in the real world.

Finally, there is room to improve the prefetching and caching strategies in the thick client models. While vTube demonstrates the advantages of intelligent prefetching strategies, it is still a developing approach. During our study, there were examples of participants who behaved in ways that were outside of the scope of vTube's prefetch knowledge. For example, one user preferred to open the image files outside of Word and then copy them into the document. This behavior was not anticipated by vTube and effectively froze the system. Better approaches to training or methods of recovery could help to alleviate these problems.

6 Conclusion

Our findings are inline with previous literature in demonstrating the adverse effects of delays in systems response on user satisfaction. However, it is also clear from our study that a single system response metric does not adequately capture the complexity of how delays affect the user experience of modern applications. Even within a single application, choices about specific interface actions (e.g. keyboard shortcuts vs. clicking icons) can be impacted by delays introduced elsewhere in the system.

For designers seeking to provision applications over the internet there is likely not a single set of network parameters that will provide acceptable performance across all applications. Instead, developers will need to consider the bottlenecks of the network (e.g. bandwidth, latency) and the characteristics of the application when choosing their provisioning strategies.

Given the network parameters provided by recent measures, we found that Thin Client provisioning tends to have a higher level of performance than Thick Client provisioning. However, as network bandwidths continue to increase, this may not prove true in the long term. Additionally, understanding how the tradeoffs between thick and thin client approaches impact usability over a longer timeframe will require additional, longer term studies.

Acknowledgements

This work was supported by the Alfred P. Sloan Foundation, the Quality of Life Technologies ERC (NSF-EEC-0540865), the Vodafone Group, and by the National Science Foundation under grant number IIS-1065336. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and should not be attributed to Carnegie Mellon University or the funding sources.

References

[1] Y. Abe, R. Geambasu, K. Joshi, H. A. Lagar-Cavilla, and M. Satyanarayanan. vTube: Efficient streaming of virtual appliances over last-mile networks. In *Proceedings of the 4th Annual Symposium on Cloud Computing, SoCC 2013*, 2013.

[2] Akamai. State of the Internet, Third Quarter 2013, 2013. http://www.akamai.com/dl/akamai/akamai-soti-q313.pdf.

[3] Apposite Technologies. Linktropy 5500 Hardware Guide. www.apposite-tech.com/assets/pdfs/linktropy-5500-hwguide.pdf.

[4] T. Butler. Computer Response Time and User Performance. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, (December):58–62, 1983.

[5] R. Chandra, N. Zeldovich, C. Sapuntzakis, and M. Lam. The Collective: A Cache-Based System Management Architecture. *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 259–272, 2005.

[6] L. Columbus. Roundup Of Cloud Computing Forecasts And Market Estimates, 2014. http://www.forbes.com/sites/louiscolumbus/2014/03/14/roundup-of-cloud-computing-forecasts-and-market-estimates-2014/.

[7] J. R. Dabrowski and E. V. Munson. Is 100 Milliseconds Too Fast? *CHI '01 extended abstracts on Human factors in computing systems - CHI '01*, page 317, 2001.

[8] G. L. Dannenbring. The effect of computer response time on user performance and satisfaction: A preliminary investigation. *Behavior Research Methods & Instrumentation*, 15(2):213–216, Mar. 1983.

[9] S. R. Ellis, K. Mania, B. D. Adelstein, and M. I. Hill. Generalizeability of Latency Detection in a Variety of Virtual Environments. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 48(23):2632–2636, 2004.

[10] R. Geist, R. Allen, and R. Nowaczyk. Towards a model of user perception of computer systems response time. *ACM SIGCHI Bulletin*, 17(SI):249–253, 1986.

[11] T. J. Goodman and R. Spence. The Effect of Computer System Response Time Variability on Interactive Graphical Problem Solving, 1981.

[12] M. Grossberg, R. A. Wiesen, and D. B. Yntema. Experiment on Problem Sovling with Delayed Computer Responses. In *IEEE Transactions on Systems, Man and Cybernetics*, volume SMC-6, pages 219–222, 1976.

[13] J. Guynes. Impact of system response time on state anxiety. *Communications of the ACM*, 31(3):342–347, 1988.

[14] J. A. Hoxmeier and C. DiCesare. System Response Time and User Satisfaction : An Experimental Study of Browser-based Applications. *Proceedings of the Association of Information Systems Americas Conference*, (2):1–26, 2000.

[15] R. Jota, A. Ng, P. Dietz, and D. Wigdor. How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks. *Chi '13*, (1):2291–2300, 2013.

[16] M. Kozuch and M. Satyanarayanan. Internet suspend /resume. In *Proc. Workshop on Mobile Computing Systems and Applications (MCSA)*, page 40, 2002.

[17] W. Kuhmann. Experimental investigation of stress-inducing properties of system response times. *Ergonomics*, 32(3):271–280, Mar. 1989.

[18] A. M. Lai and J. Nieh. On the performance of wide-area thin-client computing. *ACM Transactions on Computer Systems*, 24(2):175–209, May 2006.

[19] F. Lardinois. Akamai: Average U.S. Internet Speed Up 28% YoY, Now At 7.4 Mbps, But South Korea, Japan And Hong Kong Still Far Ahead, 2013. http://techcrunch.com/2013/04/23/akamai-average-u-s-internet-speed-up-28-yoy-now-at-7-4-mbps-but-south-korea-japan-and-hong-kong-still-far-ahead/.

[20] M. Meehan, S. Razzaque, M. Whitton, and F. P. Brooks. Effect of Latency on Presence in Stressful Virtual Environments. In *Proc. of the IEEE Virtual Reality*, volume 21, pages 141–148, 2003.

[21] R. B. Miller. Response time in man-computer conversational transactions. *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, page 267, 1968.

[22] F. F.-H. Nah. A study on tolerable waiting time: how long are Web users willing to wait?, 2004.

[23] A. Ng, J. Lepinski, and D. Wigdor. Designing for low-latency direct-touch input. *Proceedings of the 25th . . .*, pages 453–464, 2012.

[24] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[25] C. Peng. VDN : Virtual Machine Image Distribution Network for Cloud Data Centers. In *Proc. IEEE INFOCOM*, pages 181–189, 2012.

[26] J. Reich, O. Laadan, and E. Brosh. VMTorrent: Scalable P2P Virtual Machine Streaming. *Proceedings of CoNEXT*, pages 289–300, 2012.

[27] J. Reich, O. Laadan, E. Brosh, A. Sherman, V. Misra, and J. Nieh. VMTorrent : Virtual Appliances On-Demand. *ACM SIGCOMM Computer Communication Review*, 41(4):453–454, 2010.

[28] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[29] A. Rushinek and S. F. Rushinek. What makes users happy? *Communications of the ACM*, 29(7):594–598, July 1986.

[30] C. P. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. S. Lam, and M. Rosenblum. Virtual Appliances for Deploying and Maintaining Software. *LISA*, 3:181–194, 2003.

[31] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Base Cloudlets in Mobile Computing. *Pervasive Computing*, 8(4):14–23, 2009.

[32] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, A. Surie, D. R. O'Hallaron, A. Wolbach, J. Harkes, A. Perrig, D. J. Farber, M. a. Kozuch, C. J. Helfrich, P. Nath, and H. A. Lagar-Cavilla. Pervasive personal computing in an Internet Suspend/Resume system. *IEEE Internet Computing*, 11(2):16–25, 2007.

[33] N. Tolia, D. G. Andersen, and M. Satyanarayanan. Quantifying interactive user experience on thin clients. *Computer*, 39(3):46–52, 2006.