The Haunted House: Networking Smart Homes to Enable Casual Long-Distance Social Interactions

Meghan Clark and Prabal Dutta
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109
{mclarkk, prabal}@umich.edu

ABSTRACT

Despite the dominance of social networking and communications in nearly every aspect of our digital lives, little work has been done to examine the unique contributions that networked smart homes can make in the space of technologically-mediated human interaction. In this work, we introduce an application called "ghosting" that turns a smart and connected home into a socially-connected home. We show that unlike direct teleconferencing, the Internet of Things supports more subtle, ambient, and incidental exchanges that can make an environment feel co-inhabited by a person who may be many miles away.

Ghosting synchronizes audio and lighting between two homes on a room-by-room basis. Microphones in each room transmit audio to the corresponding room in the other home, unifying the ambient sound domains of the two homes. For example, a user cooking in their kitchen transmits sounds out of speakers in the other user's own kitchen. The lighting context in corresponding rooms is also synchronized. A light toggled in one house toggles the lights in the other house in real time. We claim that this system allows for casual interactions that feel natural and intimate because they share context and require less social effort than a teleconference or phone call. We describe the design points of the system and explore the successes and limitations of the ghosting user experience by implementing and deploying a ghosting application in two different settings.

Categories and Subject Descriptors

C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time and embedded systems; J.7 [**COMPUTER APPLICATIONS**]: Computers in Other Systems—*Consumer Products*

General Terms

Design, Experimentation, Human Factors, Performance

Keywords

Smart home, Application, Light control, Audio communications, Stream processing, Ubiquitous computing, Intimate computing

IoT-App'15, November 1, 2015, Seoul, South Korea.
ACM 978-1-4503-3838-7/15/11.
DOI: http://dx.doi.org/10.1145/2820975.2820976.

1. INTRODUCTION

Recent reports reveal that consumers think the Internet of Things (IoT) will be as revolutionary to society as the smartphone. Despite this optimistic prediction, consumers are not sure what the killer application will be, though the overwhelming majority are confident that they will know it when they see it [4].

However, what if consumers have already recognized the killer app? Consumer comparisons between IoT platforms and smartphones suggests that we can gain insight into what consumers find compelling by examining smartphone app usage. According to the 2014 US Mobile App Report, of the top 25 most-visited smartphone mobile apps, nine applications are social communications [11].

Social networking has permeated the fabric of contemporary social interactions. 73% of US youths use social networking websites of some type [12]. However, while the number of daily virtual interactions is unprecedented, never before have we lived so independently. The increased mobility of modern life has lead to a diaspora of friends and families across continents. Citizens are living longer and frequently aging in place, but are often unable to travel easily. Long distance relationships are common. Offices are increasingly supportive, rather than skeptical of, working remotely. These gulfs in our lives, paired with the innate desire for human companionship and community, create a desire for technologically-assisted socialization.

Smart homes are uniquely poised to impact daily social interactions in a way that existing technologies cannot. Homes are inherently a social environment, strongly tied to notions of community, support, intimacy, and safety. When augmented with networks, which inherently have the ability to overcome enormous distances; sensors, which capture context; and actuators, which can replicate it, homes across vast spaces can be made to feel close again.

In this work, we introduce "Ghosting," a smart home app that creates a shared physical space and enables casual interactions in a way that cannot be captured by explicitly-initiated and context-free telecommunications media, such as Skype, phone calls, or messaging. Occupants of a house will hear the occupants of the remote home walk through the rooms of their house, rustle papers, and clear their throat. As occupants turn lights on and off in one house, corresponding lights will turn on and off in the other. These casual interactions allow people separated by distance to be present in each others' daily lives by merging two homes into a single synchronized virtual space.

We envision a future where grandchildren run through the house playing raucously while their grandmother looks up from her book to laugh and egg them on – and the grandmother is not even in the same state. With this work, we hope to make that future a reality by opening up exploration into the space of social and messaging apps for smart homes.

2. RELATED WORK

Throughout history humans have employed technology to communicate across long distances, and the recent emergence of digital technologies has proven no exception. In 2003, Bell defined "intimate computing" to mean "technologies that enhance or make possible forms of intimacy between remote people that would normally only be possible if they were proximate." [2] The HCI community has produced a substantial body of work exploring this notion from different angles. One approach has been to imbue particular artifacts with context-sharing capabilities, such as The Bed and The Lover's Cup for couples in long-distance relationships, and the Messaging Kettle for adult children and parents who live apart [5, 6, 7]. Other works have focused on sharing physical sensations, such as hugs and hand-holding [16]. The recently-released Apple Watch allows users to send each other their heartbeats via haptic feedback.

However, ethnographic studies into long-distance relationships suggests that there is also a desire to synchronize and share *space* [3, 10, 15], which has not received as much attention from intimate computing efforts as artifacts and physical sensations. Couples in long-distance relationships report running Skype in a room for long periods of time even when not actively communicating in order to, in their own words, "simulate shared living." Existing spatially-aware communication systems do not satisfy this need. For example, the Family Intercom uses information about user location and ubiquitous audio to improve the logistics of setting up an active conversation between remote occupants, rather than making the environment feel physically coinhabited even when no one is speaking [14].

Work at Xerox EuroPARC in the early 1990s on the RAVE and Portholes projects used ambient audio and video feeds to create awareness of the activities of remote colleagues [8, 9]. Recent work has explored the ability to seamlessly capture the sound and visual context of an environment and recreate it in another location [13]. Our work ties the themes of these projects together by utilizing IoT technologies to create a shared physical environment that supports passive as well as active levels of social engagement and bonding.

3. DESIGN

3.1 Use Case: An Example Call

Alice and Bob have been partners for many years, but Alice has just moved across the country. Fortunately, they both have smart homes that support *ghosting*. Bob decides to make a "house call" and uses the ghosting app on his phone to call Alice. Alice, sitting at home, receives a notification of an incoming call and chooses to accept it. Before transmission begins, she is given the option to "mute" or "unmute" transmission of her own home's audio and light context, as well as to mute or unmute Bob's.

Once Alice accepts the settings, all the color-controllable lights in the house turn red and all the power-controllable lights flash several times, alerting any other occupants that audio and house controls are about to become synchronized. After the occupants have been alerted, transmission begins. Non-intrusive yet visible red LED indicator lights in each room turn on to indicate audio transmission, reception, or both.

When the call starts, Alice is working at the desk in her office. Though she cannot see him, she hears Bob walk from the bedroom into the living room. While he surfs the Internet, Bob hears Alice shuffle papers in the other room, and the creak of a chair as she leans back from time to time. Later, Alice sees the kitchen light turn on down the hall and hears Bob begin cooking, which reminds her that it is now dinner time. She joins Bob in the kitchen and they chat about current events. Alice interrupts the conversation occasionally to read out the next recipe instruction to Bob, who has his hands full cooking. The call remains running until the evening, when Bob decides it is time for bed. He wanders into the living room where Alice is quietly reading, and wishes her good night. She replies with the same, and Bob ends the call so that Alice cannot hear him snore.

3.2 Design Points

To ensure that occupants feel comfortable sharing audio, there are several design points that an implementation should capture.

The system should provide the ability to mute and unmute both TX and RX at any time during a call. Occupants should feel as though they have fine-grained control over what is being transmitted or received at any given time. Users of online communications services often curate their digital presence to ensure that it is not unflattering. Similarly, users should not have to transmit in certain rooms or certain situations that make them uncomfortable in order to also experience the positive aspects of ghosting.

The system should provide the ability to configure TX and RX before a call begins. This is motivated by the same insight into user preferences as the previous design point, while specifically considering the sensitive transition from no call into a call. We propose that TX/RX controls be provided on all phone and web-based user interfaces, as well as hardware TX/RX controls for individual microphone and speaker arrays in each room.

The system should warn all occupants that a call is about to begin. If you are the occupant of a home, you should be warned by the system that a call is about to begin even if you do not have a phone or computer or are not currently interacting with a phone or computer. Children in particular are often members of this population, and warrant extra consideration. Since the environment itself is the sensitive context, it is the environment that should alert the occupants when a transition is about to be made between private audio to shared audio. In this work, we propose turning all color-controlled lights red and flashing all power-controlled lights to get occupants' attention and alert them to the impending context switch.

The system should provide a visible indicator in every room to signal whether TX or RX are currently in progress. Similar to the previous point, occupants should be able to tell at a glance whether their audio is currently being transmitted, even if they are not carrying a phone. Our system design specifies that a wirelessly controllable LED indicator light array should be placed or mounted in convenient and visible locations in each room. Two small red LEDs placed next to the light switch should provide sufficient visibility yet be non-intrusively bright.

The system should stay as local possible. When streaming information as sensitive as audio and light control, keeping the application local and distributed is safer than sending it to the cloud. Point-to-point communication better protects against leakage of information to potentially malicious or untrustworthy third parties, and also make the communications more difficult to intercept than client-to-cloud communication. The vast majority of the ghosting architecture can be kept local. However, to ensure global uniqueness of usernames the application may need to employ a DNS-like registry that maps usernames to a front-end application node located in each home.

3.3 Supporting Infrastructure

Several hardware systems are required to support the application. Each participating room requires at least one microphone, speaker, programmatically-controlled light, and TX/RX indicator light. In addition, the locally-running application logic requires a publicly accessible front-end node or home gateway that can receive incoming calls from other houses. This node is also responsible for synchronizing context with the remote home. Each room may also require an additional controller to handle the room's audio processing.

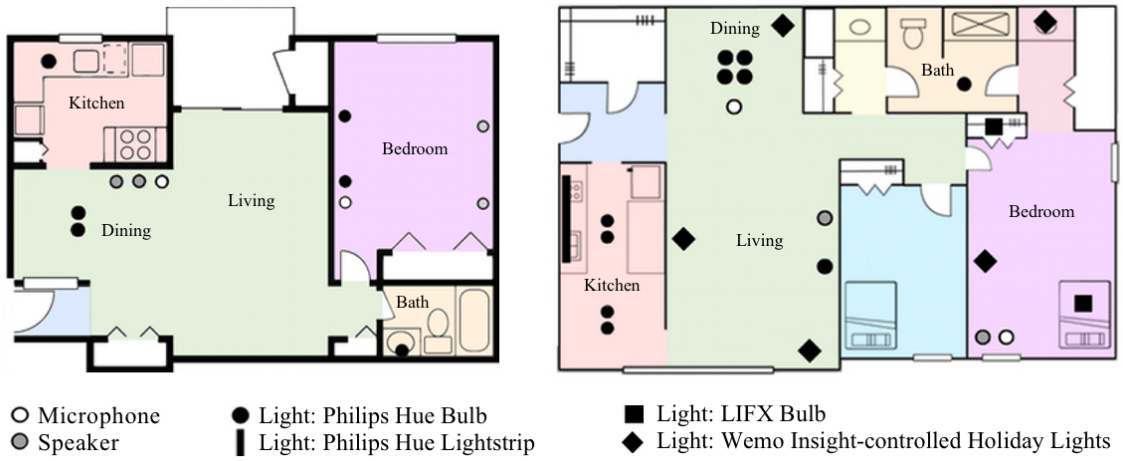

Figure 1: Deployment 2 device layout. The left figure shows the Maryland apartment floorplan, and the right figure shows the Michigan apartment floorplan. The Maryland apartment contains six color-controllable Philips Hue lightbulbs, two microphones, and four speakers. The Michigan apartment contains thirteen color-controllable lights – ten Philips Hue lightbulbs, one Philips Hue lightstrip, and two LIFX bulbs – as well as five holiday light strands that are power-controlled by Wemo Insight relays. The Michigan apartment additionally contains two microphones and two speakers. While Deployment 2 only synchonized lights and audio in the living rooms and master bedrooms, the remaining rooms could easily be light-synchronized as well.

There should be two main interfaces to the ghosting system. The first is the configuration dashboard, which would allow occupants to configure long-term settings for the system, and which should be available as a webpage on the local area network. The second is the call interface, which would allow occupants to make calls, toggle TX/RX settings, and hang up. The call interface should be available as both a local webpage on the LAN and also as a phone app, so that users can initiate or accept house calls remotely. The local webpages and synchronization between the phone app and the house can be handled by the same device that is acting as the front-end application node or home gateway.

3.4 Setup and Configuration

The setup and configuration process for this system should be as easy as possible for end users. The user should begin the initial setup by ensuring that all of the microphones and speakers and controllable light bulbs are in place in each room. Then the user will plug a "magic box" gateway device into their router to act as the front-end application node and provide the LAN-based interfaces. The gateway will display its IP address on an embedded display. The user visits that IP by typing it into their browser, and is brought to the configuration dashboard. There they select a globally unique home username. House calls made to that username will be mapped by a global registry to their home's gateway. Once the user has registered, the gateway automatically discovers the controllable lights, speakers, and microphones in the house and displays them to the user. Each device name is paired with a button that says "blink" or "chirp." The user employs these functions to discover which room a device is in, and then selects that room from a drop-down list of rooms (with the option to add a custom room). Once the devices have been labeled with their room, the initial setup process is complete.

A second configuration process occurs whenever a new home is added to the gateway's contact list. This configuration step allows the occupants of the two homes to jointly determine which rooms are paired with the rooms in the remote house. This process can be streamlined or even completely automated by selecting intelligent defaults. An obvious heuristic is to map rooms with the same labels to each other. This would allow many configurations to be done without any user input. However, the system should provide the option for more fine-grained house-to-house configuration, such as rearranging the room mapping, disabling entire rooms, or allowing specific lights to be mapped to each other. This will give users the ability to customize their experience to suit their particular setup.

4. IMPLEMENTATION

We carried out two deployments to determine whether ghosting could support the core user experience that we envisioned. The first deployment focused on the audio experience, and the second focused on the multi-room experience, incorporating both audio and real-time light synchronization.

While it is ultimately important that the infrastructure required for ghosting is not too costly or burdensome to maintain, for this work we did not try to optimize the infrastructure in terms of cost, power, or convenience. We also did not implement user interfaces for setup and configuration, nor did we include the various alerts and indicators that would be necessary in a consumer-facing application. The deployments described in the following sections were designed to incrementally explore different aspects of the ghosting experience.

4.1 Deployment 1

Deployment 1 was a multi-day study to evaluate whether it is possible to merge the ambient sound domains of two different rooms and what the psychological experience of sharing an ambient sound domain with occupants in another room is like. Our deployment took place in a fourth-floor lab and a second floor office that had ten and four residents respectively during the study period. We paired the audio between the two rooms by establishing a conference call and left it running for three days.

The conference call software ran on a desktop computer in the lab and an Ethernet-connected laptop in the office. We explored multiple conference call programs to compare sound compression and feedback handling. Each conference call endpoint included an external USB microphone and wired speakers. We upgraded the original webcam-quality Logitech microphones to studio-quality

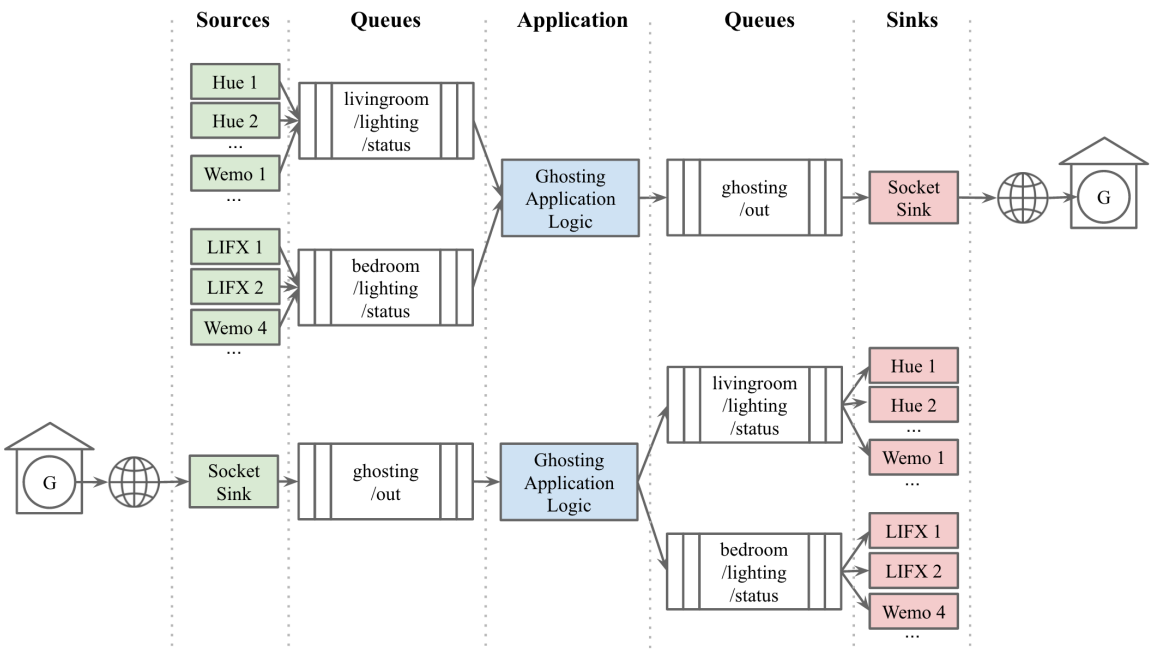

Figure 2: Dataflows for the light synchronization backend in Deployment 2. We created a custom message-oriented stream processing backend that generates message streams from devices using driver-like *sources* (in green), and controls devices by sending messages to driver-like *sinks* (in red). Messages populate queues (in white), which sources, sinks, and application logic blocks can subscribe to. The top dataflow shows how lighting state is collected from devices, processed, and shipped to the remote house's front-end application node (usually running on the home gateway, G). The bottom dataflow shows lighting state messages being received from the remote house and translated into local control actions. The use of messaging queues allows the application logic to be decoupled from the specific deployment, and also allows different brands of devices with different capabilities to respond appropriately to the control signals. During Deployment 2, the same application logic ran in both apartments despite substantial differences in their lighting setup.

Blue Yeti microphones due to fidelity issues, which we address in later sections. The lab speaker setup was 7.1 surround sound, and the office had a 2.1 speaker setup. Initially the office speakers were arrayed side-by-side, but we rearranged them to be surround sound to improve the sense of immersion.

Because Deployment 1 involved a large number of occupants talking at the same time, often in locations far from the microphone, we were able to quickly discover the critical impact that microphone quality has on room-level audio synchronization. We were able to use this knowledge to bootstrap Deployment 2.

4.2 Deployment 2

Deployment 2 explored the dynamics of multiple pairs of rooms with synchronized audio and light control. The deployment involved two apartments: a two-bedroom apartment in Michigan, USA, and a single-bedroom apartment in Maryland, USA. In each apartment, the master bedroom and the living room were paired with the corresponding room in the other apartment. Figure 1 shows the device layout for each apartment.

4.2.1 Audio Synchronization

In the Michigan apartment, each room's audio setup employed a WiFi-connected laptop hooked up to a Blue Yeti USB microphone. The bedroom relied upon the laptop's embedded speakers, while the living room sound was supplied by connecting the laptop to the TV's Sonos Playbar over HDMI. In the Maryland apartment, one laptop on WiFi and one desktop on Ethernet were responsible for running the conference call software in the bedroom and living room respectively. The bedroom microphone was a studio-quality CAD GXL2200, and the living room microphone was a USB Logitech HD Webcam C310. Audio synchronization was done using a Skype call in the bedrooms and a Google Hangouts video chat with video disabled in the living room.

4.2.2 Light Synchronization

The Michigan living room contained five Philips Hue bulbs and three Wemo Insight-controlled holiday light strands. The bedroom contained two LIFX bulbs and two Wemo Insight-controlled holiday light strands. The Philips Hue and LIFX bulbs are capable of color, brightness, and power control, whereas the Wemo-controlled lights can only be turned on and off. In the Maryland apartment, both the living room and bedroom contained two Philips Hue bulbs.

To synchronize the lights between the two apartments we created an application on our own custom stream processing platform, which we call Dolittle. The application dataflow architecture is illustrated in Figure 2. Dolittle is meant to be lightweight, local, and distributed, with an emphasis on real-time reactive control and learning applications. A major goal of the platform is to decouple application logic from specific deployment setups for maximum portability.

The basic components of Dolittle are *blocks*. Each block has input queues and output queues, and a "process" function that is called on any message that is placed into the input queues. There are three kinds of blocks: 1) *sources*, which talk to devices or external virtual data sources and generate a stream of messages; 2) *processors*, which generate new message streams based on input messages (the application logic block in Figure 2 is an example of a processor); and 3) *sinks*, which take command messages and convert them into actions for devices or external virtual resources. When a block is instantiated, it is assigned one or more input queues, and one or more output queues that any emitted messages may be sent to. An emitted

message can be sent to a particular output queue based on matching the message type field to part of a queue name, or the message will be sent to all output queues.

Each block is a stand-alone process that can run on any network-connected device, so long as the message broker managing the queueing system is network-accessible. We implemented our system using the MQTT messaging protocol. We used the Mosquitto MQTT message broker and the client blocks were all written in Python using the Paho MQTT bindings [1]. While the block graph could be sharded across multiple devices, in each apartment we ran all of the blocks on a Raspberry Pi, which also acted as the apartment's home gateway. Communication between the front-end application nodes running on the two home gateways was done using TCP sockets.

The ghosting application logic was straightforward. If the application block received a light change event from a room in the remote home, then the application block would emit the matching command to the lights in the corresponding local room. The application block also constantly monitored a stream of status messages from the local lights. When a change was detected that could not be attributed to matching the remote home, then the application emitted that light change event to the remote home. Since we designed Dolittle with application portability in mind, the same application logic ran at both apartments.

5. EVALUATION

For each deployment, we frame the evaluation with two guiding questions: Did it feel like the remote person was present? Did the interaction feel natural? We use these two questions to explore the engineering challenges behind supporting the user experience.

5.1 Deployment 1

5.1.1 Did it feel like the remote person was present?

Once the sound system was configured properly, it sounded as though the lab occupants were present in the office. However, there were several issues that needed to be addressed in order to achieve proper configuration.

The first issue was sound quality. We began the study with webcam-quality Logitech microphones. However, while those microphones would be more than suitable for conference calls, we had unique sound requirements – we wanted to capture all the sounds in a room, not just human voices close to the microphone. We upgraded to studio-quality Blue Yeti microphones, which were able to capture distant voices and non-human vocal sounds, such as squeaking chairs and paper shuffling.

We also compared different conference call software with regards to sound compression quality and feedback prevention. Google Hangouts had trouble adjusting to the feedback from speakers, perhaps due to the unusual external speaker and microphone placement. Jitsi handled the sound compression and feedback well, but is proprietary and difficult to run locally on a headless device. BigBlueButton is an open source project which runs locally, and it handled non-human voice audio and feedback decently. We used Jitsi for the majority of the study due to its ease of use. However, in the future open source teleconference software like BigBlueButton that can be run on headless devices would be ideal.

The second issue concerned sound directionality. Originally the office speakers were placed together next to the laptop, but because the sound was emitted from a point source, it did not feel as though the lab occupants were in the office. We rearranged the speakers into surround sound positions, and afterwards it sounded as though the remote occupants were actually in the room.

5.1.2 Did the interactions feel natural?

The biggest issue with interaction was that it was not possible to tell at a glance who was "in the room," so if an occupant wanted to talk to a remote person it was difficult to coordinate without first calling out the person's name a waiting for a response, which did not feel natural or casual. The presence of multiple occupants in each room made it more difficult, since hearing the sounds of occupancy did not mean the person you wanted to speak to was present. However, once the intent to communicate was established, communication felt natural.

5.2 Deployment 2

5.2.1 Did it feel like the remote person was present?

Multiple synchronized rooms improved the experience of remote presence. It was possible to hear the remote occupant walk between two synchronized rooms due to gradually changing sound attenuation. Hearing the remote occupant's voice emanate from another room also felt natural. Additionally, the remotely synchronized lights established a shared physical context in a tangible and immediate way. Talking to the remote occupant while they toggle "your" lights feels like a personal and intimate interaction.

5.2.2 Did the interactions feel natural?

In Deployment 2, the lack of visual indicators of presence was again an issue. Because multiple rooms were synchronized, if the two apartment occupants wanted to converse, they had to first shout the other person's name and listen for a response. However, the remote occupant's response was heard coming from the room they were in, which felt the same as if the person were actually in another room in the apartment.

Another interesting result from the lack of a visual presence was what we call a "rubber duck" effect. When speaking to a remote occupant, the local occupant will frequently visually fixate on an object and speak to it as a stand-in for the real person. This does imply that some visual indication of a person's remote presence may be necessary for naturalistic interactions.

6. DISCUSSION

One of the most critical lessons learned from the experiments was the importance of an at-a-glance visual indicator of presence and identity of room occupants. To address both this and the related "rubber duck" effect described in the previous section, we propose that remote occupants be embodied as light. When a remote occupant enters a synchronized room, a local light should turn to a color associated with their identity and "breathe" by pulsing at the rate of human breath. When the remote occupant speaks, the local light should flicker as though the light were speaking. As the remote occupant moves from room to room, the local light will move between the corresponding rooms. We prefer this solution because it leverages infrastructure that already exists to support the core ghosting application, indicates occupancy and identity, and provides a visual embodiment of the remote occupant that the local occupant can talk to in lieu of the real person.

6.1 Future Technical Challenges

Because ghosting is an application that requires a full operational stack to function correctly and securely, it is a good application to provide direction to foundational IoT research efforts.

For this application to be commercially viable it is critical to determine how to get the most immersive sound experience with minimal hardware. The hardware should be non-intrusive and affordable, yet the microphone must be of a reasonably high quality

to capture ambient sounds. Additionally, sound directionality, which can greatly enhance the illusion of presence of remote occupants, generally requires at least two speakers per room. However, the cost of instrumenting each room with a minimally sufficient audio setup is likely to be on par with the cost of an entertainment system, and if the underlying platform is open and emphasizes reusability and interoperability, then the same audio setup could actually be used for entertainment purposes or other IoT applications.

Making the ghosting hardware available to other applications requires exploring the design space for application platforms and middleware. The platform should be lightweight, run headlessly, support interoperability, and be able to run multiple coexisting applications. At the same time such a platform also needs to provide a high level of security. While many attempts have been made to design operating systems and application platforms for buildings, no definitive architecture has yet emerged.

The ghosting application itself faces several fundamental challenges due to the heterogeneity of building layouts. The first is how to handle room mapping edge cases, such as mapping multi-room homes to studio apartments, big open floorplans, and mappings that destroy the spatial continuity of sound attenuation. In the initial stages of these kinds of applications, we suggest providing users enough powerful configuration options that if they experience these edge cases they can correct the configuration or develop a workaround on their own. In the future, however, a more systematic understanding of the topological structure of buildings may prove useful for both this application and others.

The final challenge is how to implement and evaluate the user experience. The solution we propose in the previous section for visually representing remote occupants requires room-level occupant tracking, which is still an open research problem. Additionally, in our design we propose user interfaces for the configuration process and call alerts, but the effectiveness of these proposed elements remains to be explored. Some occupants may avoid systems like these as being too "creepy," so it is also important to study what features influence psychological acceptance of the system. Currently it is logistically difficult and expensive to perform statistically significant user studies in smart homes. The IoT community should consider establishing a large-scale smart home testbed program in the style of PlanetLab for evaluating emerging applications and services.

7. CONCLUSIONS

Smart homes present opportunities to socialize across long distances in ways that traditional communication media cannot support. In this work, we propose a novel intimate computing application called ghosting, which enables people separated by distance to feel present in each others' homes and daily lives. The outcomes of our two deployments suggest that this application supports new modes of casual yet intimate social interactions that conversation-oriented and context-independent technologies cannot capture. We believe that smart homes have a positive contribution to make to technologically-assisted social interaction, and hope that this work opens up exploration into the space of social and messaging apps for the Internet of Things.

8. ACKNOWLEDGMENTS

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA. The work was also supported by the National Science Foundation Graduate Research Fellowship. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] http://mosquitto.org. Accessed: 2014-04-09.

[2] G. Bell, T. Brooke, E. Churchill, and E. Paulos. Intimate ubiquitous computing. In *Proc. UbiComp Workshop*, pages 3–6. Citeseer, 2003.

[3] S. Bhandari and S. Bardzell. Bridging gaps: affective communication in long distance relationships. In *CHI'08 extended abstracts on Human factors in computing systems*, pages 2763–2768. ACM, 2008.

[4] M. Black, K. Dana, K. Gaskins, C. Gaynier, A. Lemieux, and K. McKinley. The internet of things: Can it find a foothold with mainstream audiences today? Technical report, The Nielsen Company, 11 2014.

[5] M. Brereton, A. Soro, K. Vaisutis, and P. Roe. The messaging kettle: Prototyping connection over a distance between adult children and older parents. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 713–716. ACM, 2015.

[6] H. Chung, C.-H. J. Lee, and T. Selker. Lover's cups: drinking interfaces as new communication channels. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 375–380. ACM, 2006.

[7] C. Dodge. The bed: a medium for intimate communication. In *CHI'97 Extended Abstracts on Human Factors in Computing Systems*, pages 371–372. ACM, 1997.

[8] P. Dourish and S. Bly. Portholes: Supporting awareness in a distributed work group. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 541–547. ACM, 1992.

[9] W. Gaver, T. Moran, A. MacLean, L. Lövstrand, P. Dourish, K. Carter, and W. Buxton. Realizing a video environment: EuroPARC's RAVE system. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 27–35. ACM, 1992.

[10] S. King and J. Forlizzi. Slow messaging: intimate communication for couples living at a distance. In *Proceedings of the 2007 conference on Designing pleasurable products and interfaces*, pages 451–454. ACM, 2007.

[11] A. Lella and A. Lipsman. The US mobile app report. 2014.

[12] A. Lenhart, K. Purcell, A. Smith, and K. Zickuhr. Social media & mobile internet use among teens and young adults. millennials. *Pew Internet & American Life Project*, 2010.

[13] A. P. Mathew and J. Taylor. Chi'08 alt. chi/auralscapes: engaging ludic ambiguity in the design of a spatial system. In *CHI'08 extended abstracts on Human factors in computing systems*, pages 2533–2542. ACM, 2008.

[14] K. Nagel, C. D. Kidd, T. O'Connell, A. Dey, and G. D. Abowd. The family intercom: Developing a context-aware audio communication system. In *Ubicomp 2001: Ubiquitous Computing*, pages 176–183. Springer, 2001.

[15] C. Neustaedter and S. Greenberg. Intimacy in long-distance relationships over video chat. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 753–762. ACM, 2012.

[16] F. Vetere, M. R. Gibbs, J. Kjeldskov, S. Howard, F. Mueller, S. Pedell, K. Mecoles, and M. Bunyan. Mediating intimacy: designing technologies to support strong-tie relationships. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 471–480. ACM, 2005.