`ConnectedHearts`: A Distributed System for Viewing the Human Heartbeat

Serena Booth[a,1], Michelle Cone[b,1]

[a] *sbooth@college.harvard.edu*
[b] *mcone@college.harvard.edu*

Abstract

We present a physical distributed system for viewing the human heartbeat. We modified a medicine cabinet by embedding 13 light bulbs around the frame, as well as by replacing the cabinet's glass with one-way mirror. When a person stands in front of the mirror, a webcam hidden behind the mirror measures their pulse. The 13 bulbs, each a virtual machine, run Bully leader election. First the leader starts pulsating with the captured pulse, and then instructs neighboring virtual machines to pulsate as well. Finally, we ensure the synchronization of the bulbs through a distributed gossip algorithm.

1. Introduction

`ConnectedHearts` straddles the boundary between art and computer science. The piece is inspired by an exhibit shown around the world in modern art museums, but our re-implementation emphasizes distributed computing, as `ConnectedHearts` is a physical representation of a distributed system. We modify a retro medicine cabinet to hold 13 light bulbs around its frame.

[1] We, Serena Booth and Michelle Cone, affirm our awareness of the standards of the Harvard College Honor Code.

While in an ideal world each light bulb would be powered by its own microprocessor, we simulate this interaction instead by each light bulb representing a virtual machine as a process in effort to reduce expenditure. These light bulbs attempt to self-synchronize while displaying the human heartbeat.

In this paper, we discuss the components and construction of `Connected Hearts`, the architecture and algorithms powering the software, as well as analysis of our system.

1.1. Artistic Inspiration

This piece is inspired by Rafael Lorzeno-Hemmer's "Pulse Room,"[1] an artistic piece in which an entire room is outfitted with light bulbs. A physical heartbeat monitor is present in the room. A user approaches this monitor and grabs onto it. As soon as the conductance of the skin is felt, all light bulbs in the room turn off. Within a few seconds, the heartbeat monitor detects a pulse. With this, a single bulb directly above the user begins to pulsate with their heartbeat. This message is then spread to neighboring bulbs, as well as bouncing off of the walls; with this message-passing, the room becomes full of chaos. After a further few minutes, the bulbs synchronize, leaving the entire room pulsating with the heartbeat. We note that this system uses a single address-space to achieve this effect.

2. Components & Construction

We construct the piece from the following materials:

- 2 × Ubiquiti mPower Strips: 8-outlet power strips running Linux
- Network switch and ethernet cables

- 13 × filament bulbs and bulb holders
- Vintage Medicine Cabinet, one-way glass
- Raspberry Pi
- Raspberry PiCamera

The end product looks to be a medicine cabinet with 13 light bulbs around its mirror; the majority of the components—the cables, the Raspberry Pi, etc.—are hidden from view, inside the cabinet.

3. System Design

Our code is written in Python and makes extensive use of the multiprocessing library. Our process infrastructure is shown in Figure **??**. In short, the program runs a `main.py`, the parent process. This process creates one process for running pulse detection; two more processes for turning all the relays off at the start; and 13 more processes, each corresponding to a bulb. Of these 13 "bulb" processes, each spawns its own process to handle inter-bulb synchronization. Of these inter-bulb synchronization processes, each spawns an additional process which is tasked with maintaining the state of a continually-blinking bulb.

3.1. Inter-bulb Communication

We of course, do not assume the existence of a global clock. However, we do allow processes to communicate via shared memory. In particular, many processes share queues. This architecture is further shown in Figure **??**.

4. Architecture & Algorithms

4.1. Pulse Detection

When a face is visible to a camera hidden behind the one-way glass of the medicine cabinet, we detect the pulse of the viewer.

4.2. Bully: Leader Election

[SERENA: TO DO: Write about Bully]

We demonstrate this leader election mechanism in Figure 1.

4.3. Realtime Synchronization via Gossip

[SERENA: TO DO: Write about synchronization]

4.3.1. Physical Layout

The physical layout of our system is predetermined, as shown in Figure 2. Each bulb is assigned a fixed, position-based id as well as a generated UUID. Because of this fixed layout, we are able to personalize gossip for synchronization in order to ascribe neighboring processes with scores of *trustworthiness*. As the leader bulb initiates the heartbeat pulsation, the leader is the single most trustworthy bulb. This has implications for all other bulbs, however: as bulbs increase in distance from the leader, a known measure based on the system's fixed layout, their trustworthiness decreases. Thus when a bulb is receiving contradictory gossip from its two neighboring bulbs, it prioritizes the message which was sent by the bulb closer to the leader.

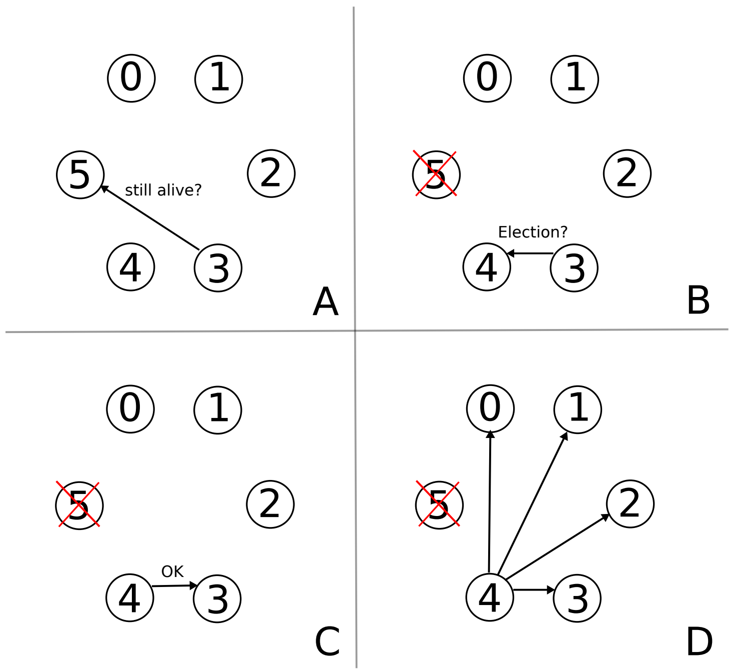

Figure 1: In the above graphic, we demonstrate the bully algorithm for determining a leader. In subimage A, node 3 attempts to confirm that the known leader, node 5, is still alive. In sumbimage B, node 5 does not respond to node 3 for a set period of time—a time set by node 3. Hence node 3 initiates an election by contacting all nodes with a higher id than itself. In subimage C, node 4 responds to node 3's request to initiate an election by confirming it remains alive. Finally, in subimage D, node 4 broadcasts a message to all nodes to inform them of its leader status.

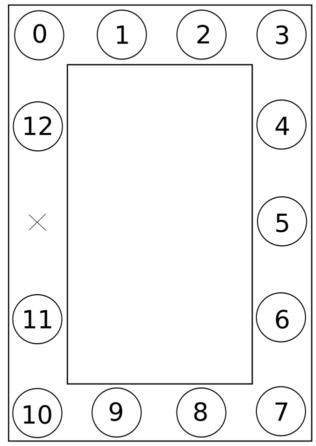

Figure 2: The system layout. While each bulb is assigned a 64-bit UUID during the program, each bulb also has an id, shown above, ranging from $0 - 12$, which allows the system to consistently converge toward the leader bulb's rate of pulsation during the synchronization routine.

5. Synchronization Testing & Analysis

In order to analyze the success of our synchronization via gossip, we implement a simple computer vision system. The steps of analysis are as follows:

1. We capture footage at a high-frame rate.

   - 120 fps; however, we note that this frame rate is slower than the processing rate of the human eye.

2. With GUI, present user with first frame of the captured footage.

   - User selects 13 pixels corresponding to bulb filaments.

3. Brightness timeseries computed.

   - The brightness of these pixels are averaged for every frame in the video, and ultimately we produce a timeseries, which we graph in comparison to idealized data. A demonstration of this result is shown in Figure 3.

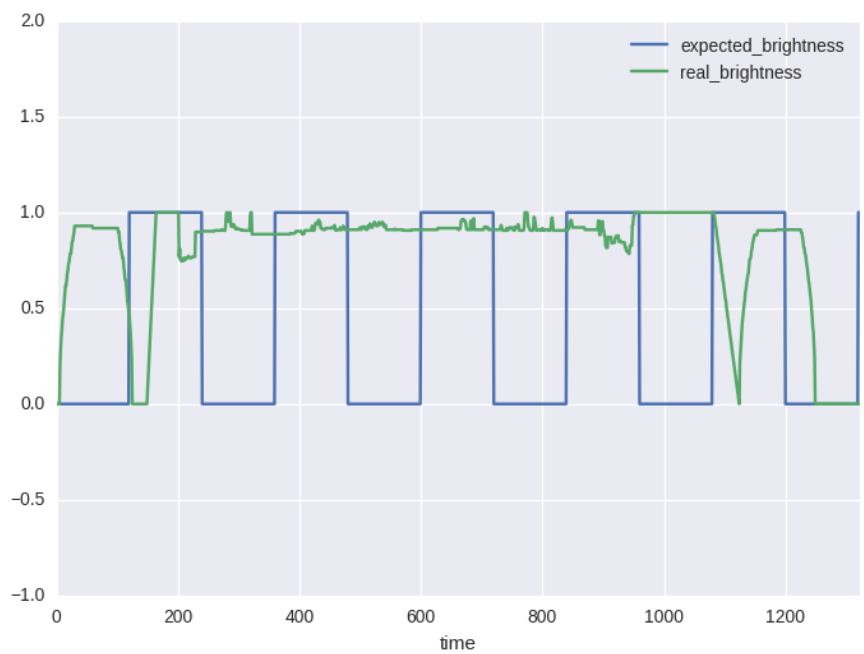

Figure 3: We demonstrate the synchronization of our system as a function of time, as compared with an ideal synchronization system. The green timeseries corresponds to the physical system while the blue timeseries corresponds to an idealized system.

6. Personal Learning

1. Debugging a distributed system is hard.

7. Conclusion

References

[1] R. Lozano-Hemmer, Pulse Room, Museum of 21st Century Art (Kanazawa), Coleccin/Fundacin Jumex (Mexico City), MONA Museum (Hobart), Jonathon Carroll Collection (NYC), and Museum of Modern Art (NYC). (2006).