Quantifying Interactive User Experience on Thin Clients

Niraj Tolia, David G. Andersen, and M. Satyanarayanan
Carnegie Mellon University

The adequacy of thin-client computing is highly variable and depends on both the application and the available network quality. For intensely interactive applications, a crisp user experience may be hard to guarantee. An alternative—stateless thick clients—preserves many of the benefits of thin-client computing but eliminates its acute sensitivity to network latency.

After a few false starts in the past decade, thin-client computing is finally gaining serious attention and acceptance among large and medium-size companies. For example, recent *Wall Street Journal* articles (17 Jan. 2005, 3 Feb. 2005) predicted that more than 3 million enterprise desktops, amounting to 10 percent of the market, will be thin clients by 2008. These articles also mention Time-Warner, Wal-Mart, and the Pentagon as examples of enterprises that are adopting thin clients in significant numbers. Microsoft is also reported to be close to releasing a stripped-down version of Windows XP that transforms an old PC into a thin client (www.brianmadden.com; 18 April 2005).

A thin client consists of a display, keyboard, and mouse combined with sufficient processing power and memory for graphical rendering and network communication with a compute server using a specialized protocol. All application and operating system code is executed on the server. The client has no long-term user state and needs no disk. A standard PC can be made to function as a thin client through software such as Virtual Network Computing (VNC).[1]

We describe an approach to quantifying the impact of network latency on interactive response and show that the adequacy of thin-client computing is highly variable and depends on both the application and available network quality. If near ideal network conditions (low latency and high bandwidth) can be guaranteed, thin clients offer a good computing experience. As network quality degrades, interactive performance suffers.

It is latency—not bandwidth—that is the greater challenge. Tightly coupled tasks such as graphics editing suffer more than loosely coupled tasks such as Web browsing. The combination of worst anticipated network quality and most tightly coupled tasks will determine whether a thin-client approach is satisfactory for an organization.

Stateless thick clients, an alternative to thin-client computing, preserve many of the benefits of thin-client computing but eliminate its acute sensitivity to network latency. This alternative approach achieves this improvement by asynchronously transferring more runtime state to a client and executing from that state on a local processor.

TIME-SHARING REDUX

In his 1983 classic, "Hints for Computer System Design,"[2] Butler Lampson offered the following quote from Jim Morris: "The nicest thing about the Alto is that it doesn't run faster at night." This quote succinctly captures the joy of a time-sharing user who has just discovered the crisp, unvarying performance of a personal computer on highly interactive tasks. Gone is the pain of sluggish interactive response during periods of peak load. By colocating a processor with each user, personal computing offers a computing experience that is unaffected by the actions of other users.

Published by the IEEE Computer Society
0018-9162/06/$20.00 © 2006 IEEE

Low I/O latency and high display bandwidth make possible the tight user-machine coupling of today's GUIs and interactive applications. Never having experienced the frustration of poor interactive response, most users today cannot relate firsthand to the Morris quote. It is just a relic of an archaic world, long gone and never to return.

Alas, history has come full circle. Our collective time-sharing experience is relevant to thin-client computing. The similarity arises from the fact that even trivial user-machine interactions incur queuing delay on a resource that is outside a user's control.

In both cases, queuing delay is acutely sensitive to the vagaries of the external computing environment. In time-sharing, the shared resource is the processor. In thin-client computing, it is the network. Thin clients can incur additional queuing delay at a shared compute server. For example, some proposed designs use a small set of blade servers for a large number of thin clients. Server queuing delay can be eliminated by dedicating a compute server per client, but network queuing delay cannot be eliminated. It is intrinsic to the thin-client model.

Trivial interactions suffer the full queuing delay of the network, yet they must meet the user's highest expectation of performance.

WHY THIN CLIENTS ARE ATTRACTIVE

Two distinct factors motivate interest in thin clients. The first is concentration of personal computing state. In large organizations, the physical dispersion of personal computing hardware complicates system administration. Isolating an infected machine, forcing certain security upgrades, or restarting a crashed machine are examples of actions that typically require physical access to the hardware. Concentrating all relevant state in compute servers simplifies this physical access. Rather than walking from machine to machine, access is available at the system administrator's fingertips in a server room.

These considerations are especially relevant at enterprise scale, where the total cost of ownership of personal computers is of growing concern. As hardware costs plummet, an increasing fraction of the total lifetime cost of owning a personal computer goes to its ongoing maintenance rather than to its initial purchase. Thin clients offer the possibility that concentration of state will lead to reduced total cost of ownership.

The second reason for interest in thin clients is user mobility. A user can authenticate at any thin client and have immediate access to a unique computing environment. This thin-client anonymity harkens back to time-sharing, where a user could log in at any dumb terminal. It enhances collaboration and spontaneity and simplifies the logistics of hardware deployment.

INTERACTIVE RESPONSE

What performance goals should thin-client computing strive for so that users will embrace it? The most critical performance measure is the crispness of interactive response. For example, when a user presses a mouse button, she expects the popup menu to appear with no perceptible delay; in freehand drawing, she expects the onscreen curve to track her mouse movements with no lag; when enlarging or shrinking an object, she expects the onscreen rubber-banding effect to smoothly and precisely track her mouse. This is the standard of interactive performance today, and users are loath to settle for less.

Notice that it is the trivial interactions that are the most challenging for thin clients. Such interactions involve end-to-end communication—from user to application code and back to user—but negligible delay occurs within the application itself. These interactions suffer the full queuing delay of the network, yet they must meet the user's highest expectation of performance. This is in contrast, for example, to a click on a Web link, in which case the user is already anticipating a download delay.

Over a 40-year period, the HCI community has built up a substantial body of knowledge about the impact of interactive response times on user satisfaction and task productivity.[3-7] From these studies, a broad consensus has emerged on acceptable response times for trivial interactions:

- User productivity is not impacted by response times below 150 milliseconds. This is therefore a good quantitative definition of crisp response.
- In the range from 150 ms to one second, users become increasingly aware of response time. They strongly prefer response times well below one second.
- Above one second, users become unhappy. When forced, users can adapt to response times over one second, but this is accompanied by frustration with the system and a drop in productivity.

Good response time is the key to overall satisfaction with an interactive session. User anxiety is positively correlated with poor response time. Also relevant is the finding from psychology and economics that negative experiences have much greater impact than positive experiences on judgment and risk taking.[8] The implication for thin clients is that poor response time incidents will be overweighted in users' memories. Even a few sluggish interactions in an otherwise acceptable interactive session may be sufficient to turn off a user.

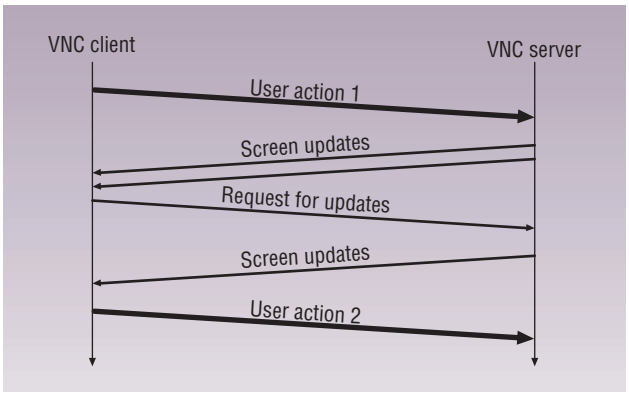

Figure 1. Packet trace simulation. The simulation assumes that screen updates following action 1 are causally related to action 1 and are not causally related to the intervening request for updates. During replay on a slower network, the last screen update could be sent before the request for update, improving the interactive performance of the simulated VNC system.

THIN-CLIENT PEFORMANCE

How well do thin clients perform under different network conditions, and for what kinds of applications are they best suited? To answer these questions, we apply HCI guidelines to a set of three application traces under a variety of network conditions to determine how often each application's response time is crisp, annoying, or unacceptable.

Data collection

To model user activity, we used the Xnee record and replay tool to capture three different user input traces.[9] These traces capture the keyboard and mouse actions of a user working with a photo-editing application, a presentation creator, and a word processor.

We replayed these user input traces on a VNC thin client[1] connected to the VNC server on a 100-Mbps switched Ethernet network with submillisecond round-trip time. The server exported a 1600 × 1200 desktop with a depth of 16 bits per pixel. VNC was set to use hextile compression for screen updates. During replay, an intermediary machine captured the packet traces for all client-server interaction. All machines were 3.2 GHz hyperthreaded Pentium 4s with 2 Gbytes of SDRAM and ran the Fedora Core 2 2.6.10-1.770-SMP Linux kernel.

We analyzed the packet traces to identify the VNC protocol packets containing client-user input and screen-update requests and the replies from the server.

Simulation and analytical method

The packet traces drove a simulator—built using *ns*-2—that had a high-level understanding of the VNC protocol. The simulator deterministically modeled the impact of different network characteristics on user performance.

Our approach is similar to slow-motion benchmarking.[10] Both systems gather operation response time by extracting the intervals between causally related groups of packets. While the data collection method differs, the end results from the capture are similar. A slow-motion study emphasizes a comparative analysis of different thin-client systems at different bandwidths and looks at noninteractive applications such as Web page loads and video replay. Our study instead focuses on how applications with varying degrees of interactivity interact with both bandwidth and latency.

Feeding decoded protocol messages into a network simulator let us explore a wider set of network conditions. Here, we present only the results of 10- and 100-Mbps networks with a low round-trip latency of 1ms, a moderate latency of 20 ms, and high latencies of 66 and 100 ms. These latencies represent the delays a user might experience when using a compute server located on the same network, the same geographic region, across the country in the best case, and across the country in the common case.

The simulator preserves quality. It does not drop updates on slower networks because doing so could also degrade the user experience, but it batches updates more aggressively than the real VNC system might. This batching ensures that the simulated performance meets or exceeds that of the real system, making an aggressive assumption in favor of thin clients.

The packet trace records the user action packets, the client processing time due to the user's action, and the subsequent series of screen updates and requests for updates from the client. As Figure 1 shows, these requests each release additional screen updates from the server. When simulating a slower network, an incoming request for updates releases all screen updates that were ready on the server.

We define an operation as beginning at the time the client sends a user action and terminating with the last screen update from the server before a subsequent user action. Each operation captures a single user event such as a keystroke or mouse click. The analysis only counts the response time for user actions that generate a screen update. Based upon our HCI research, we place response times into one of several bins:

- Crisp: < 150 ms;
- Noticeable to annoying: 150 ms to one second;
- Annoying: one to two seconds;
- Unacceptable: two to five seconds; and
- Unusable: > five seconds.

Trace descriptions

To exhibit the impact of latency on different interactivity requirements, we chose four different applications for our study. The applications ranged from a highly interactive image manipulation program to moderately

interactive use of a word processor.

The GNU Image Manipulation Program, similar to Adobe Photoshop, is a popular application for photo editing. The GIMP trace captured an experienced user separating headshots from a group photograph, removing red eyes from another image, and sharpening another blurry image. The trace was about five minutes long.

Impress, similar to Microsoft PowerPoint, is part of the OpenOffice suite. This trace captured a user creating a new slide for a presentation. Examples of user actions captured included importing external images, creation of various shapes on the slide and the addition of text to them, addition of animation, and previewing and tweaking the animation. The trace was about 10 minutes long.

Writer, similar to Microsoft Word, is the word processor in the OpenOffice suite. To distinguish between different usage scenarios, this trace was divided into two parts. One part consisted of a user transcribing a page of text into Writer. The other part captured the actions of tracking changes made to a large document and either accepting or rejecting these changes. The combined duration of this trace was approximately nine minutes.

Table 1. Operation response time at 100 Mbps.

Application	RTT	Crisp < 150 ms	Noticeable 150 ms - 1 sec	Annoying 1 - 2 sec	Unacceptable 2 - 5 sec	Unusable > 5 sec
GIMP	1 ms	3,278	40	0	0	0
	20 ms	3,214	82	4	18	0
	66 ms	2,710	572	12	3	21
	100 ms	2,296	973	20	6	23
Impress	1 ms	5,879	15	1	0	0
	20 ms	5,855	37	2	1	0
	66 ms	5,621	269	2	1	2
	100 ms	5,520	363	9	1	2
Writer: Tracking changes	1 ms	270	15	0	0	0
	20 ms	254	29	2	0	0
	66 ms	201	82	0	0	2
	100 ms	182	99	2	0	2
Writer: Typing	1 ms	2,911	56	0	0	0
	20 ms	2,909	56	2	0	0
	66 ms	2,899	66	0	0	2
	100 ms	2,887	77	1	0	2

Table 2. Operation response time at 10 Mbps.

Application	RTT	Crisp < 150 ms	Noticeable 150 ms - 1 sec	Annoying 1 - 2 sec	Unacceptable 2 - 5 sec	Unusable > 5 sec
GIMP	1 ms	3,244	51	3	20	0
	20 ms	3,197	98	3	20	0
	66 ms	2,552	730	12	3	21
	100 ms	2,129	1140	20	6	23
Impress	1 ms	5,869	23	1	2	0
	20 ms	5,848	44	1	2	0
	66 ms	5,614	276	2	1	2
	100 ms	5,514	368	10	1	2
Writer: Tracking changes	1 ms	267	16	1	1	0
	20 ms	250	33	1	1	2
	66 ms	196	87	0	0	2
	100 ms	181	100	2	0	2
Writer: Typing	1 ms	2,909	56	1	1	0
	20 ms	2,909	56	1	1	0
	66 ms	2,899	66	0	0	2
	100 ms	2,887	77	1	0	2

Results

The interactive performance of a thin-client setup varies greatly with two parameters: the degree of interactivity of the application and the latency of the network. Highly interactive applications such as GIMP perform poorly with even moderate latency. High latency impairs all but the simplest applications.

Table 1 shows the number of operations at 100 Mbps that fell into each time bin for the four applications. The table should be read in two directions—within an application as latency increases and, for a given latency, between applications.

Table 2 shows the same data for a 10-Mbps network. The similarity of the two tables suggests that, above 10

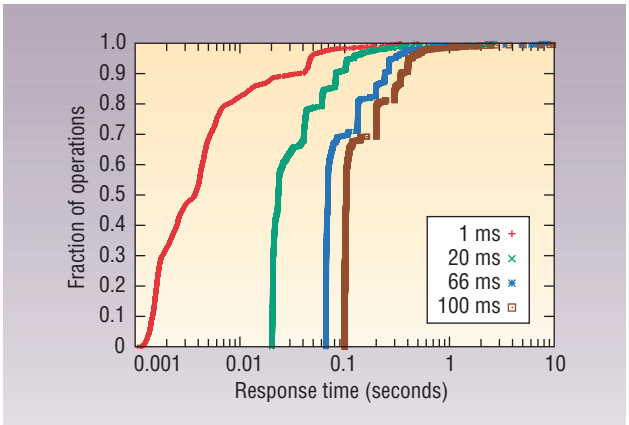

Figure 2. GIMP operations. As latency increases, the mean response time remains low, but the number of operations above the annoying and unacceptable thresholds increases dramatically.

Mbps, bandwidth does not limit thin-client performance for these applications.

The data in Table 1 shows that on a 100-Mbps network with 1 ms latency, all of the traced applications perform well. Few operations require more than one second to complete, and most complete before the 150 ms threshold. This finding agrees with that of previous work[11] and with our subjective observation that VNC performed very well on a local network.

Figure 2 shows, however, that as latency increases, the mean response time remains low, but the number of operations above the annoying and unacceptable thresholds increases dramatically.

GIMP was the most demanding application. When run over a 100-Mbps network with 100 ms of round-trip latency, 29 percent of the operations required more than 150 ms to complete, and 1.5 percent required more than 1 second, compared to 1.2 percent and zero percent, respectively, with a 1-ms round-trip latency. These figures also matched our experience running the application over such a network: Response time was poor, and at times, the lag was severe enough to make using the program difficult.

The application mix also makes a significant difference in perceived performance: Certain applications appear much better suited for thin-client computing than others. The columns in Table 1 show how the performance changes between applications. Typing in a text editor, for instance, involves only small screen updates; the performance barely changes between the 1 ms case (56 operations >150 ms) and the 100 ms case (80 operations >150 ms). The presentation creation script occupies a middle ground, while the GIMP and tracking-changes traces update large sections of the screen. These applications with tightly coupled interaction cycles perform measurably worse at high latencies.

Discussion

After an organization determines acceptable limits of network latency and bandwidth for its unique mix of interactive applications, it must ensure that network quality never falls below those limits. Adding bandwidth is relatively easy, but reducing latency is much harder. Additionally, network jitter has the potential to further worsen the problem. User studies have shown that jitter leads to a further decrease in productivity and an increase in user errors.[12]

In addition to physical-layer transmission delays and end-to-end software path lengths, technologies such as firewalls, virtual private networks and other overlay networks, and lossy wireless networks add latency and other hurdles. While our simulation does not model these factors, their inclusion would have further biased our results against thin clients. Quantitative knowledge of acceptable latency and bandwidth provides a clear and unambiguous performance target for networks that must support thin clients. It can also be useful in negotiating service-level agreements with network providers.

STATELESS THICK CLIENTS

User growth, network evolution, and introduction of new applications are common events in the IT infrastructure of any large organization. An organization that uses thin clients must pay careful attention to the network latency and bandwidth impact of such events. This implies greater attention to system management than is apparent from a first glance at thin-client computing. It is not yet known how this will impact the hoped-for reduction in total cost of ownership associated with using thin clients.

Is there a more robust solution? Is it possible to achieve performance benefits without the brittleness of a thin-client solution? Virtual machine technology enables clean encapsulation of a user's entire personal computing state. A distributed file system can store VM state and deliver it on demand to any thick client. This approach trades off start-up delay for crisp interaction: Once execution begins, all interaction is local.

Since VM state corresponds to a user's entire personal computer—including the OS, applications, and user files—clients are stateless from the viewpoint of long-term user state. Such clients offer the same benefits of state concentration and user mobility as thin clients. Work on Internet Suspend/Resume[13] and by Constantine Sapuntzakis and colleagues[14] has confirmed the feasibility of *stateless thick clients*. In contrast, the SoulPad work[15] uses portable storage rather than network storage to implement thick clients.

ASYNCHRONOUS NETWORK DEPENDENCE

A thick-client approach assumes that clients have sufficient resources (disk, CPU, and so forth) to run the desired applications locally. The stateless thick-client

approach further depends on high network bandwidth for VM state transfer. This approach is predicated on the assumption that shipping a lot of bits is simpler and cheaper than sustaining the management attention and system administration discipline needed to maintain individual systems or to use more frugal but more complex abstractions such as process migration. Logical state transfer does not, however, always mean immediate or complete physical state transfer. Clever policies can give the illusion of complete VM state transfer while transferring much less data.

The issue of network availability is distinct from concerns about the volume of data transferred. Once it fully caches the state, a stateless thick client does not require the network to be available. Disconnected operation capability in the underlying storage system can provide the illusion of connectivity. The client buffers the updates and eventually reintegrates them when network connectivity is restored. Such operation can be advantageous, for example, for a laptop user who must spend time in an area with poor or no connectivity.

The stateless thick client approach is thus asynchronous in its network dependence. Its performance, even under a pure demand-fetch policy, is insensitive to network latency, although it is sensitive to network bandwidth. The approach requires connectivity while fetching state and during eventual reintegration. For extended periods between these two events (possibly lasting many hours), total disconnection is acceptable and has absolutely no performance impact. With a stateless thick client, a laptop user can remain mobile and productive.

In contrast, thin clients have synchronous network dependence. The network quality must be sufficient at all times for crisp interactive response. Note that it is the worst case—not the average case—that determines whether a thin-client approach will be satisfactory. An organization that adopts thin-client computing must also invest in system management resources to ensure adequate network quality at all times for its most demanding interactive tasks. By definition, disconnected operation is impossible with thin clients.

Crisp computing response is taken for granted today. The building blocks of modern-day GUIs such as scrolling, highlighting, and popup menus are built upon this key assumption. Over time, we have learned how to use these building blocks to create applications that are well-matched to human cognition. These applications, ranging from spreadsheets and word processors to CAD tools and medical imaging aids, are the backbone of personal computing. Based on the assumption of excellent interactive response, an entire infrastructure of OSs, GUIs, applications, user expectations, and practices has evolved over two decades.

Thin clients pose an inadvertent threat to this world. Born of frustration with high total cost of ownership, interest in thin clients is very high. Unfortunately, dependence on thin clients could hurt the hard-won goal of crisp response, particularly for highly interactive applications.

An alternative approach is to continue using thick clients but to encapsulate user state using VM technology and deliver it on demand from servers. We are exploring a hybrid that starts with using a thin client to mask state transfer delay and then switches to VM execution for crisp response.

Regardless of the best final solution, developers must take care to ensure a good user experience for demanding interactive applications under adverse network conditions.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under grant CNS-0509004. The views and conclusions in this article are those of the authors and do not represent the opinions, express or implied, of the US government, the NSF, or Carnegie Mellon University.

References

1. T. Richardson et al., "Virtual Network Computing," *IEEE Internet Computing*, vol. 2, no. 1, 1998, pp. 33-38.
2. B.W. Lampson, "Hints for Computer System Design," *Proc. 9th ACM Symp. Operating Systems Principles*, ACM Press, 1983, pp. 33-48.
3. J.L. Guynes, "Impact of System Response Time on State Anxiety," *Comm. ACM*, vol. 31, no. 3, 1988, pp. 342-347.
4. G.L. Martin and K.G. Corl, "System Response Time Effects on User Productivity," *Behaviour and Information Technology,* vol. 5, no. 1, 1986, pp. 3-13.
5. R.B. Miller, "Response Time in Man-Computer Conversational Transactions," *Proc. AFIPS Fall Joint Computer Conf.*, AFIPS Press, 1968, pp. 267-277.
6. A. Rushinek and S.F. Rushinek, "What Makes Users Happy?" *Comm. ACM,* vol. 29, no. 7, pp. 594-598.
7. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd ed., Addison-Wesley, 1997.
8. D. Kahneman, "Maps of Bounded Rationality: A Perspective on Intuitive Judgment and Choice," *Nobel Prizes, 2002*, Almqvist & Wiksell Int'l, 2003.
9. H. Sandklef, "Testing Applications with Xnee," *Linux J.*, Jan. 2004, p. 5.
10. J. Nieh, S.J. Yang, and N. Novik, "Measuring Thin-Client Performance Using Slow-Motion Benchmarking," *ACM Trans. Computer Systems*, vol. 21, no. 1, 2003, pp. 87-115.
11. A. Lai and J. Nieh, "Limits of Wide-Area Thin-Client Computing," *Proc. 2002 ACM Sigmetrics Int'l Conf. Measure-*

ment and Modeling of Computer Systems, ACM Press, 2002, pp. 228-239.
12. C. Gutwin, "The Effects of Network Delays on Group Work in Real-Time Groupware," *Proc. 7th European Conf. Computer-Supported Cooperative Workshop*, Kluwer, 2001, pp. 299-318.
13. M. Kozuch and M. Satyanarayanan, "Internet Suspend/Resume," *Proc. 4th IEEE Workshop Mobile Computing Systems and Applications*, IEEE Press, 2002, pp. 40-46.
14. C. Sapuntzakis et al., "Optimizing the Migration of Virtual Computers," *Proc. 5th Symp. Operating Systems Design and Implementation*, Usenix, 2002, pp. 377-390.
15. R. Caceres et al., "Reincarnating PCs with Portable SoulPads," *Proc. 3rd Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 2005), ACM Press, 2005, pp. 65-78.

Niraj Tolia *is a graduate student in electrical and computer engineering at Carnegie Mellon University, where he received an MS in electrical and computer engineering. His research interests include content-addressable storage systems and virtualization. He is a member of the ACM and Usenix. Contact him at ntolia@cmu.edu.*

David G. Andersen *is an assistant professor in the Computer Science Department at Carnegie Mellon University. His research interests focus on computer systems in networked environments. He received a PhD in computer science from MIT. He is a member of the ACM, Usenix, and the IEEE. Contact him at dga@cs.cmu.edu.*

M. Satyanarayanan *is the Carnegie Group Professor of Computer Science at Carnegie Mellon University. His research interests span mobile computing, pervasive computing, and distributed systems. Satyanarayanan received a PhD in computer science from Carnegie Mellon University. He is a member of the IEEE Computer Society and a Fellow of the ACM and the IEEE. He was the founding director of Intel Research Pittsburgh. Contact him at satya@cs.cmu.edu.*