How Close is Close Enough ? Understanding the Role of Cloudlets in Supporting Display Appropriation by Mobile Users

Sarah Clinch†, Jan Harkes‡, Adrian Friday†, Nigel Davies†, Mahadev Satyanarayanan‡
†Lancaster University and ‡Carnegie Mellon University

***Abstract*—Transient use of displays by mobile users was prophesied two decades ago. Today, convergence of a range of technologies enable the realization of this vision. For researchers in this space, one key question is where to physically locate the application for which the display has been appropriated. The emergence of cloud and cloudlet computing has increased the range of possible locations. In this paper we focus on understanding the extent to which application location impacts user experience when appropriating displays. We describe a usage model in which public displays can be appropriated to support spontaneous use of interactive applications, present an example architecture based on cloudlets, and explore how application location impacts user experience.**

I. Introduction

Transient public display use was prophesied in Weiser's seminal paper on ubiquitous computing [1]. Today, a convergence of technologies enable this vision: mass-market display hardware for affordable deployment, virtual machines (VMs) for transient customisation, smartphones for actuation and access control, wireless networks for untethered communication, and cloud offloading for resource-intensive tasks. All the pieces are at hand to realise Weiser's vision.

Mobile devices face a fundamental tradeoff between size, weight and usability; the ability to briefly appropriate a display relaxes these constraints. We envision a dynamic, interaction-rich model whereby users walk up to a display and temporarily use it to augment their mobile device. User devices remain the root of identity, trust, customisation and interaction. Likewise, privacy-sensitive information only appears on the mobile device (e.g. as in [2]).

To illustrate this vision, consider the following scenario: *Dr. Jones is at a restaurant when she is contacted about a pathology slide that must be interpreted while surgery is in progress. Walking up to a large display in the lobby she views the slide at full resolution over the Internet. Using her smartphone for control, she zooms, pans and rotates the slide as if at a lab microscope. Privacy-sensitive clinical information displays on her smartphone. Dr. Jones interprets the slide, telephones the surgeon, and returns to dinner.*

One can imagine other scenarios involving transient display use, perhaps by those many miles from a home environment. Tourists might use displays to show personalised city information, whilst a visiting expert in an industrial setting might use displays to view detailed engineering plans.

We suggest that this vision can be realised through dynamic VM synthesis supported by cloudlets [3]. We focus on a set of critical questions: Where should an application execute for good user experience? Can it execute on a distant cloud with high network latency? Or, is it necessary to execute closer to the display and user? Can we quantify the impact of latency on user experience?

We do not attempt to provide full analysis of cloudlet use for display appropriation—issues such as appropriation overheads, start-up costs, trust and privacy and sharing remain items for future work. Instead, we focus on offering insight into the impact of application location on user experience through detailed measurements and a user study.

II. Cloudlets and Display Appropriation

Our envisaged usage model has significant architectural consequences. End-to-end latency between user action and the appearance of causally-related frames on the display must be tightly bounded – particularly critical for immersive applications which may depend on demanding interaction methods (e.g. gesture recognition).

The need for substantial computation within tight latency bounds poses a unique challenge in the context of mobile computing. Although a user's mobile device is just one wireless hop from the display, such devices are always resource-constrained. The alternative, leveraging cloud computing, is constrained by the need for tightly-bound latency; user experience may be negatively impacted by latency, jitter, congestion and the failures that can occur with WAN access.

An alternative option uses *cloudlets* to reconcile these contradictory demands. A cloudlet is a trusted, resource-rich computer or cluster, with good connectivity, available for use by nearby mobile devices (cf. WiFi access points today).

For display appropriation, cloudlets may be co-located with displays; the mobile device acting as a thin client to applications executing in the cloudlet driving the display. Close proximity of the cloudlet helps address latency requirements.

A system for display appropriation can be built using virtual machines, c.f. *dynamic VM synthesis* [4], in which a mobile user rapidly instantiates a customised server on a nearby cloudlet. A small *VM overlay* is delivered from mobile to cloudlet (the cloudlet already possesses the base VM). The cloudlet applies the overlay, deriving a *launch VM*, and starts execution in the precise state at which the overlay

was created. Synthesis does not mandate Internet access and can work wherever mobiles and cloudlets are proximate. Synthesis times of 1–2 minutes have been reported [4]; our experiments have demonstrated times between 20–30 seconds, making the approach viable for display appropriation.

This paper explores the extent to which cloudlets are necessary to support display appropriation for interactive applications, focussing on how execution location impacts user experience. This is an important area to address to inform the debate as to the extent to which cloudlets will become a key part of future pervasive computing infrastructures.

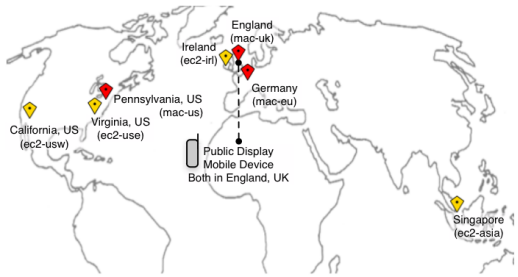


Figure 1. Geographic Distribution of Mac Cloudlets and EC2 Nodes

III. The Impact of Application Location

Based on the scenario in Section I, we approximate Dr. Jones' interactions with the display. We measure VNC interaction from a mobile device via a cloudlet (VM guest) hosted in both commercial cloud and dedicated cloudlet hardware at varying distances from the study location (Figure 1).

A. Mobile Device & Display

To simplify automation and packet tracing we emulate the mobile device with a Mac Mini (2 GHz Intel Core Duo, 1GB RAM) running Ubuntu 9.10 over wireless (802.11) networking. We measure task performance with an actual mobile device in our user study (Section V).

The display machine was a Mac Mini (2.26 GHz Intel Core 2 Duo, 4GB RAM) running Mac OS X 10.5 and a wired LAN. Chicken of the VNC, an open-source VNC client, was used to connect to the cloud/let VM.

B. Application Host Environments

We deployed 7 servers: 3 physical cloudlet machines and 4 cloud VM instances. The cloudlets were placed within i) the same subnet, ii) central Europe and iii) Eastern USA (see Figure 1). Each cloudlet (a Mac Mini: 1.83GHz Intel Core Duo, 2GB RAM) ran Mac OS X 10.4 and VirtualBox 3.0.12 with a Ubuntu 9.10 guest VM (allocated 1 CPU, 395MB RAM and 8GB disk). The VM ran the Vino VNC server and Python 2.6 ; network was bridged to the host's Ethernet.

The cloud instances ran on the Amazon Elastic Compute Cloud (EC2[1]) with 1 Compute Unit (32-bit Ubuntu 9.10 Server, 1.7GB RAM, 160GB storage). Instances ran vnc4server and Python 2.6—we observed no significant performance difference between vnc4server and Vino.

[1]http://aws.amazon.com/ec2/, accessed 25^{th} January 2012.

C. Measurement Methodology

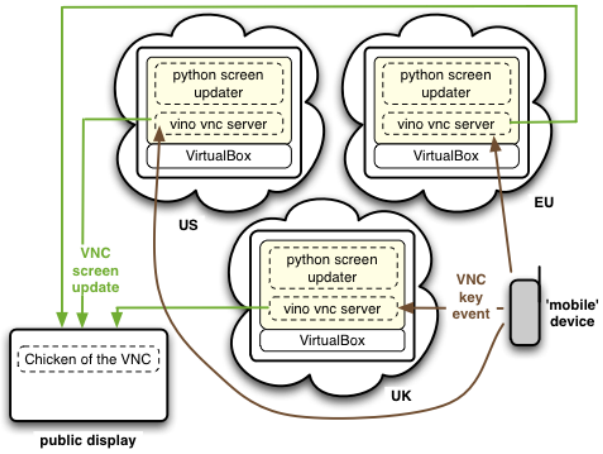


Figure 2. Benchmarking Architecture

Figure 2 depicts our experiment. The 'mobile device' sends pairs of VNC key events to the cloud/let VM; this triggers a screen update which is then received by the display machine over VNC. We measure from the first VNC key down event to reception of the screen update. We assume a delay between the screen update arriving and screen refresh, but this will be uniform across experimental conditions (the same display, OS and VNC client are used throughout).

We use packet traces (tcpdump) to log protocol interactions. The display machine's Ethernet is mirrored to allow capture with a common clock. We also packet capture on all the nodes to measure application processing latency. System clocks are synchronised using NTP. The resulting measures give us a simple aggregate for typical application performance including VM, OS, and IP stack latency.

IV. Results for Mac Cloudlets

We took measurements from 18^{th} Nov. to 6^{th} Dec. 2010, 1am–11pm in batches of 15, ~1/min, in a UK, EU, USA rotation. Our dataset contains 6,011 measurements (~2,000 per location) [Table 1/Figure 3].

The median update time (reflecting typical delays that could be expected by users) are 60ms, 92ms and 171ms (UK, EU, US respectively) [Table 1]. Distributions are tight to the median (IQR 31ms), but significant outliers due to TCP retransmissions skew the mean. Removing measurements impacted by retransmissions drops the maximum update time by 15–24 seconds and reduces the SD to $\frac{1}{10}^{th}$ of their previous values. Neither day nor time is significant.

Using linear regression (retransmissions excluded) we reject the null hypothesis (H_0: no effect of location on screen

Site	Min	1Q	Median	Mean	3Q	Max	SD	SE	n
Including TCP retries (latency with retries excluded shown in parentheses)									
us	122 *(122)*	157 *(155)*	171 *(168)*	345 *(186)*	188 *(183)*	23680 *(1482)*	897.65 *(99.03)*	19.92 *(2.35)*	2030 *(1770)*
eu	40 *(40)*	77 *(74)*	92 *(88)*	257 *(93)*	108 *(102)*	15440 *(1702)*	1046.41 *(54.36)*	22.79 *(1.27)*	2108 *(1828)*
uk	10 *(10)*	45 *(43)*	60 *(57)*	192 *(59)*	75 *(71)*	25940 *(527)*	1123.48 *(27.99)*	25.96 *(0.69)*	1873 *(1651)*
Cloud processing time vs. network latency excluding retries [network latency shown in square brackets]									
us	6 [114]	38 [115]	52 [115]	69 [117]	66 [116]	1367 [342]	98.87 [7.47]	2.35 [0.18]	1770 [1770]
eu	10 [10]	42 [31]	56 [31]	60 [34]	69 [32]	706 [1640]	38.88 [38.17]	0.91 [0.89]	1828 [1828]
uk	8 [1]	39 [2]	54 [2]	55 [4]	68 [3]	525 [120]	27.45 [6.26]	0.68 [0.15]	1651 [1651]

Table 1
SUMMARY OF MAC CLOUDLET UPDATE TIMES IN MS (MIN, MAX, MEAN, QUARTILES, STANDARD DEVIATION AND STANDARD ERROR)

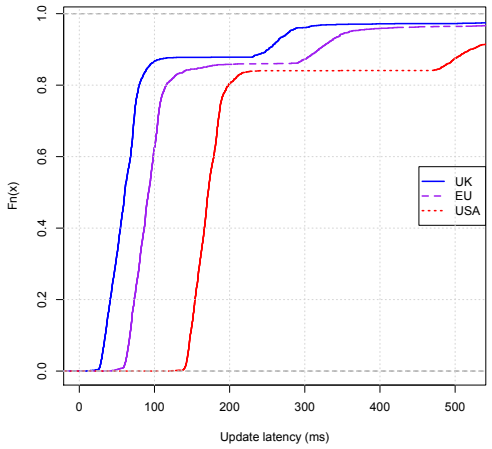


Figure 3. CDF of update response time (ms) by Mac cloudlet location. The bottom 95% (<533ms) is shown for clarity; the long tail (due to TCP retries) affects 13% of requests. The bottom 80% of the data is normally distributed (Shapiro-Wilk W=0.96, 0.98 and 0.98 respectively, p <0.001).

update time): EU 34.7ms > UK ($t = 15.11, p < 0.001$) and US 127.6ms longer ($t = 55.08, p < 0.001$). Including retransmissions, we still reject H_0 (strongly in the US case) but the model is a poorer explanation of the data (Fisher's F-statistic $F_{2,6008} = 11.02$ vs. $F_{2,5246} = 1644$, $p < 0.001$).

Homogenous hardware and software gives comparable processing time across sites (mean ~50ms). Small O(4–14ms) but statistically significant differences correlate with network distance (EU 4ms > UK, $t = 2.2, p < 0.03$; US 14ms > UK, $t = 6.6, p < 0.001$). Unexplained outliers (up to 1.4 seconds!) suggest missing factors—however, 98% of updates fall below $130ms$ suggesting these are insignificant.

The number of retransmissions does vary by site: single retransmissions account for over 85%: presumably due to the high error rate of the 1^{st} (wireless) hop. We attribute a larger number of 2^{nd} and 4^{th} attempts in the US case to a longer path (20 hops with a transatlantic link).

V. USER EXPERIENCE

To explore the impact of application location on user interaction, and emulate the pace of interactions observed in experts comparing medical images, we developed a simple interactive game ('whack-a-mole', Figure 4). This latency-sensitive game has a deliberately shallow learning curve.

A. Methodology

The game ran locally on an Apache server instantiated on each cloudlet and displayed in a fullscreen web browser, mirrored on the public display using VNC. Participants use an iPhone as a trackpad via the 'RoboHippo' VNC client.



Figure 4. The whack-a-mole game

Participants were opportunity sampled and randomly assigned to one of two conditions: games played in increasing or decreasing order of network distance. We chose these progression/regression groups to balance order effects, and avoid a factorial design of 6 groups (potentially with few participants in each condition). Users were given a practice round (in a randomly selected location) to reduce the impact of learning effects (a trial without this round suggested users hit considerably fewer moles on their first attempt).

B. Results

Results were collected from 29 participants: staff (10), students (15) and visitors (3). Fourteen aged 18–24 years, thirteen aged 25–34 and two aged 45–54. 66% were male.

Game performance (moles hit, mean hit time and cursor movement) was collected during play and relayed to a central server at the end of each round. User experience was surveyed using a questionnaire.

We found VNC to be highly robust: approximately 40 connections were required to each cloudlet. Of these, only 4 (all between the phone and the US cloud) were dropped.

1) User Performance: Mole hit-rate differs across the conditions: 87%, 85% and 71% in the UK, EU and US respectively. There is a significant difference between UK and US ($t = 6.29, p < 0.001$) suggesting that the longer latency (+79/+111ms from EU/UK) did impact gameplay.

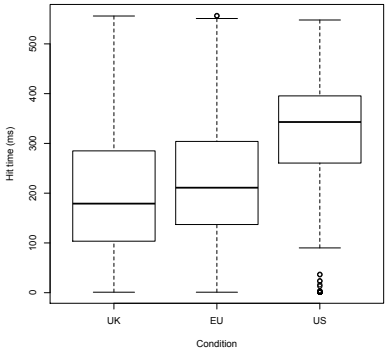


Figure 5. Mole hit time by site. The US condition is visually distinct.

Table 2 summarises the hit times in each condition. Whilst the interquartile range and standard deviation remain fairly consistent ($IQR = 132$–$179ms$, $SD = 126 - 142ms$), the median time increases with underlying latency (UK $179ms$, EU $211ms$ and US $343ms$) [Figure 5].

Gameplay is impacted by cloudlet location. Using linear regression ($F_{2,1056} = 57.03$) we find that the EU takes 25.4ms > UK with low statistical significance ($t = 2.55, p <= 0.01$) and US 108.5ms longer ($t = 10.35$, $p < 0.001$). We suspect that the lowest hit times (e.g. $1ms$) are due to the player fortuitously having the cursor positioned where the mole appears. The upper bound on hit time (~$550ms$) is consistent across conditions. The effect of ordering was not statistically significant $t = 1.24$, $p = 0.22$.

2) User Experience: Users marked points on a printed scale showing the extent to which they felt the game was responsive, usable, frustrating, and under their control. They also marked the point at which they would choose not to play the game. This method allowed subjective comparisons rather than absolute ratings on a traditional Likert scale (Figure 6). In each case, UK and EU games scored positively (median responsiveness: UK 45, EU 23.5; usability: UK 50, EU 33; frustration (scale inverted): UK 42, EU 26; sense of control: UK 52, EU 32) while US games were borderline negative (median responsiveness: -1.5, usability: 8, frustration (scale inverted): -3, sense of control: 0).

16 participants provided additional comments: 10 made specific comparisons about the control or speed of the games, whilst a further 4 commented more generally that speed or control differed. Of the comparative comments, 3 participants suggested that the EU and US conditions were easier to control and/or faster than the UK; 6 that the EU and UK conditions were both easier to control/faster than US; and 1 that the UK was faster/easier to control than the EU which was faster/easier than the US.

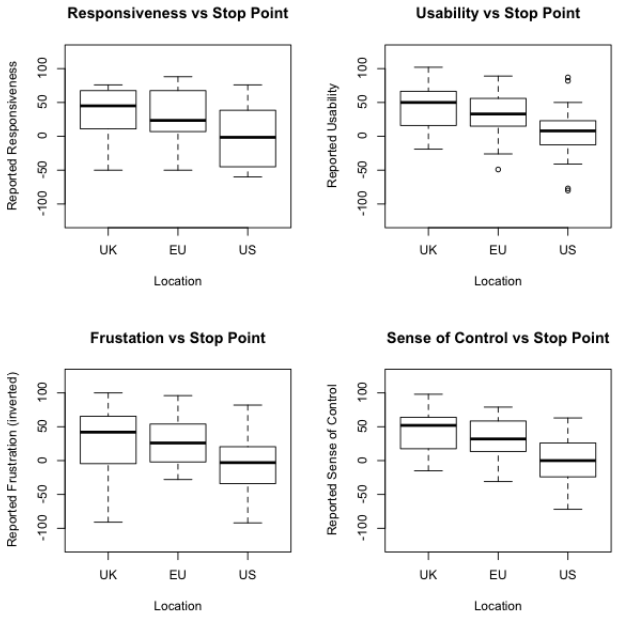


Figure 6. User perception: positive values indicate users would play the game (reported value greater than stop point); negative values the converse.

VI. Results for EC2 Cloud VMs

To understand how commercial cloud hosting could provide an infrastructure for cloudlets we repeated our methodology [Section IV] gathering 5,266 measurements across the four EC2 locations (~1,300 per site) [Table 3/Figure 7]. We found median update times of 90ms (Ireland), 161ms (East Coast), 227ms (West Coast) and 319ms (Asia).

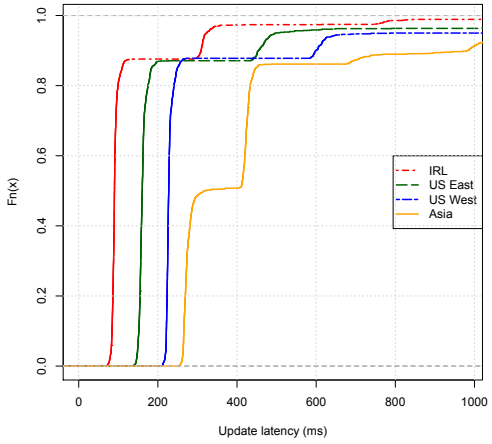


Figure 7. CDF of update response time (ms) by EC2 site. The top 5% are excluded to allow us to focus on the shape of the main distribution. The long tail is again TCP retries (12.8% of requests include retransmissions). An interesting feature is the step at 50% in the 'Asia' case.

The EC2 instances offer more consistent processing delay than the Mac cloudlets, $IQR =$ ~$12ms$ and $SD = 6ms$. They are homogeneous and broadly normally distributed

Site	Min	1Q	Median	Mean	3Q	IQR	Max	SD	SE	n	Misses	Misses (%)
mac-usa	1	263	343	316	395	132	548	126.35	6.06	435	126	29.0%
mac-eu	1	137	211	233	304	167	557	138.81	6.66	435	64	14.7%
mac-uk	1	104	179	207	284	180	556	142.48	6.83	435	55	12.6%

Table 2
MOLE HIT TIME AGAINST LOCATION (MIN, MAX, MEAN, QUARTILES, INTER-QUARTILE RANGE, STANDARD DEVIATION AND STANDARD ERROR)

Site	Min	1Q	Median	Mean	3Q	Max	SD	SE	n
Including TCP retries (latency with retries excluded shown in parentheses)									
asia	254 *(254)*	271 *(270)*	319 *(284)*	692 *(337)*	429 *(422)*	18160 *(488)*	1642.92 *(75.54)*	44.71 *(2.22)*	1350 *(1162)*
usw	212 *(212)*	224 *(224)*	227 *(226)*	394 *(228)*	235 *(230)*	12750 *(272)*	917.49 *(8.20)*	25.26 *(0.24)*	1319 *(1158)*
use	135 *(135)*	156 *(155)*	161 *(159)*	295 *(161)*	169 *(164)*	9956 *(227)*	738.24 *(9.30)*	20.21 *(0.27)*	1334 *(1162)*
cirl	72 *(72)*	87 *(87)*	90 *(90)*	187 *(91)*	96 *(93)*	14710 *(137)*	754.88 *(7.05)*	21.23 *(0.21)*	1264 *(1107)*
Cloud processing time vs. network latency excluding retries [network latency shown in square brackets]									
asia	55 [191]	68 [198]	71 [212]	72 [265]	74 [354]	98 [402]	6.35 [75.37]	0.19 [2.21]	1162 [1162]
usw	57 [154]	68 [155]	71 [155]	71 [157]	73 [156]	107 [201]	5.79 [5.67]	0.17 [0.17]	1158 [1158]
use	51 [82]	67 [88]	70 [89]	70 [90]	73 [91]	105 [152]	5.72 [7.43]	0.17 [0.22]	1162 [1162]
irl	51 [19]	66 [20]	69 [20]	68 [22]	71 [22]	84 [71]	4.39 [5.67]	0.13 [0.17]	1107 [1107]

Table 3
UPDATE TIMES FOR EC2 NODES IN MS (MIN, MAX, MEAN, QUARTILES, STANDARD DEVIATION AND STANDARD ERROR)

(Shapiro-Wilk $W = 0.84 - 0.94$, $p < 0.001$). Our cloudlets were capable of lower processing latency (1^{st} quartile ~$38ms$ vs. ~$67ms$); however, the 3^{rd} quartile is almost identical (~$68ms$ vs. ~$73ms$). Again, there are small (order of $2-3ms$) but statistically significant differences between the sites. We do not see the same kinds of outliers observed in our cloudlet nodes: all the data falls under $107ms$. This consistency is a nice emergent property of the EC2 cloud.

Asia's update time behaves differently: SD is ~2× the other EC2 nodes, 10× ignoring TCP retransmissions. Figure 7 clearly indicates $> 50\%$ of the updates take less time ($median = 272ms$) than the rest ($median = 425ms$). There is a significant reduction ($157ms$) in network latency during December, highlighting the lack of control developers have over end-to-end performance to cloud services.

Applying linear regression (retransmissions excluded), we reject H_0 (no effect of location on screen update time): East Coast takes 70ms longer than Ireland $t = 43.16$, West Coast 138ms longer $t = 84.79$ and Asia 246ms longer $t = 151.65$, $p < 0.001$ in all conditions. Once again, adding in retransmissions and all the Asia data, we still reject H_0, although the model is a less good fit: F-statistic: $F_{3,5263} = 52.94$ as opposed to $F_{3,4585} = 8385$.

As before, the number of retransmissions varies with location. Single retransmissions account for 54–80%. Longer paths (West Coast and Asia) exhibit more 2^{nd}, 3^{rd} and 4^{th} level retransmissions. The number of retransmissions varies from 157–188, but is not correlated with network distance.

Comparing our cloudlet testbed with EC2 sites at similar network distances: we find Germany and Ireland directly comparable. Mean and median update times are within $2ms$ and quartiles within $10ms$. The median processing time is within $10ms$ and the 3^{rd} quartile within $3ms$. The cloudlets are more variable, in the best case, up to 55% faster ($30ms$).

VII. ANALYSIS

A key argument for co-location of cloudlets and users is that user experience suffers if execution of latency-sensitive applications is too far away. So, "How close is close enough?" Answering this question is not easy. The maximum tolerable distance between application and user depends on: the interaction-intensity of the application; end-to-end network latency (whilst loosely correlated with physical distance, measurements presented here show that the nature of the correlation is complex); the host's hardware and software; and the user—some are more tolerant of delays than others. Even the same user may respond differently over time (e.g. becoming less tolerant when in a hurry).

Overall, we find EC2 nodes offer dependable interaction performance with good statistical properties (i.e. a tight performance envelope: mean $SD = 5.56ms$ and lower bound $51ms$). Screen update time is correlated with network location and the lower bound performance in our study was between $72ms$–$254ms$. It is worth noting that network paths vary and are outside of developer control (we found a $157ms$ latency reduction overnight). Longer paths exhibit a higher proportion of compound successive failures, leading to long interaction delays, and suggesting that local cloudlet use *is* more dependable. Interestingly, the maximum delay was for our UK cloudlet (1 hop)—implicating the wireless hop.

User performance was poorer in the US condition with participants missing 2× as many moles. Mean hit time is 1.5–3× the underlying latency, increasing with network delay (median latencies in parentheses): UK $207ms$ ($60ms$), EU $233ms$ ($92ms$) and US $316ms$ ($171ms$). There is no statistically significant relationship between number of hits and hit time or cursor movement. Human perception theories

suggest perceptual processing takes 50–$200ms$ depending on task and conditions [5]. We hypothesise that interactions are perceived as instantaneous in the UK and EU conditions, but that the feedback loop breaks down in the US condition, slowing reactions and contributing to misses. We found users surprisingly adaptable, suggesting a range of possible application locations (beyond the immediate network edge).

Whilst our results suggest a preference for the game to be as local as possible, the distinction between conditions is less clear. Many users struggled to distinguish between conditions: 8 users scored them equally on at least one attribute. Informal comments indicate that participants attributed poor US performance to personal factors ("*I was much slower to react*"), scoring the games equally on the questionnaire. This misattribution may not carry over to other applications (e.g. if less playful or more latency-sensitive).

VIII. Background and Related Work

Falling hardware costs have seen the emergence of commercial networks of digital displays, whilst research projects have explored user-customization of display content. An early work, FLUMP [6], showed personalised content to passers-by identified using Active Badges. GroupCast and UniCast [7], also used badges to identify users at a display, showing information that reflected their interests.

Recent projects have explored the use of mobile phones to support user appropriation of public displays. MobiLenin [8] used a mobile phone application to allow users to vote for music in a restaurant; a projection displayed the voting outcome and resulting music video. Other works (e.g. [9]) used Bluetooth-enabled mobile devices to allow users to appropriate displays; Bluetooth device names encoded commands which determined display content.

Each of these works allowed display appropriation within specific constraints (e.g. a set of predefined applications). In contrast, this paper explores VM use to widen the potential for display appropriation. As with commercial cloud computing today, we expect that this approach will support the widest range of applications and operating systems.

The combination of VMs and mobile devices has been explored for application partitioning [10] – in which processing is automatically ofloaded from a mobile to the fixed network. In this paper we offer insight into where such processing should be placed to support crisp user interaction.

IX. Conclusion

Transient user appropriation of public displays offers possibilities for new applications and usage models. In designing systems to support appropriation there are numerous possible architectures, and applications may be executed in a range of locations. Our results help inform architectural and placement choices. Through measurement and user study we have shown that application placement can significantly impact performance and user experience.

We confirm the intuition that moving applications closer improves user experience. At the same time, results suggests that provision of displays for appropriation need not wait for the deployment of localised cloudlets. *For minimally immersive/interactive applications, display appropriation can leverage commercial cloud provision.*

If localised cloudlet infrastructure emerges, it may broaden the applications and usage scenarios benefiting from display appropriation. We do not yet know whether work-related scenarios (e.g. Dr. Jones') require cloudlets. Future work will expand our study to include: the extent to which highly interactive/immersive applications demand different placement strategies; the potential for a hierarchical approach to application placement; and dynamic VM migration based on runtime discovery of interaction characteristics.

Acknowledgements

This work was supported by European Union Seventh Framework Programme (FP7/2007-2013), grant #244011 and National Science Foundation, grant CNS-0833882.

References

[1] M. Weiser, "The Computer for the 21st Century," *Scientific American*, 1991.

[2] S. Berger, R. Kjelsen, and C. Narayanaswami, "Using Symbiotic Displays to View Sensitive Information in Public," in *PERCOM '05*, 2005.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Internet Computing*, vol. 8, no. 4, 2009.

[4] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient Customization of Mobile Computing Infrastructure," in *Proc. of the MobiVirt 2008 Workshop on Virtualization in Mobile Computing*, 2008.

[5] S. Card, T. Moran, and A. Newell, *The model human processor: An engineering model of human performance.* John Wiley & Sons, 1986.

[6] J. Finney, S. Wade, N. Davies, and A. Friday, "Flump: The flexible ubiquitous monitor project," in *Cabernet Radicals Workshop*, 1996.

[7] J. McCarthy, "Using public displays to create conversation opportunities," in *Workshop at CSCW 2002*.

[8] J. Scheible and T. Ojala, "Mobilenin combining a multi-track music video, personal mobile phones and a public display into multi-user interactive entertainment," in *Multimedia 2005*.

[9] Rui José and Nuno Otero and Shahram Izadi and Richard Harper, "Instant Places: Using Bluetooth for situated Interaction in Public displays," *IEEE Pervasive Computing*, pp. 52–57, 2008.

[10] M.-R. Ra, A. Sheth, L. B. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices." in *MobiSys 2011*.