Latency Perception in Cloud-Based Workspaces and Environments

By Adam Burke and Ricardo Figueroa

Abstract

This study investigates the user experience in cloud-based environments. As a result of the Coronavirus Disease 2019 (COVID-19) restrictions, we were not able to execute the experiments as initially planned and hence this study is being presented as a pilot experiment that can be used as a prototype in future research. The aim of these experiments is to find a metric that can be used in estimating the performance of remote desktop systems. This study mainly focuses on latency, within and across networks, as the primary factor in the remote-desktop user experience.

Keyword

Cloud, IP, latency, PCoIP, remote desktop environments

Introduction

Cloud computing has been increasingly used of late in all industries, including the motion picture and video industry. This offers inexpensive, scalable computing resources when a particular user or organization requires these resources. However, when the same resources are not needed, these can easily be reallocated to other users or organizations. These dynamic resources allow for economic scalability, where the customer only pays for what is needed at that exact moment. Along with this technology, virtual desktops, also referred to as virtual machines (VMs), have gained popularity for their flexibility and ease of distribution. However, this technology can hinder user experience as a result of latency between the client and the server. This can have increased importance for end users working in the media industry. This study investigates a number of use cases and provides prototypes for gathering both objective and subjective metrics. The aim of this study is to identify potential metrics and thresholds related to the user experience, which can be used when implementing cloud-based systems.

Cloud computing, also known as "the cloud," can be thought of as nothing more than a large data center full of servers and computers. The terms "public cloud" and "private cloud" are commonly used to differentiate between data centers that serve multiple clients (also known as multiple companies) and a single client (also known as a single company), respectively. Private clouds are generally maintained by the clients themselves and have seen decreasing use in recent years, as a result of the economic advantages using a public cloud provider. These advantages come from the ability to scale cloud services in addition to not having to employ an entire team to maintain a data center. Using remote cloud services can also be highly beneficial in areas where the rent is high.

Background

Cloud computing, also known as "the cloud," can be thought of as nothing more than a large data center full of servers and computers. The terms "public cloud" and "private cloud" are commonly used to differentiate between data centers that serve multiple clients (also known as multiple companies) and a single client (also known as a single company), respectively. Private clouds are generally maintained by the clients themselves and have seen decreasing use in recent years, as a result of the economic advantages of using a public cloud provider. These advantages come from the ability to scale cloud services in addition to not having to employ an entire team to maintain a data center. Using remote cloud services can also be highly beneficial in areas where the rent is high. Unless otherwise noted, this study focuses on public cloud service providers, as the services they provide are most applicable to the research of this writing. There are a number of cloud service providers, the largest being Amazon Web Services (AWSs), in addition to Google Cloud Platform (GCP), Microsoft Azure, and IBM Cloud. They offer a similar set of tools, including machine learning applications, web hosting, and desktop workspaces, for cloud computing to fit the need of the clients' use case. The focus will be on desktop workspaces, as they can have a significant impact on end-user interaction and user experience.

Digital Object Identifier 10.5594/JMI.2021.3076568
Date of publication: 3 June 2021

Private Infrastructure

The internet has facilitated more efficient communication, allowing for motion picture and television productions to become increasingly globalized and less centralized. With this, the transition to digital data pipelines is increasingly more popular. The combination of these two technologies makes it easier to share production assets, without physically transporting film, and so on. However, this technology also has limitations when sending files over public internet due to data size. This can be particularly problematic as data passes through the last mile of the network—the section of the network that physically connects to the end users' home or office.

Cloud providers invest heavily in building their cloud infrastructure, strategically placing data centers and network infrastructure around the world. **Figure 1** shows the map of AWS's global cloud footprint. AWS has a number of regions across the world, which are all connected to one another by the AWS private network infrastructure. These low latency and redundant networks consist of at least one pair of parallel 100 Gb/s data paths, including trans-Atlantic and trans-Pacific links. This allows for fast, efficient data transfer around the world, without relying on public internet protocol (IP) infrastructure. This is highly efficient and reliable when working with large data files.

Consider a production filming in New York City working with a visual effects (VFX) house in Los Angeles. If the production wanted to send the footage back to Los Angeles, they could either ship a physical hard drive across the country or send the data files over the internet. However, if the original high-quality camera files (raw) were needed, this could be challenging from a data-size perspective, as a result of last-mile network bottlenecks. Traditionally, these files would not only have to be uploaded to a server, but also downloaded to the VFX house's infrastructure. This can be problematic if, say, the VFX house is limited to a 250Mb/s last-mile connection.

This last-mile connection can be partially eliminated using cloud infrastructure. Once the files have been uploaded to the cloud, the large files can be moved across regions (to different parts of the country or the world) using the private infrastructure described previously, bypassing unpredictable public IP networks, as shown in **Figure 2**. In addition, these files do not have to be redownloaded; the VFX work can take place "in the cloud," by leveraging cloud-based desktop workstations. An additional benefit of this workflow is that the VFX house does not need to have local storage infrastructure to support the files; the files are merely accessed through a virtual-desktop connection using a client computer. It should also be noted that AWS supports direct connections from companies to its private network infrastructure. This can provide even faster access to data in the cloud. An additional benefit of storing data in the cloud and working on it remotely is that it can be easily accessed by other cloud-based processes that may have already been launched. These could include machine learning, content distribution, transcoding, and other tasks.

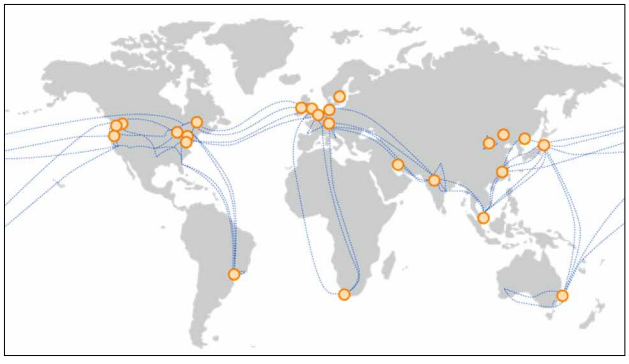

FIGURE 1. AWS Global Cloud Infrastructure. Orange dots represent regions, whereas blue dashed lines represent network infrastructure.[1]

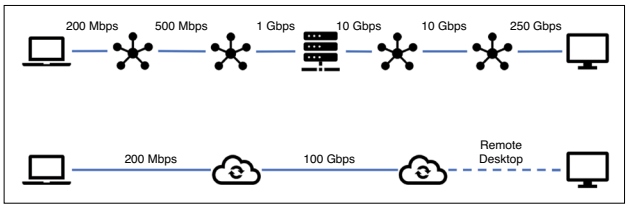

FIGURE 2. Illustration of the data path in a public IP environment (top) versus cloud IP infrastructure (bottom). Notice that the cloud infrastructure can provide a more robust and direct path.

Work From (Almost) Anywhere

When leveraging virtual environments, cheap, lightweight, or low-powered client computers or tablets can be used to access the scalable workspaces that can be created in the cloud. This can allow lightweight portable computers to have the same advantages as a desktop workstation and can provide numerous advantages for traveling and working on location.

When employees are hired or leave the company, their workspaces can quickly be created or decommissioned. This also has the advantage of allowing employees to have multiple "computers" without the need to carry multiple devices. Users may also have different machines with different configurations for specific projects (potentially running different operating systems), adding an extra layer of security. In the event that the client device is lost, stolen, or damaged, the VM can be accessed through a different device without data loss. The lost or stolen client device can then be prevented from accessing the cloud workspace, ensuring no data is compromised.

Virtual Desktop Protocols

A number of virtual desktops are currently on the market. Some readers may be familiar with the virtual network connection (VNC) protocol, which takes screenshots on the host computer and sends them to the client. Other protocols send instruction sets to the client computer and rely on the client to render the graphics

(such as X11 Tunneling). However, these protocols are rather old and do not necessarily offer the best user experience. Today, a number of protocols exist and are promoted by cloud providers. These include:

- Teradici's "PCoIP" protocol (PC-over-IP)
- Citrix's "Citrix Workspaces"
- AWS's "AWS Workspace Client."

These protocols are optimized for cloud workspaces and are designed for high-performance graphics, such as media production. Teradici has both hardware and software solutions for both the host and client sides. The host hardware is a peripheral component interconnect (PCI) Express card, allowing for the host computer to read universal serial bus (USB) and other peripherals remotely. The host computer sees this card as a display, as it is physically connected to the host via a display connection [High-Definition Multimedia Interface (HDMI)/Display Port]. The benefit of the card acting as a display is that all compression and encryption is completed on the card and not on the host's central processing unit (CPU). The PCoIP zero client acts as a "plug-and-play" client, allowing for peripherals to be easily attached, without having to forward them through a host operating system. This takes advantage of the Tera2 host card's PCI Express. **Figure 3** shows the generic connection schematic of a PCoIP host card configuration.

Due to the proprietary nature of the PCoIP protocol, there is not much publicly available information with regard to the protocol's structure. However, one of the interesting advantages of the PCoIP protocol is that it leverages User Datagram Protocol (UDP) at the transport layer[2] as opposed to Transmission Control Protocol (TCP), which implements error checking and recovery services. UDP looks to deliver as much information as quickly as possible and forgoes error checking.[3] In other words, UDP continuously sends streams of information regardless of whether or not the client received the packets. It should be noted that PCoIP does use TCP for other services such as the initial connection handshake, but uses UDP for the frame transmission. Teradici promotes the use of UDP as being more effective and quicker when handling media resources.

Ping

The most common way of measuring round-trip latency (RTL) between computers on and across networks is using the "ping" networking utility. Versions of this open-source and public utility can be found on all major operating systems (maybe with a few exceptions), including

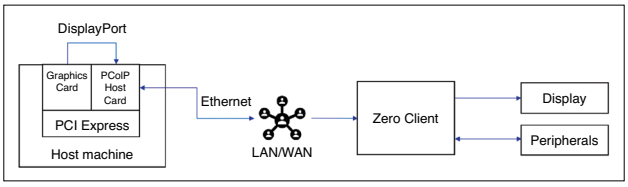

FIGURE 3. Generalized PCoIP implementation using a PCoIP remote workstation card and a zero client.

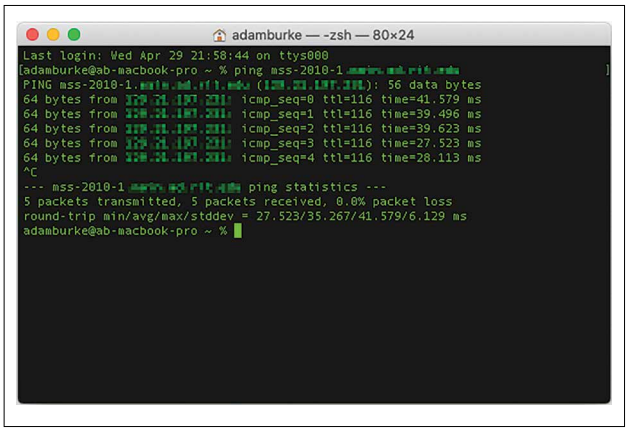

FIGURE 4. Ping network utility running on MacOS in terminal. Note that the average RTL is approximately 35 ms. IP addresses and hostnames have been redacted.

Microsoft Windows, Apple's MacOS, and many Linux distributions. This utility takes advantage of internet control message protocol (ICMP) Echo requests, where a packet of bites is sent to a specified computer. The recipient then replies to the original computer, echoing the request, from which the original sender can calculate the round-trip time (or latency) within or across networks. **Figure 4** shows an example ping from MacOS to an Rochester Institute of Technology (RIT) lab. It should be noted that some firewalls are configured to ignore ICMP Echo requests to help prevent denial-of-service attacks.

Impacts of Latency in Cloud Workspaces

As described in the previous sections, cloud computing can have significant advantages in transferring and storing data. However, the impact of working on a machine, which could be geographically hundreds of miles away, has not been thoroughly studied, particularly in media applications such as motion pictures and television. As mentioned in the previous section, latency can be defined as the amount of time it takes for an input to be perceived as an output; a simplified flow can be seen in **Figure 5**. Therefore, internet packets that

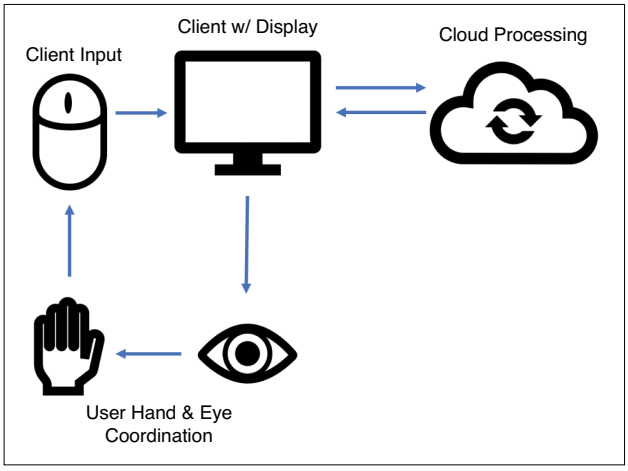

FIGURE 5. Simplified illustration showing input to computation pathway where latency can be accumulated.

travel farther or travel slower generate greater latency. However, it should be noted that latency is accumulated on both the host and the client machines. Although, the greatest source of latency in cloud workflows can often be attributed to transport between the client and the host.

Previous Research

Studies have produced a broad range of reported values for the human ability to detect latency. One study reported that the minimal perceivable latency rests somewhere between 2 and 100 ms. This broad range is primarily contributed to the various form factors of input/output (IO) devices.[4] A paper published by NVIDIA discussed the effects of display refresh rate on the performance of competitive, pro-esports. They reported that humans experience a sensorimotor delay between 150 and 200 ms.[5]

Application and IO play major roles in detecting latencies as detailed in the aforementioned paper.[4] This is important to consider, as there are many IO variations for media applications. For example, editorial often only uses a mouse, keyboard, and display. In comparison, animators often use pen displays such as the Wacom Cintiq, and VFX artists often use pen tablets such as Wacom Tablets. These various input devices can be broken down into direct and indirect inputs. Pen displays and touchscreens fall into the direct input category as the user is physically touching a user interface (UI) element, and so on. Indirect input devices consist of devices such as keyboards and mice, where the users are moving the mouse while looking at the screen in front of them. Research has shown that users are more sensitive to latency when using direct input devices than they are with an indirect input device. Likewise, the type of interaction can also make a difference in the perception of latency. Dragging or tracking motions are much more susceptible to perceived latency than tapping actions, such as pressing buttons.[4,5] An interesting way to think about this is if users are moving their fingers at a rate of 10 cm/s with a 50-ms delay, a displacement of 5 mm can be seen between the UI object and the user's finger.[6] However, this displacement would not be noticeable with an indirect input device such as a mouse; there is no visual reference as to where the UI object should be at that moment in time. Some readers may have experienced this when using cheaper touch screens that have high latency. For these reasons, the type of work that a cloud workstation is being used for can have an effect on the users' perceived tolerance.

Goals

The goal of this study was to establish a metric that can be used to determine whether users have an acceptable maximum latency between their client and server to effectively complete their tasks and work. In other words, the latency between the client and the host is under a specific point on a scale of latencies. Once established, this "latency score" can be applied to implementing cloud workstations, determining whether a particular latency is low enough for an acceptable user experience.

Methods—The Initial Plan

To establish a metric and scale for determining the user experience under a specific latency condition, both objective and subjective data collection methods were used. However, due to the COVID-19 pandemic, these initial methods were used to conduct a pilot study instead, which can be used as a starting point for future research. Read the section titled "Methods—The Backup," as the subjective methods were adapted for the pilot study.

In simplified terms, the objective and subjective experiments looked to answer these respective questions:

- *Objective*: At what point can an increase in latency be detected from a baseline?
- *Subjective*: At what point does the latency introduced via the network become unacceptable?

Objective Data

First, objective data collection through the use of a psychophysical forechoice experiment determines the just-noticeable-difference (JND) in latency a user perceives. The experiment consists of a series of trials where participants are subjected to a baseline of zero milliseconds ($RTL_0 = 0$ ms) and a second latency condition of increased latency $RTL_n = RTL_0 + \Delta RTL_n$ a sequential order. The order in which RTL_0 and RTL_n are presented is randomized between trials, so that the probability of each being presented first is $p = 0.5$. Under each RTL condition (RTL_n), the participants are asked to complete a particular task, such as rotoscoping, editing, animation, and so on. The participants are pretrained to accomplish this task under a baseline environment, that is, participants are sourced from classes where a specific skill is taught. They are then asked which condition (RTL_0 or RTL_n) provided a better user experience. **Figure 6** shows the testing sequence graphically.

Subjective Data

Once a reliable JND is determined, participants are then asked to rate the "usability and performance" of a particular latency condition on a Likert scale, while performing a particular task once again (rotoscoping, editing, animation, etc.). Zero on the Likert scale is considered the baseline and is guided by the JND found in the prior objective test. After being subjected

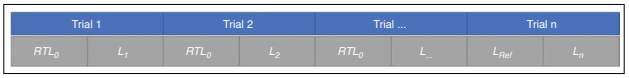

FIGURE 6. Objective testing diagram. Note: although RTL_0 is shown in this diagram as always coming first, in reality the order of the latency states is randomized.

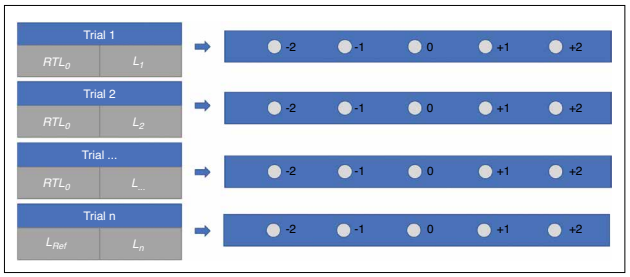

FIGURE 7. Subjective testing diagram. Note: although RTL_0 is shown in this diagram as always coming first, in reality the order of the latency states is randomized.

to a particular latency condition, participants have the option to select plus or minus two steps on the Likert scale rating, where "+2" represents the user experience is much better than the baseline and "−2" represents the user experience is much worse than the baseline. **Figure 7** shows this graphically.

Testing Setup

To collect precise and accurate results, a highly controllable, repeatable, and reliable testbed is needed. Additionally, to ensure there is no nonexperiment-related traffic that could influence results, the testbed is built on its own local network that is separate from all other networks. A high-level diagram of the testbed configuration is shown in **Fig. 8.** A more detailed schematic is shown in **Fig. 9**.

Testbed Hardware

For the prototype experiments, the testbed consisted of a local computer, running on Windows 10, that was

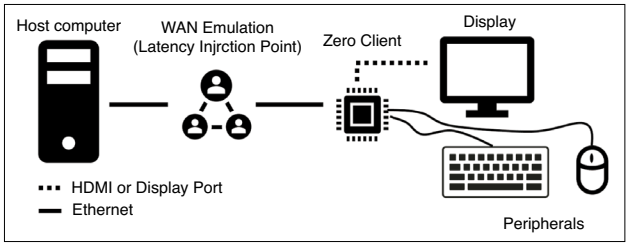

FIGURE 8. Diagram of testing configuration with general-purpose peripherals.

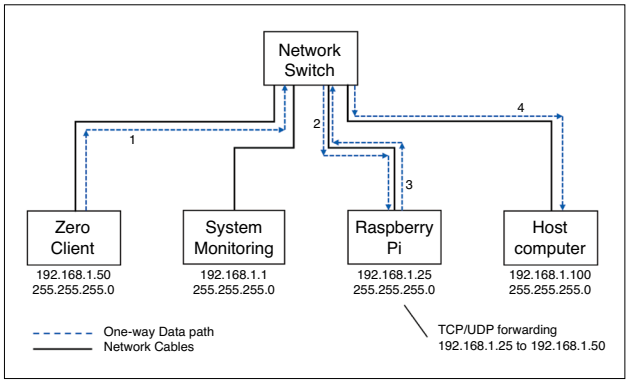

FIGURE 9. Master schematic of the testbench. Black lines represent category-5/6 Ethernet cables. Dashed lines represent a one-way data path; the resulting return packets would traverse the schematic backwards.

outfitted with a Teradici host card. In this system, the card used was the Teradici PCoIP Dual Display Remote Workstation Card (Model: TERA2220 Dual Display). A DisplayPort cable was then connected between the host computer's graphics card and the remote workstation card, as discussed and diagrammed in the "Background" section. This computer acted as the host machine or "cloud" in the testbed. The PCoIP card was then directly connected to a 1G network switch using the built-in network interface (Ethernet terminal), to connect with the testbed's other elements. Note that the network interface on the host card is separate from the computer's network interface. The host computer does not see the host card as a network interface!

A Raspberry Pi 2, Model B+, running Ubuntu Server 20.04 LTS, was also connected to the network switch. This device acted as a proxy and was used to inject latency into the network, as well as forward the PCoIP packets to the zero client through IP routing and forwarding. More information on the software configuration can be found in the section titled "Testbed Software."

The PCoIP zero client (the client receiver) was then connected to the network switch. A monitor, mouse, and keyboard were connected to the zero client, as user IO devices. As discussed in the "Background" section, additional peripherals could be connected through the zero client's built-in ports. Such peripherals may include a Cintiq or tablet. Finally, an additional and optional computer was connected to the switch. This computer was used to monitor the various devices on the network through their built-in hypertext transfer protocol (HTTP) and secure shell protocol (SSH) endpoints.

Testbed Software

Each device on the network was assigned a unique IPv4 address such that:

- PCoIP host card: 192.168.1.100/24
- PCoIP zero client: 192.168.1.50/24
- Raspberry Pi 2 B+: 192.168.1.25/24
- Monitoring computer: 192.168.1.1/24.

These could be configured differently if needed.

To set up the Raspberry Pi to forward the PCoIP packets, a Linux command line program was used called *socat*. Using this program, all TCP and UDP traffic on port 4172 (the port PCoIP uses) was sent to the host machine. The command line format, with arguments, is as follows:

```
sudo socat TCP-LISTEN:4172,fork
TCP:192.168.1.100:4172
```

```
sudo socat UDP-LISTEN:4172,fork
UDP:192.168.1.100:4172
```

Next, the Raspberry Pi was configured so that packets would be delayed, adding latency. This is accomplished through another Linux command, tc, to add a latency rule:

`sudo tc qdisc add dev eth0 root netem delay <latency>`

example:

`sudo tc qdisc add dev eth0 root netem delay 10ms`

To remove the latency rule:

`sudo tc qdisc del dev eth0 root netem`

When adding latency, the number passed through the command line should be half of what the target RTL should be. In other words, if 20 ms is desired, then "10 ms" should be passed into the command line, since packets will pass through the Raspberry Pi twice (on the way to the host machine and back to the client).

Wrapper scripts for these commands can be found on GitHub: https://github.com/burkeac/Capstone

Using the PCoIP zero client's configuration portal, the session was configured to connect using the "direct to host" option. The IP address of the Raspberry Pi was then entered, so that the packets would first be directed to the Raspberry Pi, which would then forward the packets to the proper IP address of the host machine and optionally delayed. **Figure 10** shows the session configuration page. **Figure 9** also shows a one-way data path from the client to the host machine.

Once connected, the latency can be verified using the zero client's web portal by typing the IPv4 address into a web browser on the monitoring computer. The Raspberry Pi's latency rules can then be updated to adjust the latency accordingly. It should be noted that

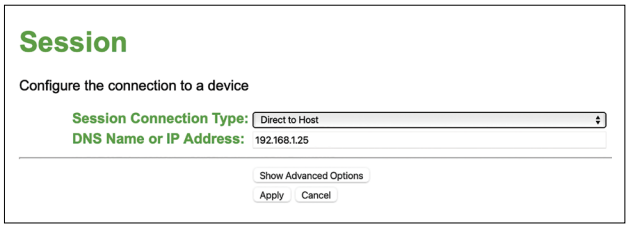

FIGURE 10. PCoIP zero client configuration page, with IP address of Raspberry Pi configured.

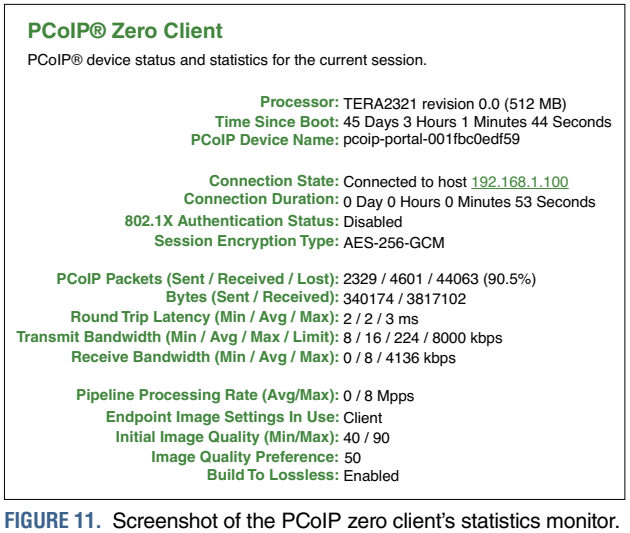

FIGURE 11. Screenshot of the PCoIP zero client's statistics monitor.

although the Raspberry Pi's latency rules take effect immediately, it does take a few minutes to get an accurate reading from the web portal's stats. An example stats page is shown in **Fig. 11**.

For the experiments, a Python script was used to randomize the order in which each latency condition was presented. In addition, the script set the latencies and recorded the response of the participant. This script can also be found on the GitHub repository.[7]

Methods—The Backup

Due to the COVID-19 pandemic, it was not safe to conduct large-scale face-to-face research as planned. As a result, some modifications had to be made to both the objective and subjective experiments.

Objective

Instead of running a full-scale JND experiment, a "pilot study" was conducted with only a handful of participants. Due to the temporal instability of latency within wide area networks, it would not be possible to run an objective JND experiment from afar, that is, across the internet. The participants used in this pilot study were not specifically trained in any tasks related to film and animation. As a result, participants were trained to perform a simple task in Adobe Photoshop: "cutting a person out of an image" using the eraser tool. Based on some pretesting, the latency deltas used in the experiment were

$$\Delta_{\mathrm{RTL}} = \{3,7,9,11,15,20\}\ \mathrm{ms}$$

where each value was presented twice and in random order.

Subjective

As a result of students being asked to go home during the pandemic, the RIT College of Art and Design responded by implementing virtual desktop environments that students could access from home to complete their assignments. This was an opportunity to collect subjective data in a "real-world" implementation. This virtual desktop solution leveraged the Microsoft and Apple remote desktop protocols (RDP) in addition to Apache Guacamole for browser-based access.

The goal during this window of opportunity was to gather subjective Likert scale data alongside objective RTL values between the client and the host. To collect this data, an online form-based solution was used. Although a number of off-the-shelf solutions do exist, such as Google Forms, they do not have a means to directly collect the objective RTL data. Likewise, the standard ping application is a command line program and not intuitive to the average film and animation student. Additionally, ICMP Echo requests are sometimes blocked by firewalls to prevent denial-of-service attacks. As a result of these issues, a custom solution was created.

This web-app could easily be accessed by students on their local machine, after using the virtual desktop environments for their assignments. Upon visiting the website, the participant would be served a short survey asking about their experience. The data collected and questions posed were as follows:

- Select the lab you are accessing remotely.
- In comparison to working at a physical lab workstation (or your personal machine), how "responsive" is using the remote system? Please consider this in terms of IO (mouse/keyboard/tablet/display, etc.).
- If given the option (outside of current circumstances), would you use a remote desktop environment in this state? (for school or work).
- What type of work are you using the remote desktop environments for?
- Please specify any software you were using in addition to the type of task.
- If you have any additional comments, please list them.

In this variation, the Likert scale goes from 1 to 4 compared to the original plan that utilized −2 to 2, as this range is more logical to the adapted questions' phrasing, particularly since there is no baseline associated with the JND data.

While filling out this form, the web-app "pings" a specified server and submits the RTL data along with the user responses. In addition, the IP address of the client is submitted and stored. A screenshot of a portion of the web-app can be found in **Fig. 12**. To collect the correct RTL data, the web-app must be running in a browser on the participant's local computer. In the event that a student loads the page from an RIT lab computer using the virtual desktop connection, the application will serve them a warning that they need to open the page via their local machine, which is based on RIT's registered IP addresses.

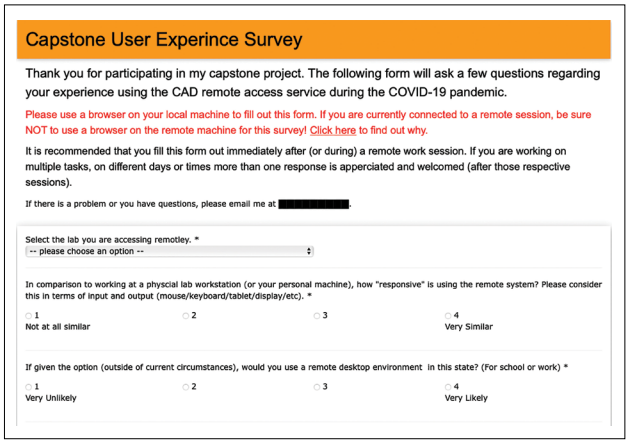

FIGURE 12. Screenshot showing a portion of the front-end user interface of the web application.

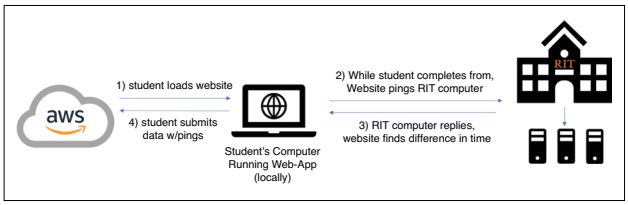

FIGURE 13. High-level diagram of the web-app for collecting subjective data.

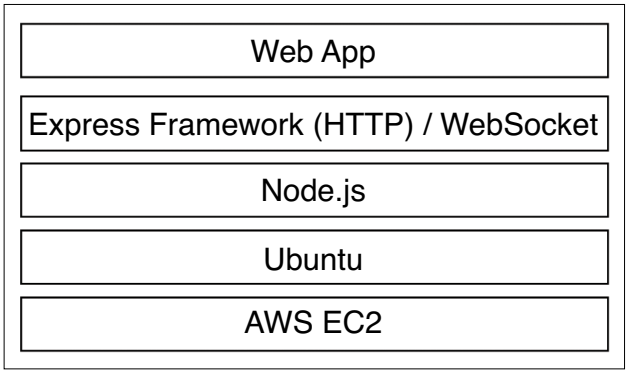

FIGURE 14. Tech-stack of the subjective web-app. Note: the ping-response server was run on an RIT server and not an AWS EC2 instance.

Web-App Backend

Figure 13 shows the overall structure of the web-app. The web-app server was hosted on an AWS EC2 instance running Ubuntu. The server ran on Node.js, the Express framework, and the WebSocket framework. The schematic of the full tech-stack is shown in **Fig. 14**.

In contrast to the main application, which served the client resources and was hosted on AWS, a separate application was run on an RIT server. This application's sole job was to respond to ping requests from the client side. It was separated from the main server for two reasons.

1. An application responsible for only returning a short reply "ping" message, through WebSockets, is computationally lighter than a server, which needs to serve client resources and accept the form input. This removes the possibility of the extra overhead adding latency.
2. This separation allows for scalability and flexibility and creates the opportunity to run the ping-response server on the virtual desktop machines themselves. This would allow the web-app to ping particular machines in varying geographic locations.

When compared to the standard command line application, this custom web solution proved to be accurate within a millisecond or two, providing an accurate representation of the RTL between the client and the remote desktop environment.

It should be noted that although the final application used WebSockets to determine the RTL, an initial implementation used Asynchronous JavaScript and XML (AJAX) requests. Although a number of forums stated that WebSockets have minimal speed advantages over AJAX, it was found that at the millisecond level, the AJAX implementation was consistently higher than the standard command line ping and less repeatable. The results of the WebSocket implementation proved to be more consistent and better aligned with the traditional command-line tool. Additionally, while implementing and testing, a tool was created that only reports the RTL data to the user. This can be found at <web-app URL>/ping. Once again, all the codes for this web-app can be found on the capstone GitHub repository.[7]

Results and Discussion

Based on the data collected in this pilot study, results are presented and discussed for both the objective and subjective studies. Due to the nature of the pilot study and the limited participant pool, results are limited.

Objective

Due to the health concerns previously discussed, only four people were able to participate in the objective study and their combined results are shown in **Fig. 15.** As discussed, participants were asked to choose which latency condition presented the worst experience. In the event they were unsure or unable to detect a difference between the two conditions, they had a 50/50 probability of selecting the correct answer. As a result, when the probability is below 0.5, it cannot be statistically determined whether the participant truly saw a change in latency or not. The probability $p = 0.75$ was selected so that from a statistics point, at least half of the participants truly saw a change and it was not a random 50/50 guess. As a result, we can determine from this set of data that the approximate JND in latency is 28 ms for the task the participants were asked to complete.

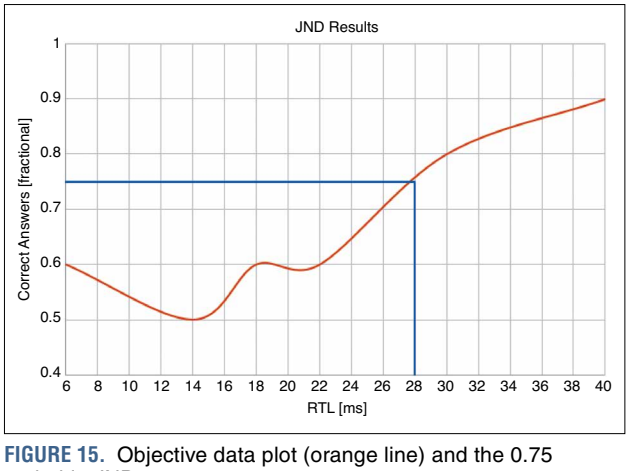

FIGURE 15. Objective data plot (orange line) and the 0.75 probable JND.

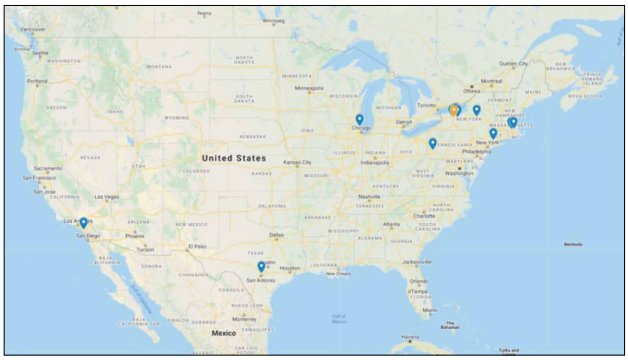

FIGURE 16. Estimated geographic locations of participants based on IP addresses.

This number does fall within the range presented by various studies as discussed in the "Background" section. However, a larger scale study should be conducted to gather more data, making it more statistically accurate and less noisy. Similarly, the Δ_{RTL} numbers may require some tweaking to obtain finer detail. Due to the restrictive subject pool, getting participants who were trained in specific film and animation skill sets was not possible. The subjects were also familiar with the research and "knew what to look for," which may have caused bias. However, I do believe that this pilot experiment begins to demonstrate the effectiveness of such a study and that with further research more defined trends will be discovered.

Subjective

As mentioned, it was not really possible to conduct the subjective study using the testbed as planned and this section will focus on the subjective web-app.

Based on the IP data collected using the web-app, IP-geolocation was used to approximate the location where clients were accessing the virtual desktop environments, as shown in **Fig. 16.** As expected, geographic regions farther from the RIT campus saw much higher RTLs.

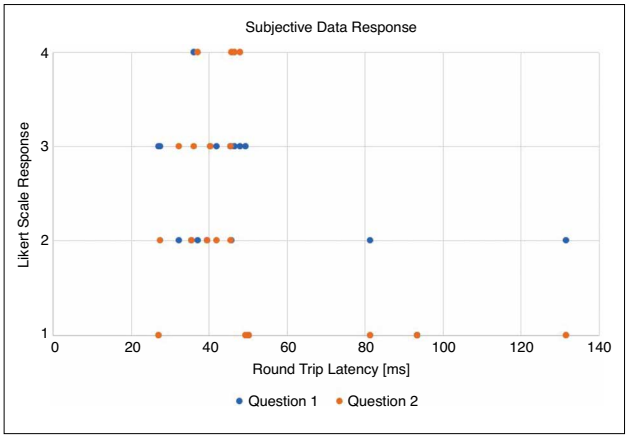

FIGURE 17. Plotted results for the subjective web-app study. Question 1: In comparison to working at a physical lab workstation (or your personal machine), how "responsive" is using the remote system? Please consider this in terms of IO (mouse/keyboard/tablet/display, etc.). Question 2: If given the option (outside of current circumstances), would you use a remote desktop environment in this state? (for school or work?)

The participant in Texas saw an RTL of approximately 80 ms, while the participant in California saw an RTL over 100 ms. Within the northeast region, particularly around RIT, latencies were around 30 ms.

In **Fig. 17**, the responses to the Likert scale have been plotted as a function of RTL. Even with the limited set of data, some trends began to emerge. Primarily, and unsurprisingly, high latencies above 60 ms have very poor user-rated performance. Latencies less than 60 ms have a large range of user preference responses. With more participation, it would be expected more significant trends would emerge.

In addition to variable latency within a specific connection, participants were also being subjected to other network variations such as bandwidth and network jitter. This makes it difficult to associate user experience with latency alone. But, it does give a nice view of a real-world implementation. The following set of numbers demonstrates the network variation of sequential RTL calculations across the internet:

[44,45,47,53,77,58,48,48,49,46,46,45,
46,45,45,37,44,46,42,46,45] ms.

The average of this dataset is: 50.1 ms, while the minimum and maximum are 37 and 77 ms, respectively.

In addition to the numerical data, written commentary from participants gave an interesting perspective. A number of users reported that working with intensive workloads, where large numbers of pixels were quickly updating, increased the latency significantly. One 3-D animator, working in Maya, called out the panning/zooming in his scene, which became "very laggy," while simple UI interactions, such as clicking buttons, worked fine. This should be looked into more to see why there is an increase in latency. Some possibilities may include the remote desktop protocol reducing the frame rate during this time or increasing the latency to receive more data before drawing the screen for the client. Unfortunately, there are no statistics from the remote desktop. However, this could give significant insights into future studies.

Additional Discussion and Moving Forward

Although minimal data was collected, this research can act as a pilot study, moving forward. It also begins to show the effectiveness of such a study and highlights potential areas where greater research could be conducted.

On the objective side, conducting a larger study with more trained participants and tasks targeted to focus on a specific craft would give significant insights into the ability for creatives to work under various latency conditions. Results from this study could then be used to create a quantitative metric in judging network performance. From here, a subjective study based on the results of the objective JND experiment can correlate the acceptability of such latency.

In addition, performing a large-scale, real-world implementation study, using similar methods to the web-app presented in this paper, could yield interesting results, particularly if social data collection was done through Reddit and other outlets. This could give additional insights into not only the preferences of users, but also last-mile delivery.

One of the major insights gained while working on this project is that last-mile delivery is very challenging, particularly in residential settings. For example, pinging the AWS East 1 region from an RIT campus resulted in an RTL of around 12 ms. However, pinging the same region from my residential network a mile down the street resulted in an RTL of 35 ms. Similar trends can be seen throughout the subjective study, where residential RTLs rarely go below 35 ms, even though some are located close to RIT. This is primarily a result of enterprise networks having more direct connections. As another example, Rockefeller Center has a direct connection to AWS East 1 and sees an RTL below 10 ms. This is based on a presentation by NBC Universal at the 2019 SMPTE Technical Conference.

Conclusion

Cloud computing and remote desktop environments can be very powerful tools for the media industry. However, since the concept remains young, there is certainly more progress to be made, particularly in terms of desktop user interaction. Yet, the industry should not be afraid to continue testing and developing such methods, particularly since a number of emerging technologies, such as 5G, promise to significantly reduce the latency between client and host.

The conclusions drawn from this study and its respective pilot study identify a number of areas where future research should be conducted. With this further research, the industry should begin to feel more confident in strategically implementing remote desktop workflows, allowing for greater flexibility and scalability in production pipelines. Methods presented and prototyped throughout this pilot study should be able to provide a starting point for future research.

Acknowledgments

I would like to thank my capstone advisor Dr. Ricardo Figueroa and Dr. David Long for their guidance throughout my capstone project, particularly during the questionable time at the beginning of the COVID-19 pandemic, at which point my project had to pivot. I would also like to thank Kylee Pena, Michael Oliver, Johnny Uribe, Blake Penido, Jennifer Zeidan, and Chris Clark of Netflix for their support in providing equipment and mentorship. Lastly, I would like to thank my friends and family for all of their support throughout my entire time at RIT.

References

1. Amazon Web Services, "Global Network," Amazon AWS. Accessed: Nov. 10, 2019. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/global_network/
2. G. Suart, "PCoIP: What Is PC-Over-IP and How Does It Work?," Petri, 27 Feb. 2012. [Online]. Available: https://www.petri.com/what-is-pc-over-ip
3. S. Doyle, "TCP Vs. UDP: Understanding the Difference," Private Internet Access, 17 Dec. 2018. [Online]. Available: https://www.privateinternetaccess.com/blog/tcp-vs-udp-understanding-the-difference/
4. J. Deber et al., "How Much Faster Is Fast Enough? User Perception of Latency & Latency Improvements in Direct and Indirect Touch," *CHI*, Seoul, South Korea, Apr. 2015.
5. J. Spjut et al., "Latency of 30 ms Benefits First Person Targeting Tasks More Than Refresh Rate Above 60 Hz," *Technical Briefs (SA)*, Brisbane, QLD, Australia, Nov. 2019.
6. C. Cámara et al., "Eye Movements in Interception with Delayed Visual Feedback," Experimental Brain Research, Apr. 2018.
7. A. Burke, "Capstone Code," 2020. Accessed: May 20, 2020. [Online]. Available: https://github.com/burkeac/Capstone

About the Authors

Adam Burke is a graduate of the Motion Picture Science program at Rochester Institute of Technology (RIT), Rochester, NY. During his time at RIT, Burke interned at Warner Bros. and was an active member of the RIT SMPTE Student Chapter. In addition, he worked at RIT SportsZone, the university's sports broadcasting group and managed the technical aspects of the end-of-semester, academic screenings for the School of Film and Animation. In 2019, he received the Louis F. Wolf Jr. Memorial Scholarship, which was followed by the Student Paper Award for his capstone project and this paper. Now graduated, he works for Peraton, a government contractor, as a sensor engineer.

Ricardo Figueroa received a BSEE and MSEE degree from the University of Puerto Rico at Mayagüez, Mayagüez, Puerto Rico, and a PhD degree in computing and information sciences from Rochester Institute of Technology (RIT), Rochester, NY. He is currently an associate professor and the program director of the Motion Picture Science Program, and the interim co-director of the School of Film and Animation at RIT. He joined RIT after working at Eastman Kodak for ten years, where he held positions as the film and digital lab manager, the digital/hybrid technologies regional director, the digital imaging research engineer, and the kodak operating system manager. His research interests are in the areas of machine learning and deep learning for image processing applications.

This paper was the winner of the SMPTE 2020 Student Paper Award.