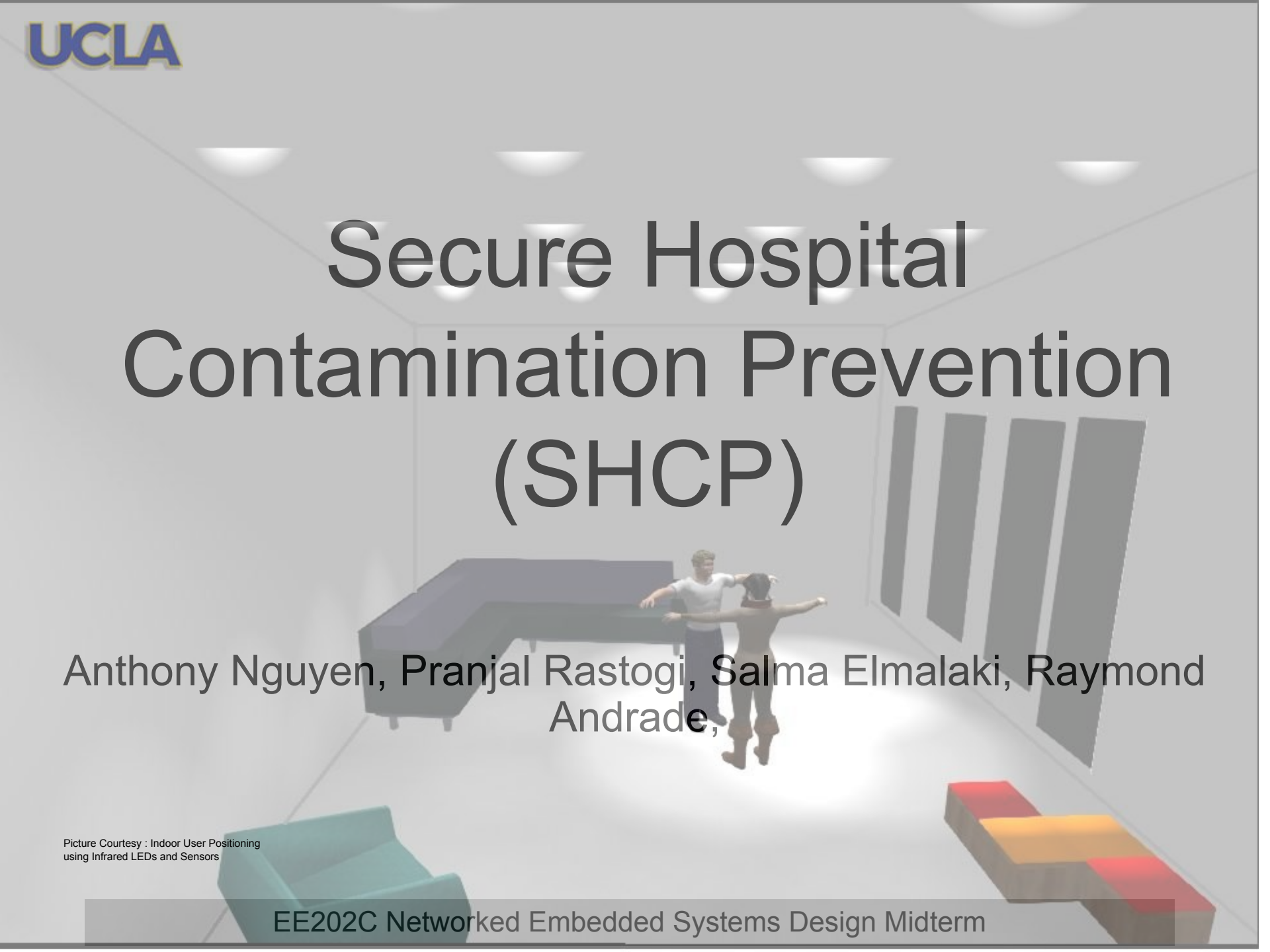
UCLA

Secure Hospital Contamination Prevention (SHCP)

Anthony Nguyen, Pranjal Rastogi, Salma Elmalaki, Raymond Andrade,



Picture Courtesy : Indoor User Positioning using Infrared LEDs and Sensors

EE202C Networked Embedded Systems Design Midterm

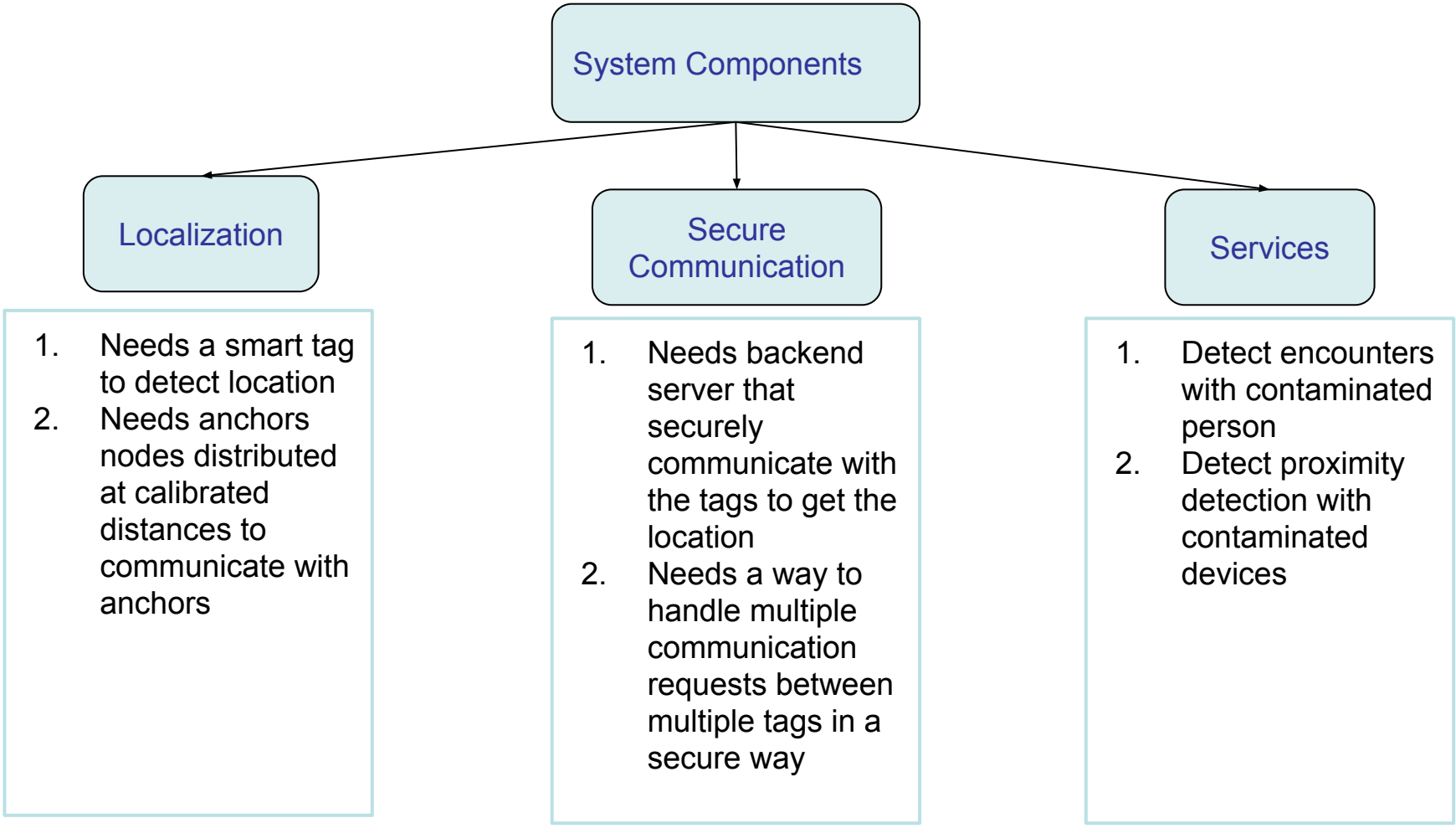
Product Development Mission Background

- Ensure fast, concentrated response to contagious disease exposure
- Provide protection to and from patients and visitors as well as employees
- Provide these and other services within a secure infrastructure

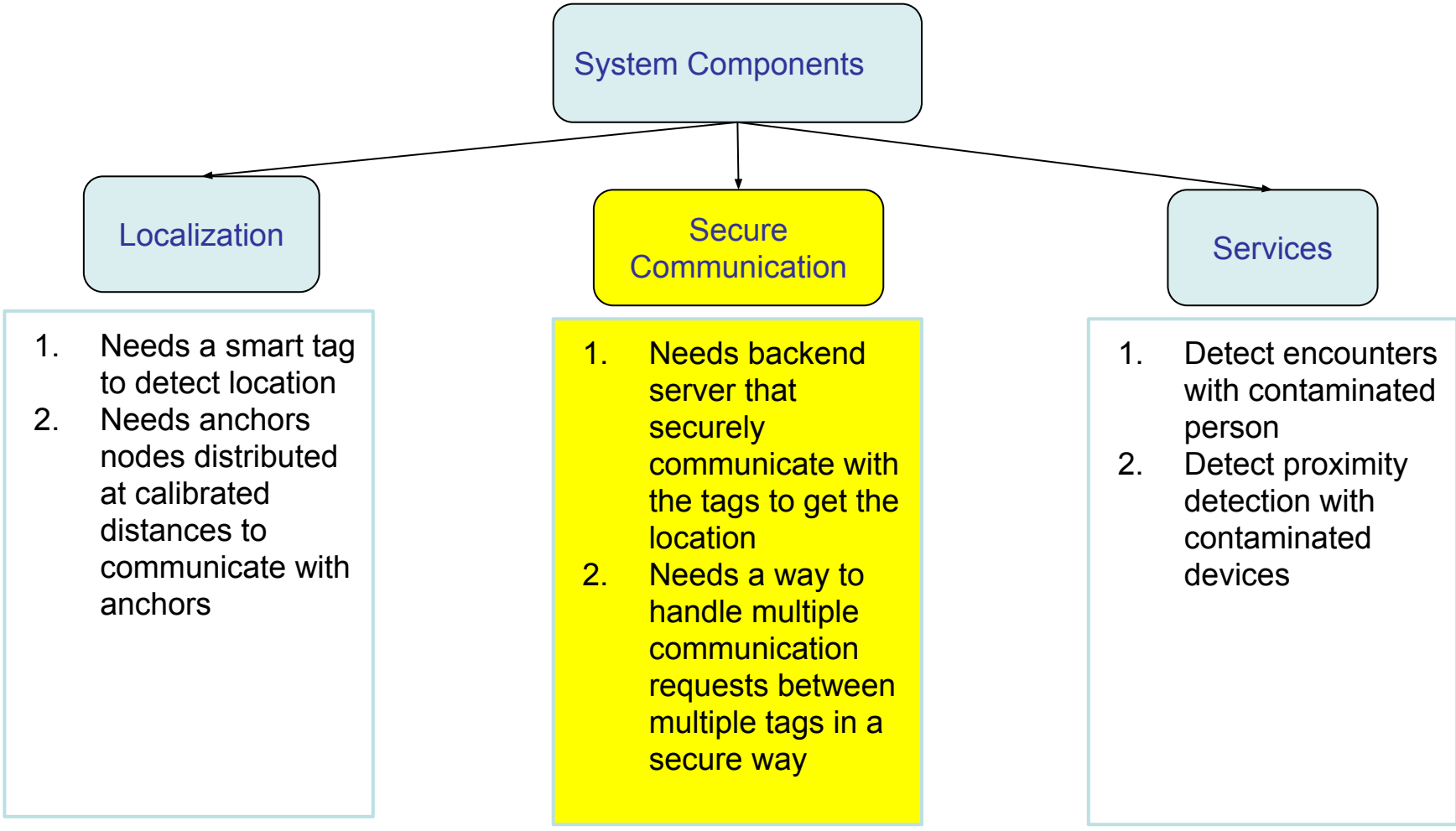
This can be achieved in three folds:

1. Localizing everyone in the hospital
2. Storing localization data
3. Handling the information securely

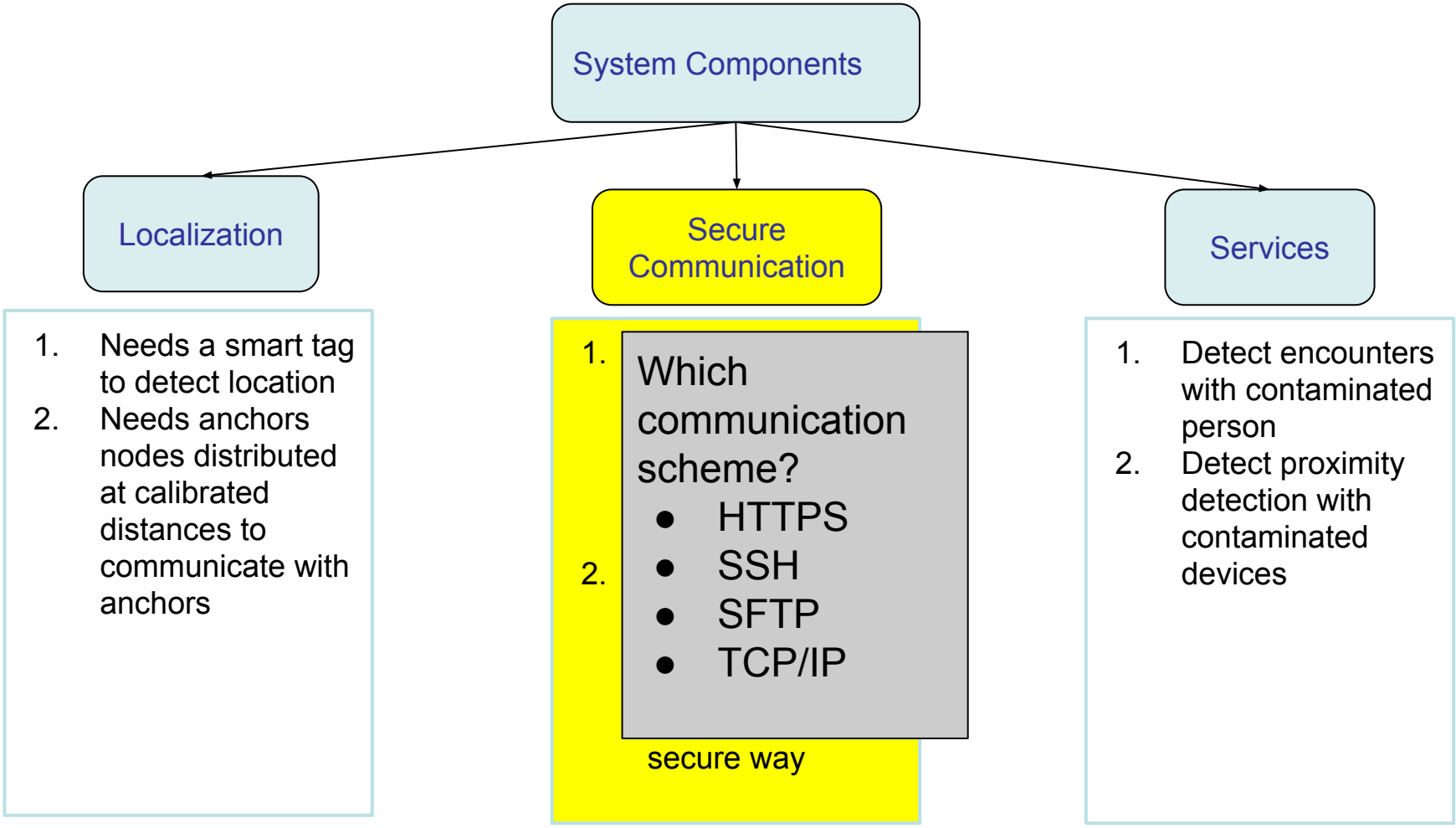
IoT approach for SHCP



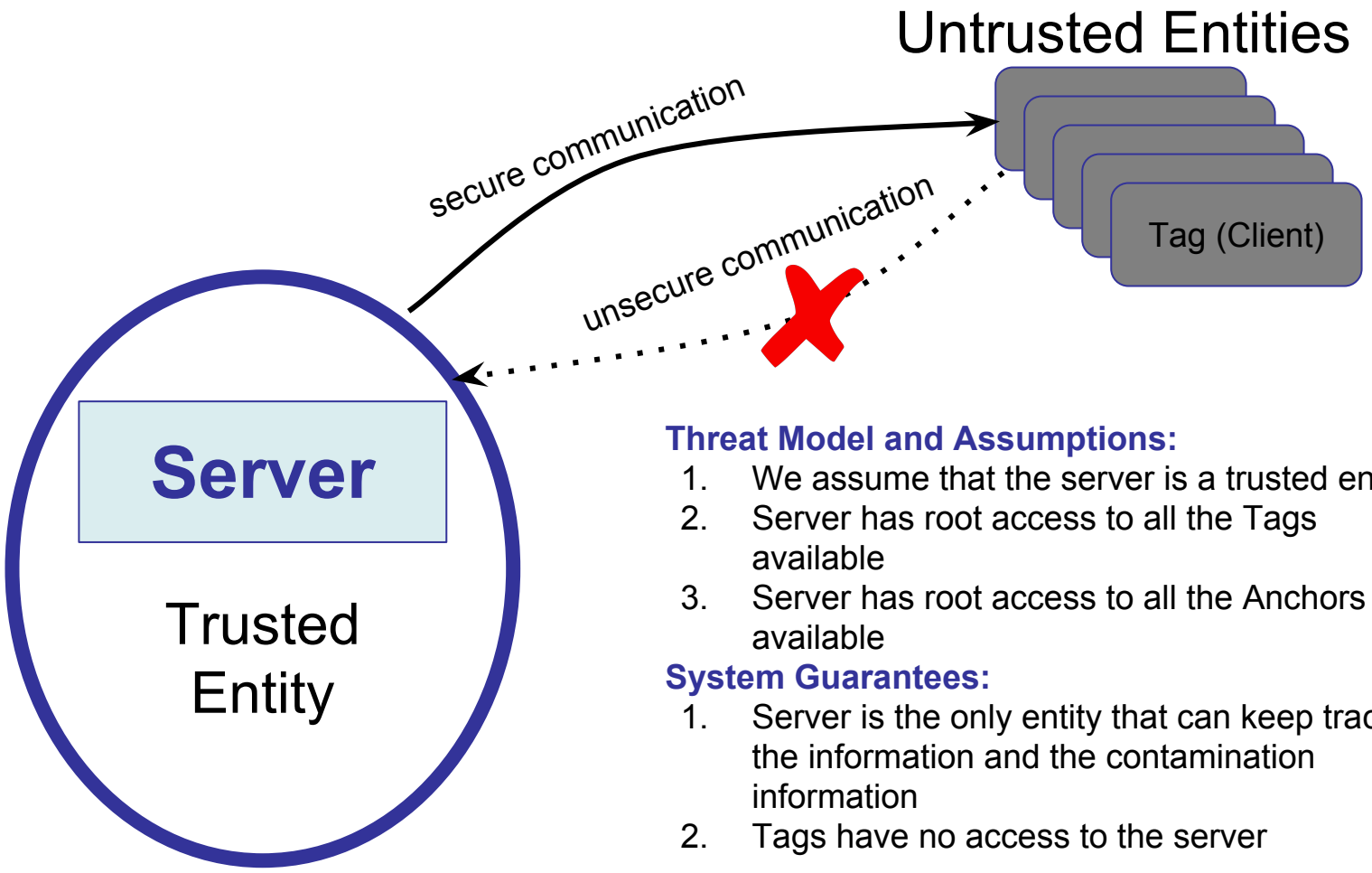
IoT approach for SHCP



Initial Challenges



Secure Communication



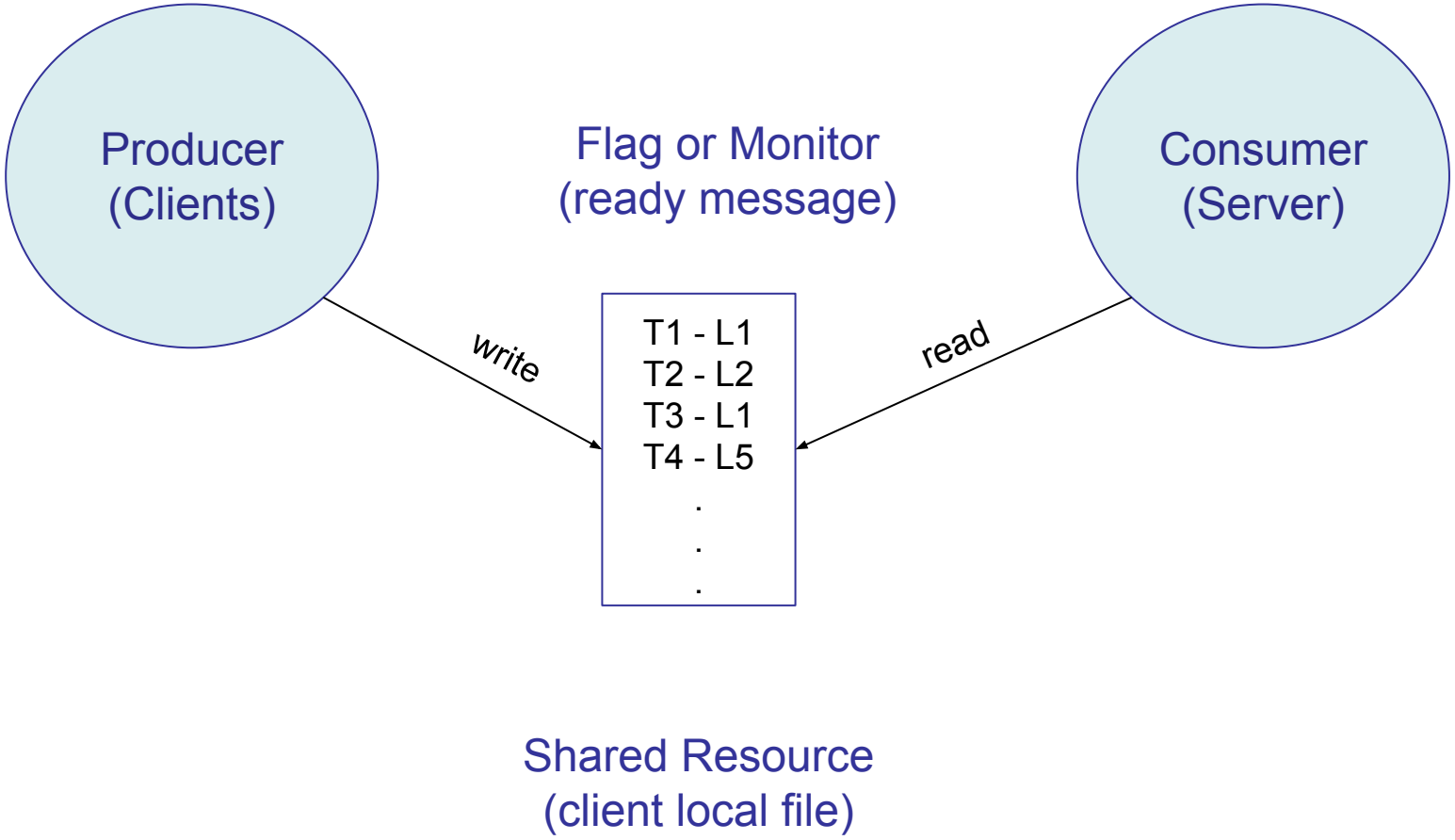
Threat Model and Assumptions:

1. We assume that the server is a trusted entity
2. Server has root access to all the Tags available
3. Server has root access to all the Anchors available

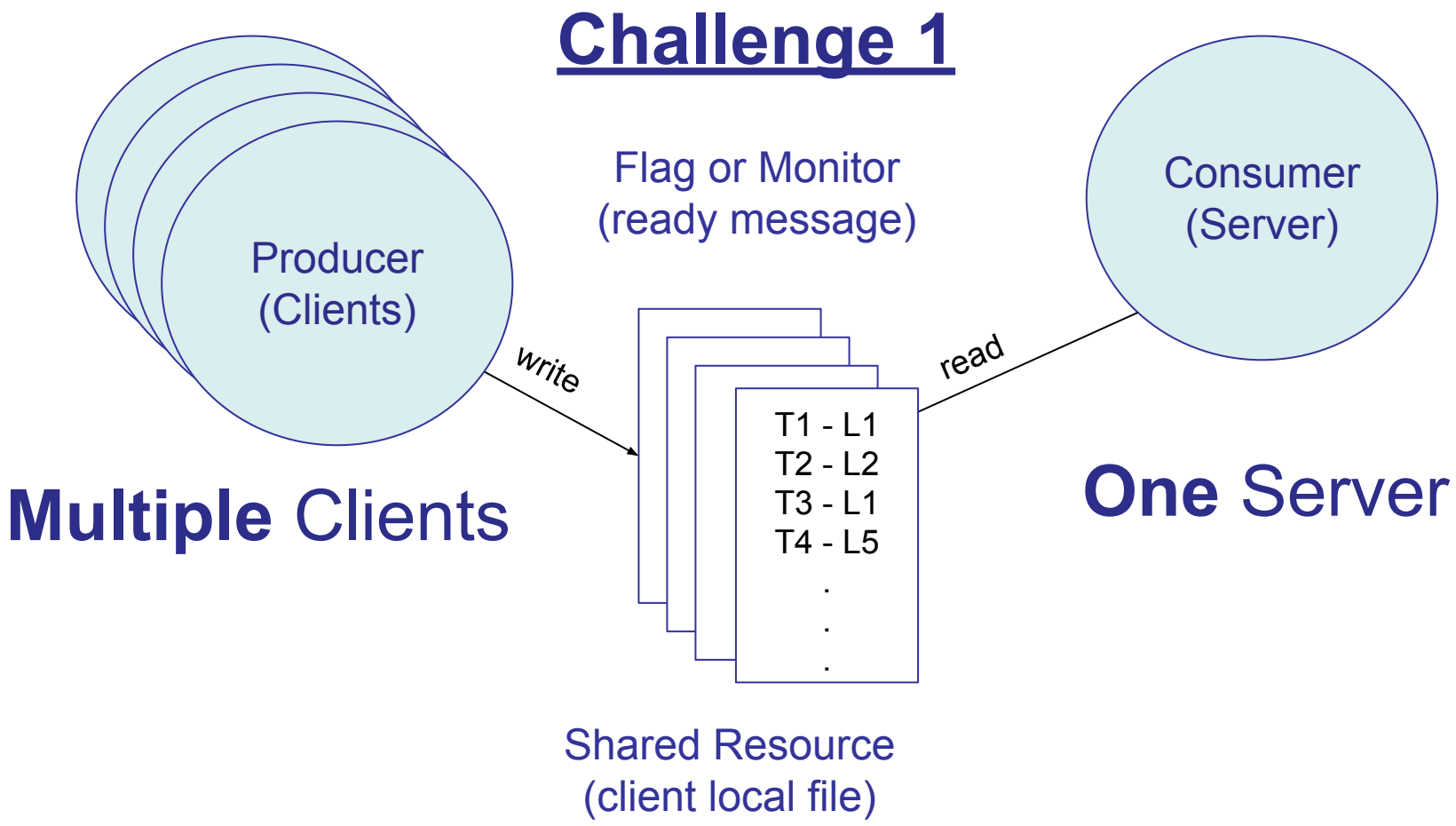
System Guarantees:

1. Server is the only entity that can keep track of the information and the contamination information
2. Tags have no access to the server

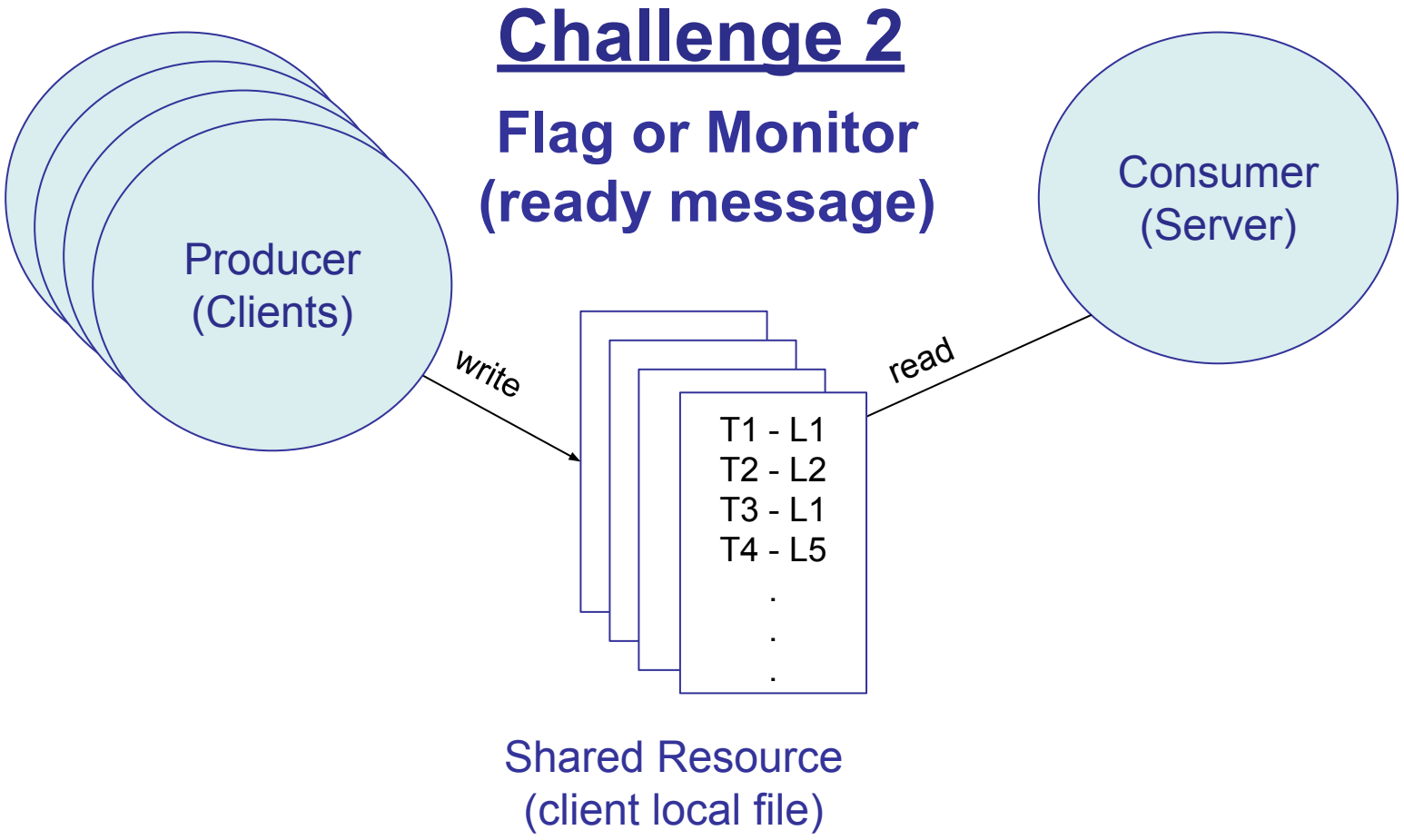
Design Overview
Flavor of "Producer-Consumer"



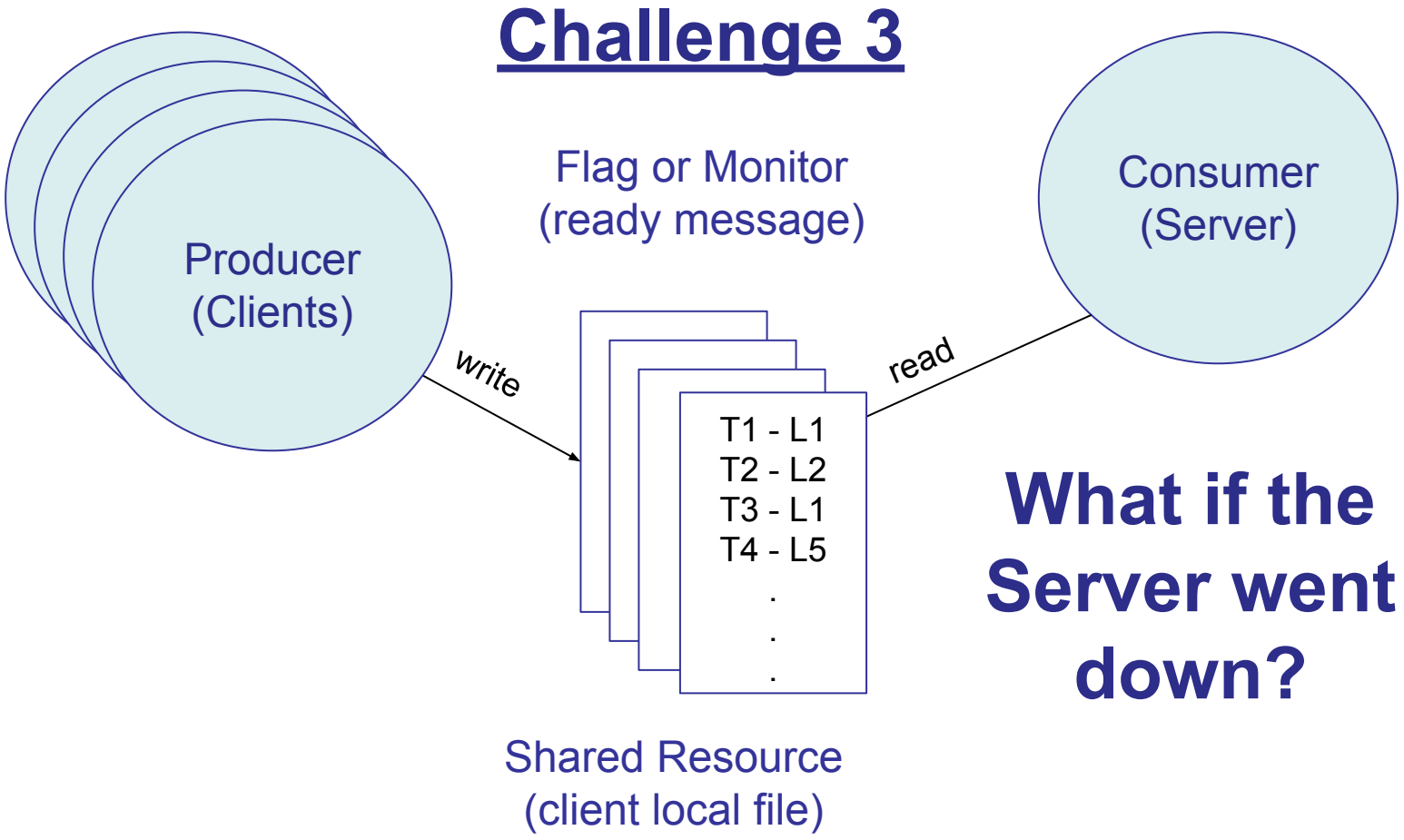
Technical Challenges



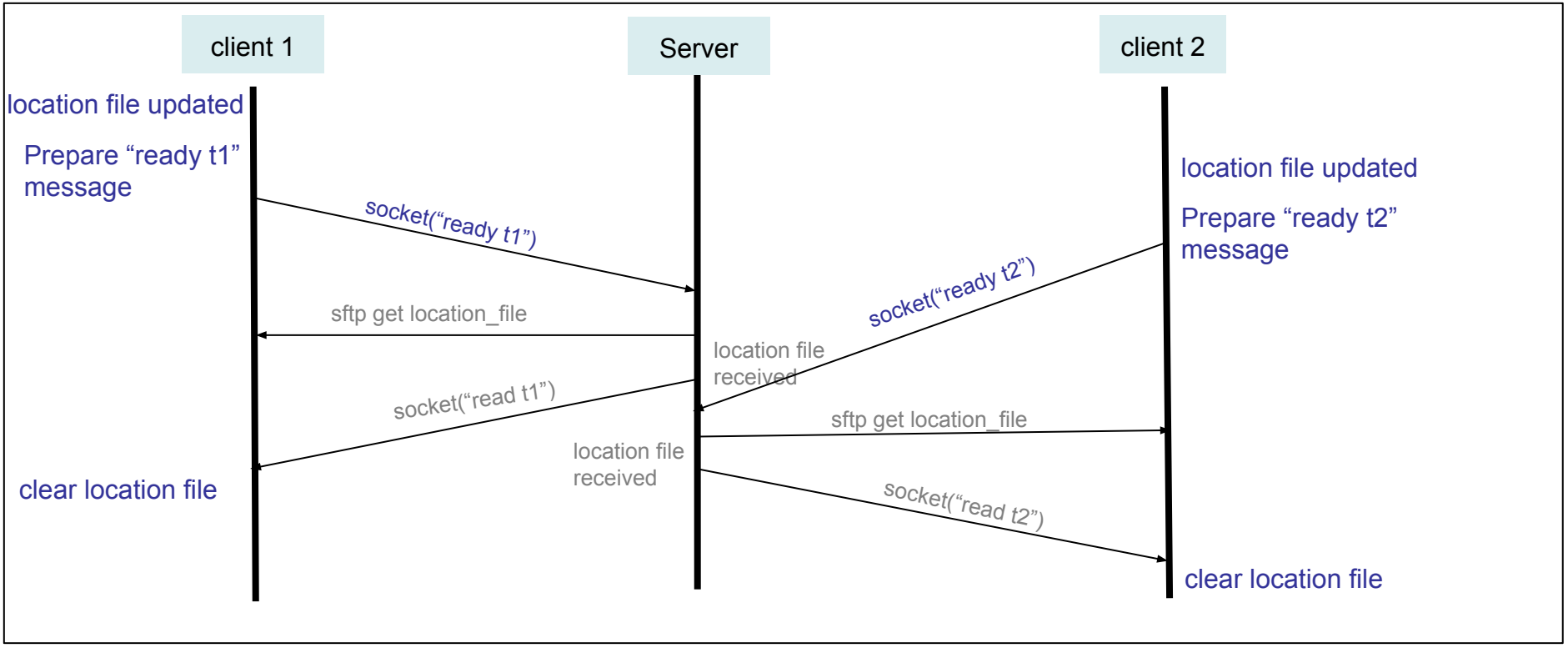
Technical Challenges



Technical Challenges



Secure Communication Design



Challenge 1: Threaded dispatch asynchronous Server

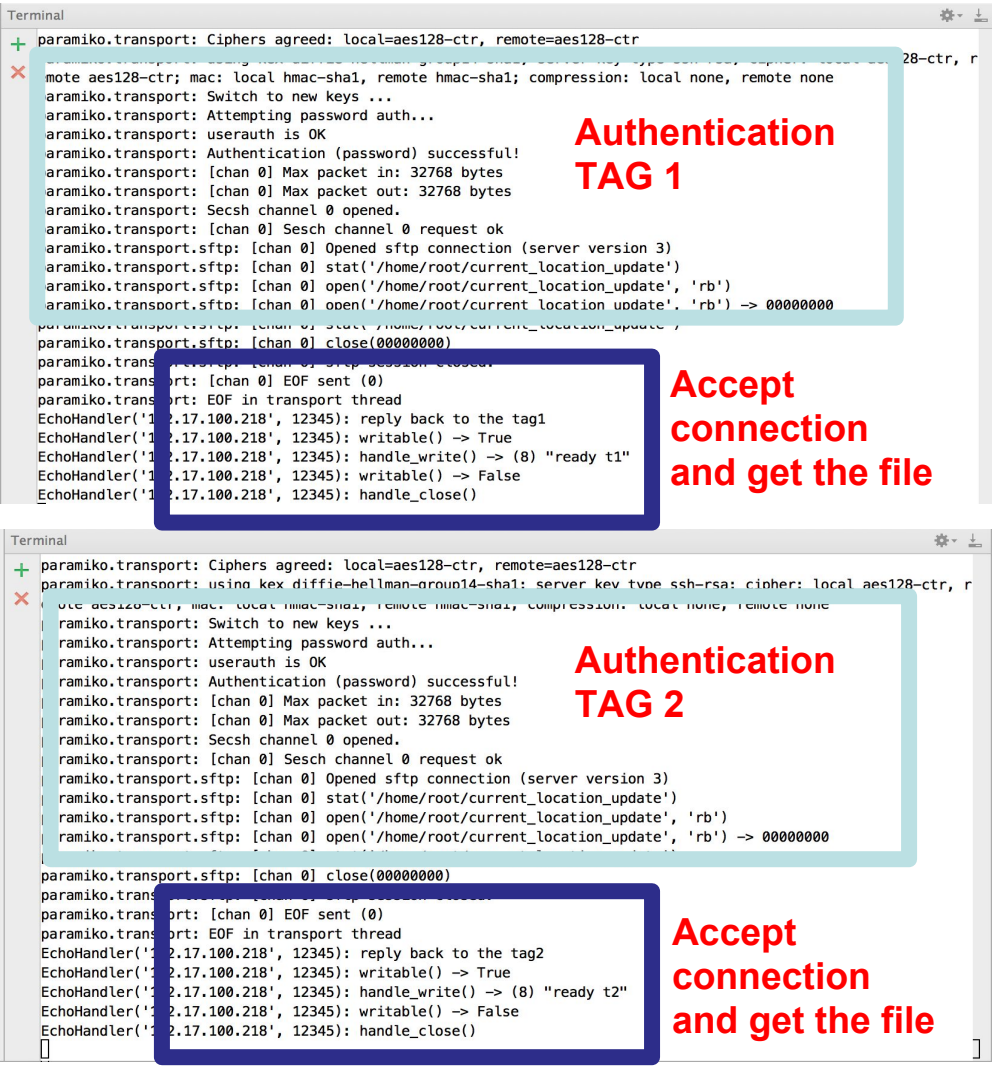
Challenge 2: Use SFTP/TCIP communication

Challenge 3: Update/Empty location file on 'Ready/Read’ message

System Implementation: Progress to Date

- Two tags (Clients) communicating with a backend Server to send their location with time stamp.
- Simulated location values are generated using sampling over list of locations.
- Server updates its database with the location for future queries
- **Video Demo!**

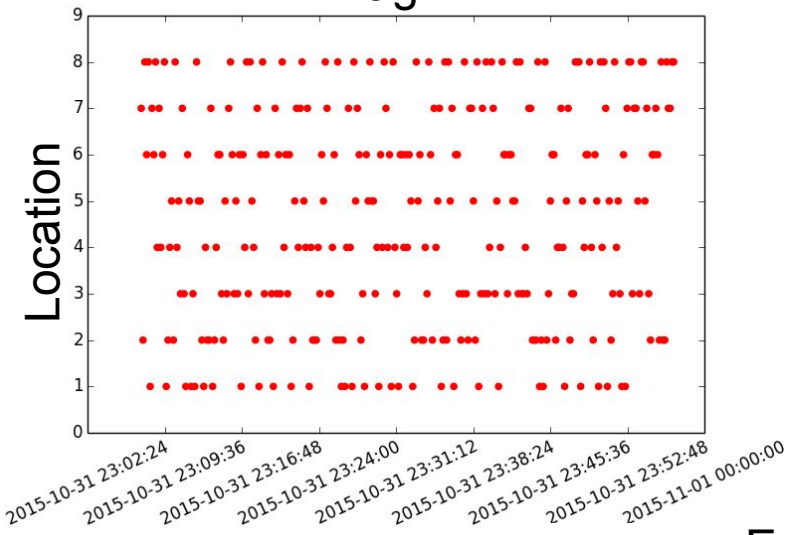
Testing and Verification



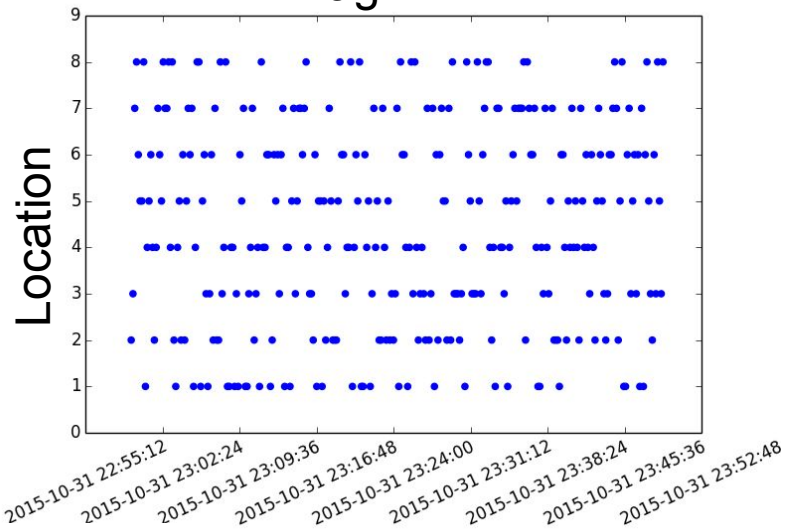
- Random location generated on Tag 1 and Tag 2 every 10 seconds
- Server gets the location from Tag 1 and Tag 2
- The server keeps the location of each tag for future queries
- Currently, we support offline analysis

Testing and Verification

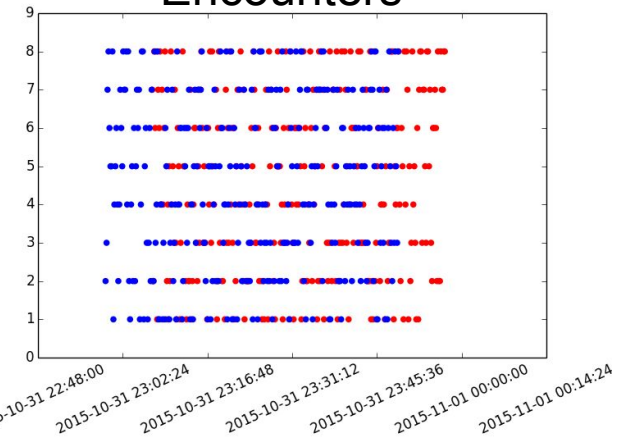
Tag 1



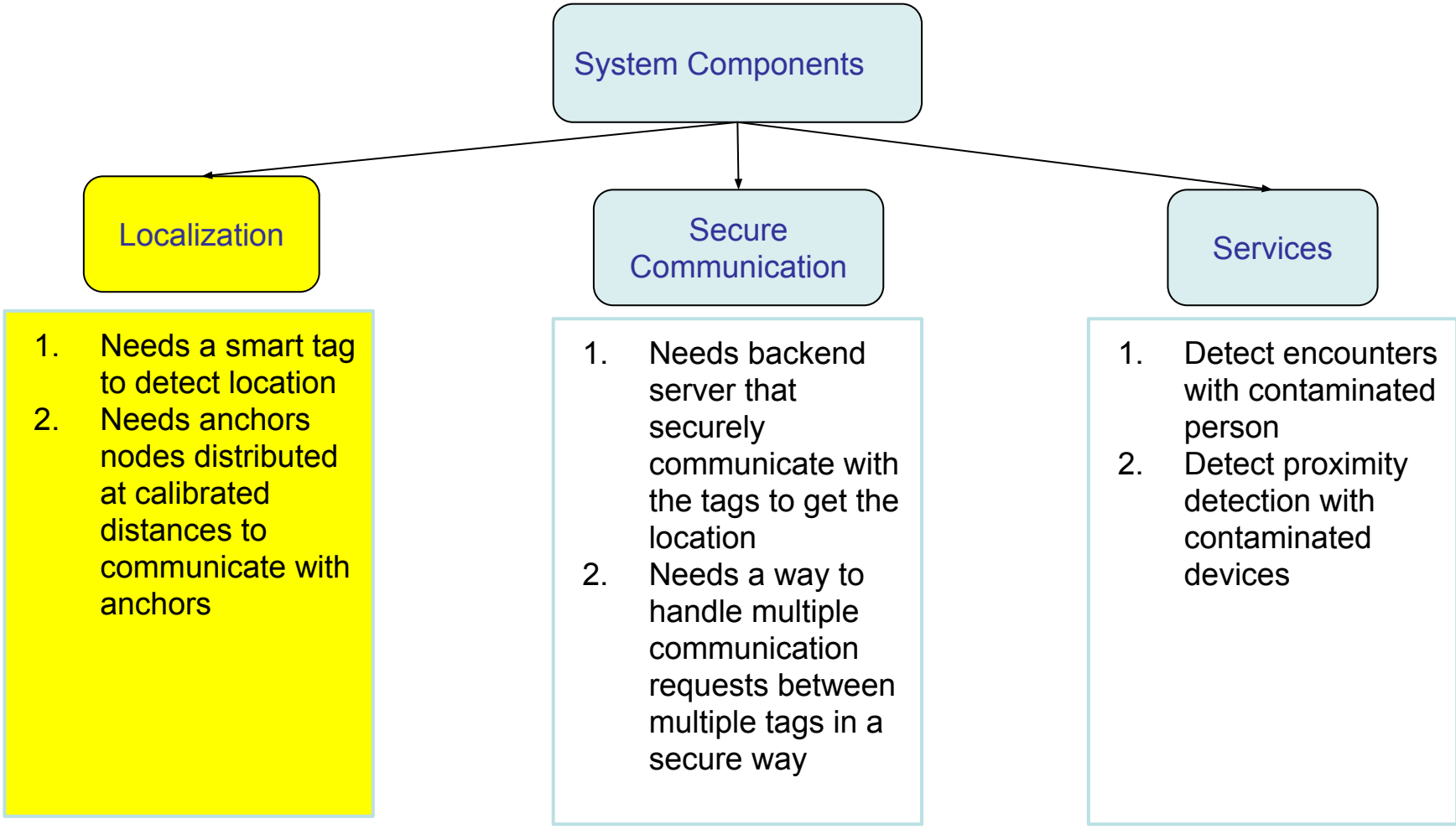
Tag 2



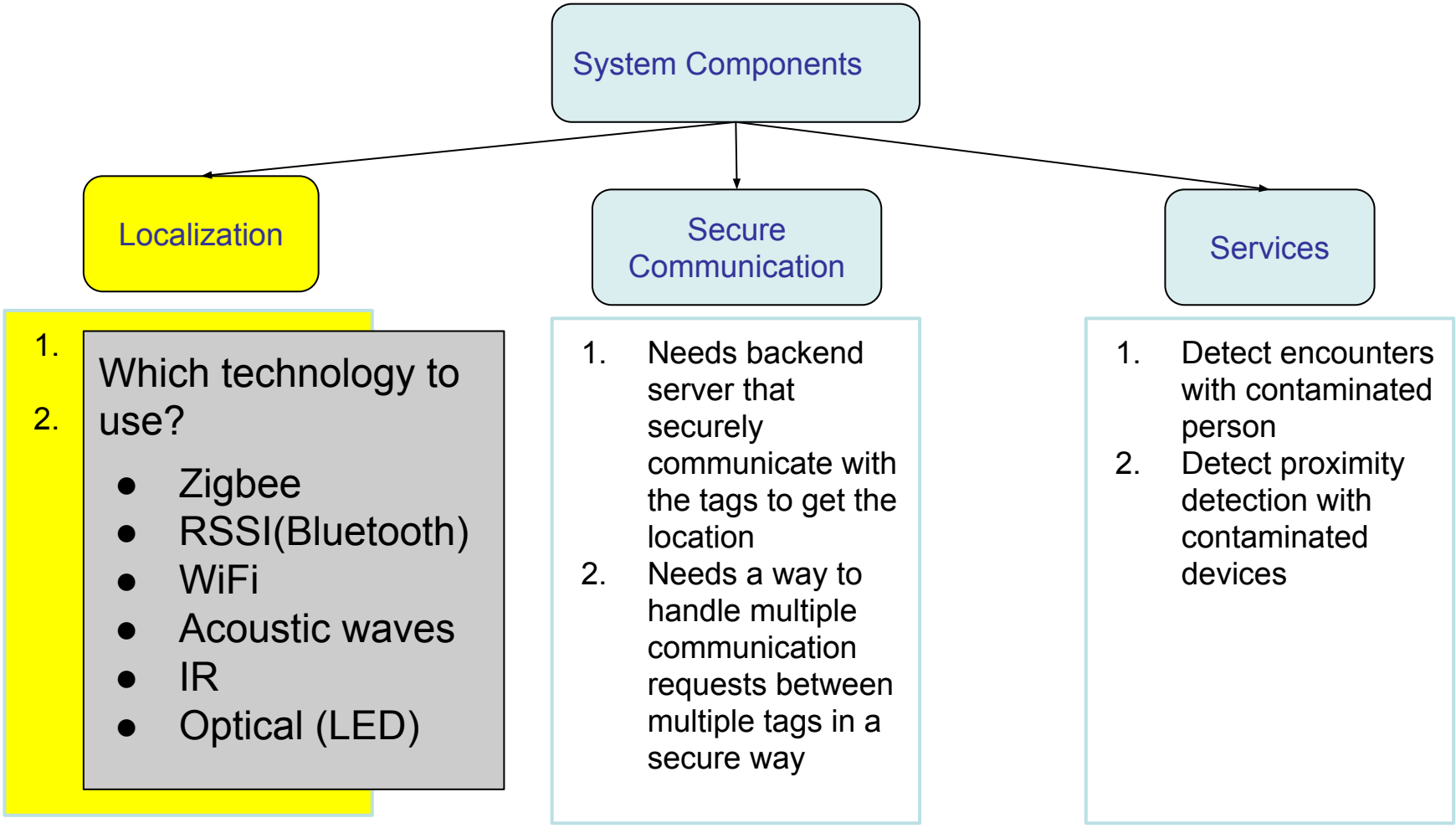
Encounters



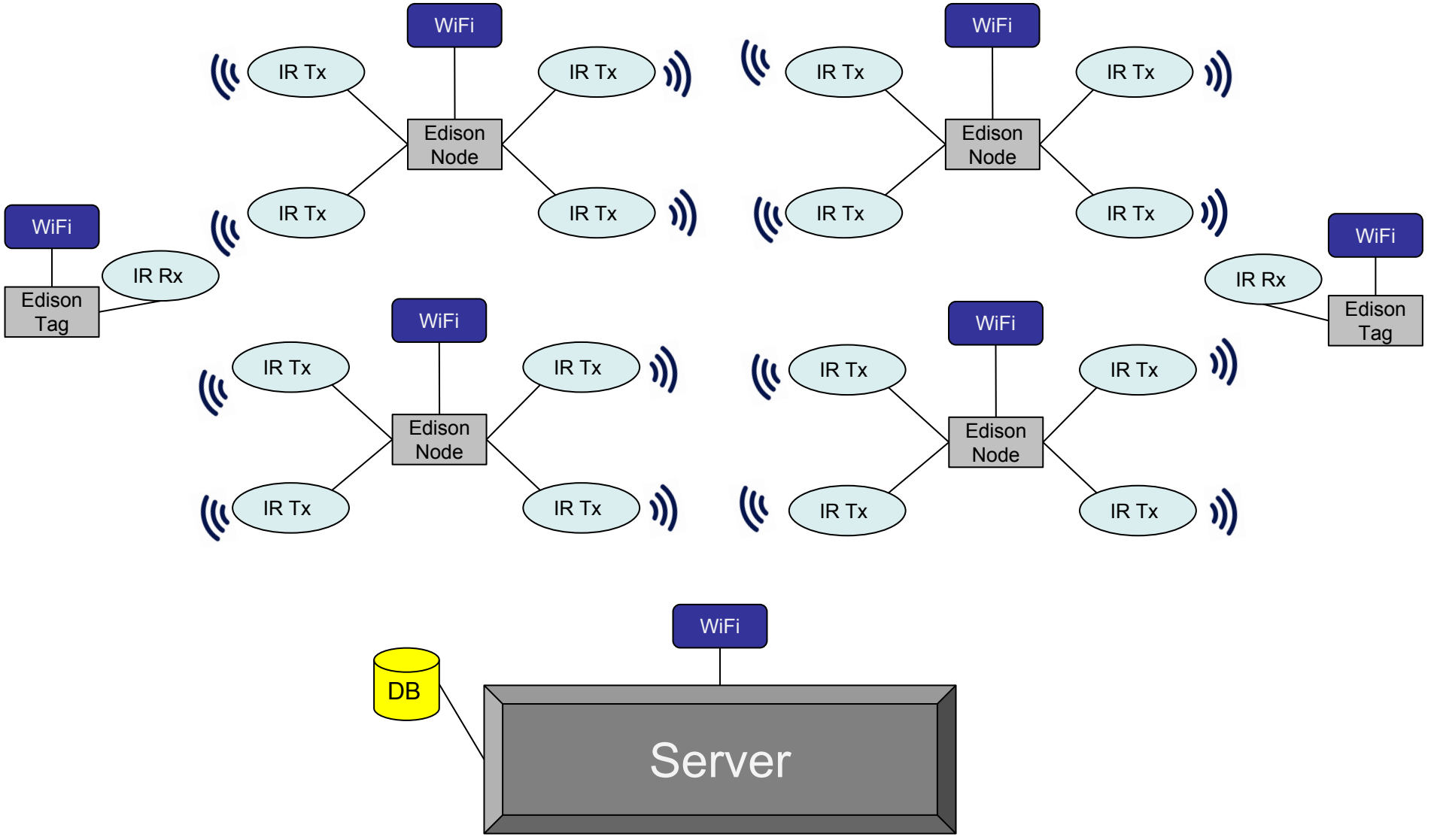
IoT approach for SHCP



Initial Challenges

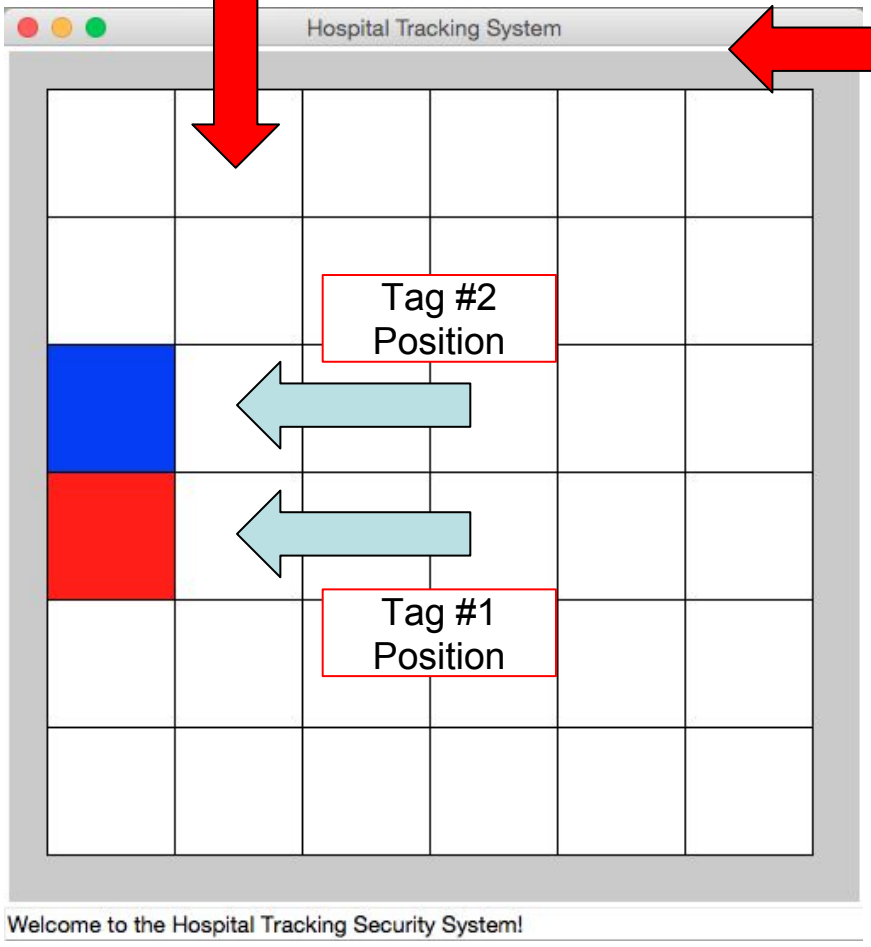


System Design Plan



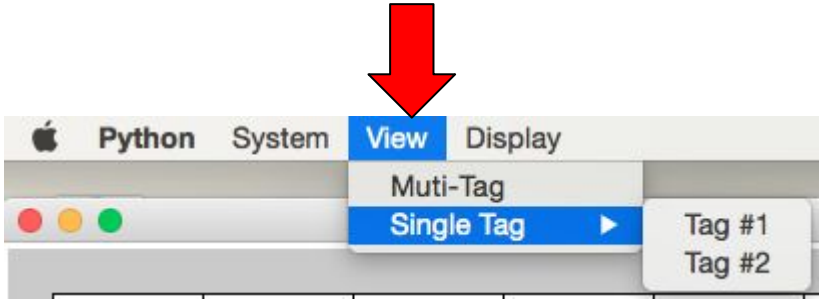
System Design GUI

Represents a subsection of an area



Main window that shows a tag's position

Menus that will give access to features



IR Emitter Circuit

-Vdd = 5V

-R_ir = 42 Ohms

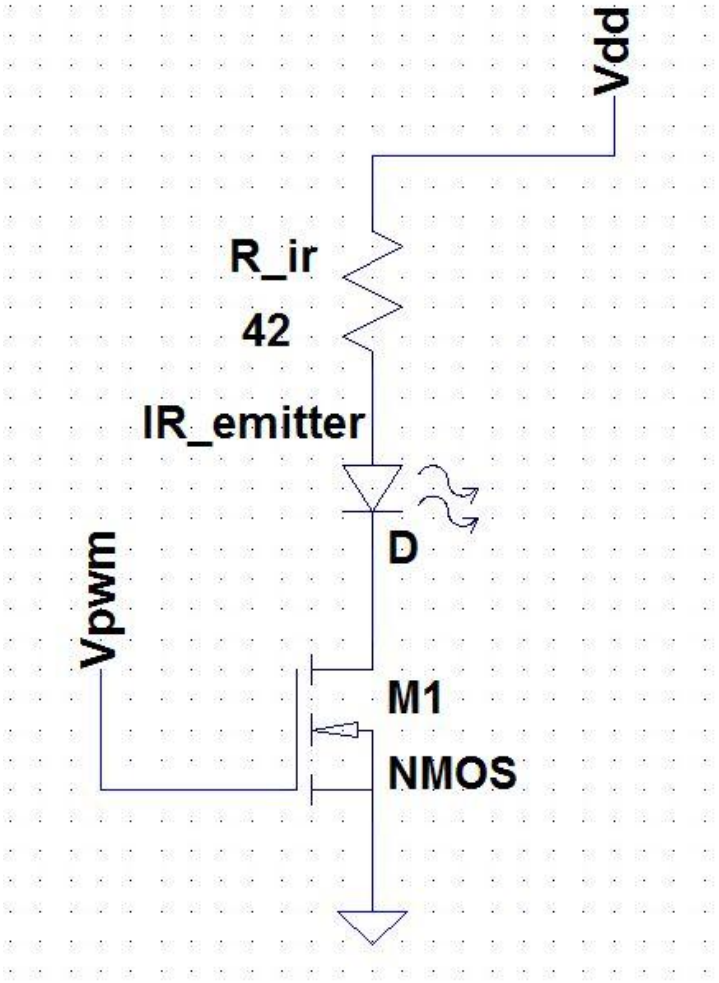
-Vpwm = Driving pwm from Edison

Observation:

- The range for the IR emitters is short not more than **2m**

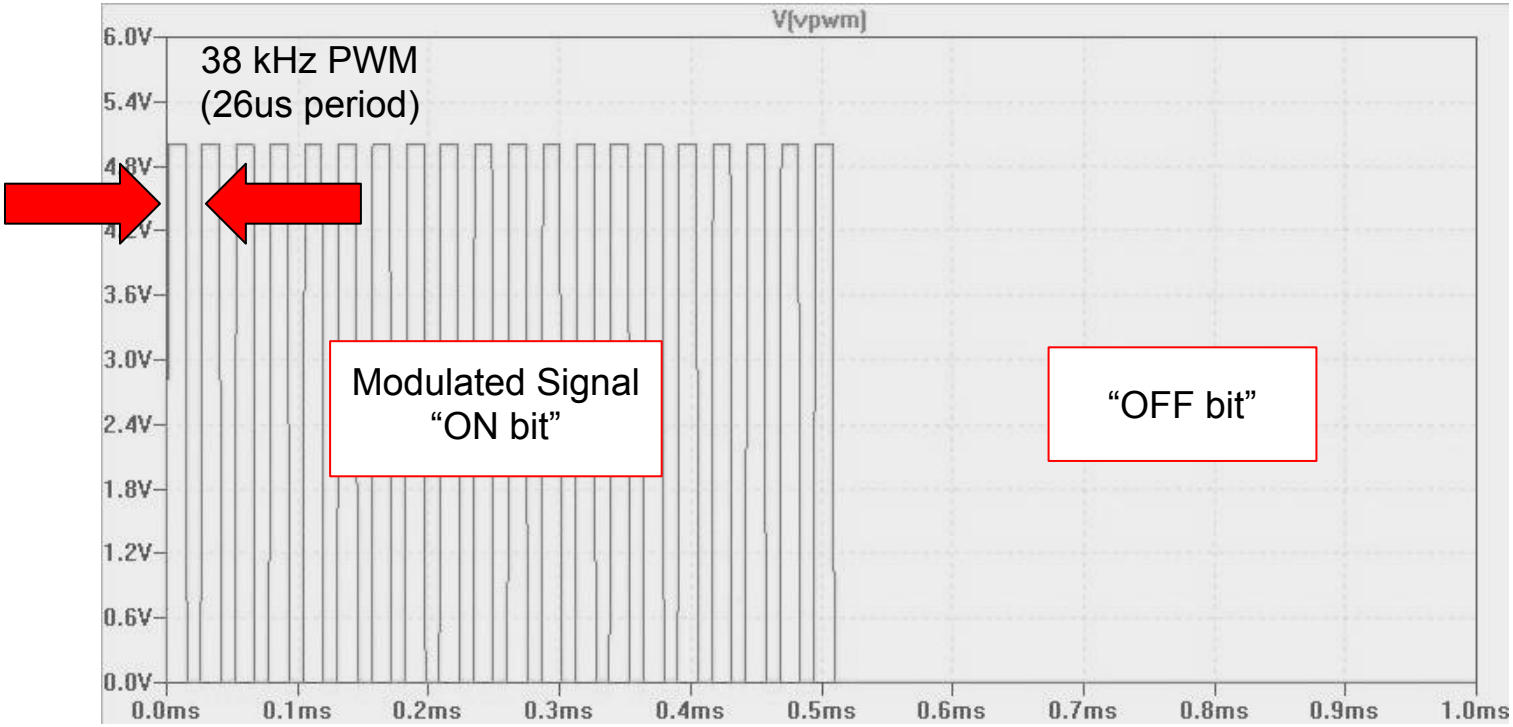
Need more range!

- We need a driver circuit to source more **current** to the **IR emitters** to increase their range



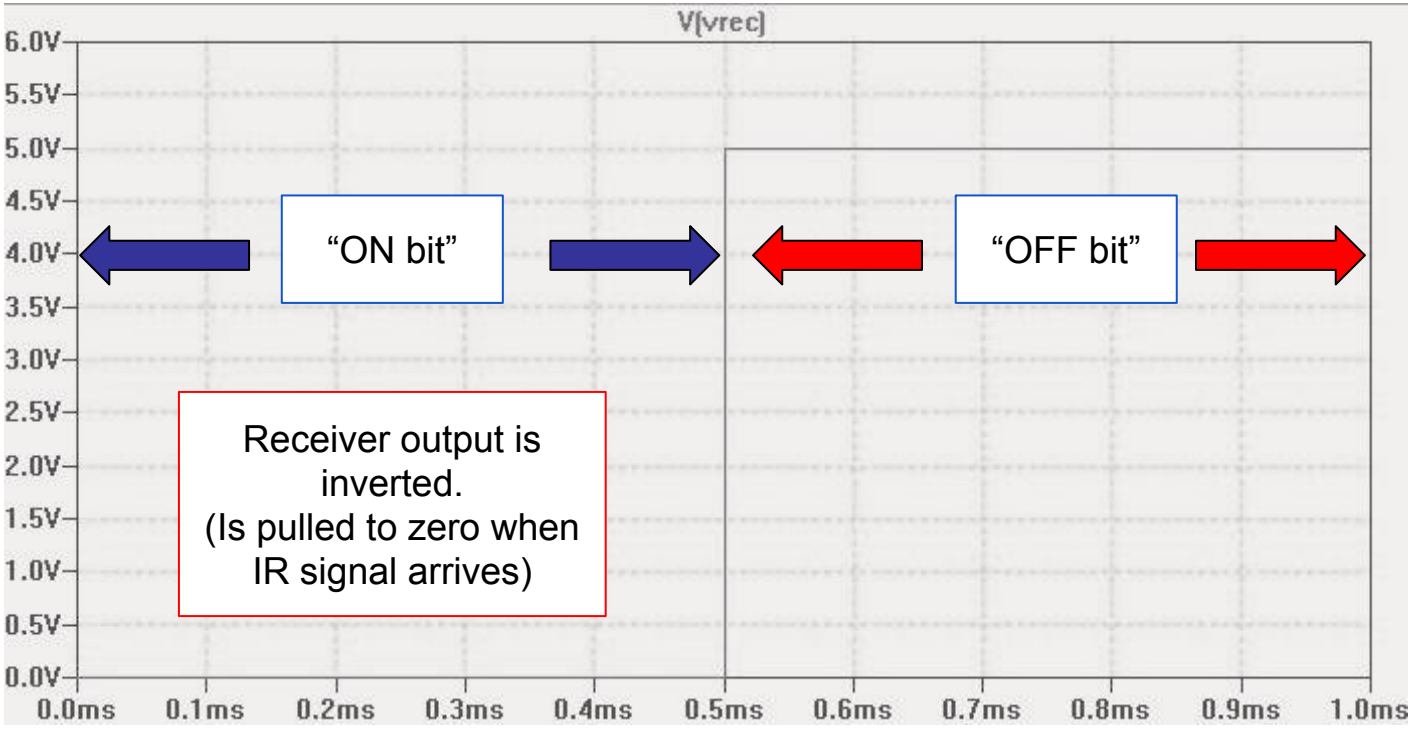
Visualizing the IR Transmission Process

- IR communication protocol is to modulate the IR signal transmission, and is typically done at **38 kHz**
- **38 kHz** is not the only modulation frequency used but is most common



Visualizing IR Reception

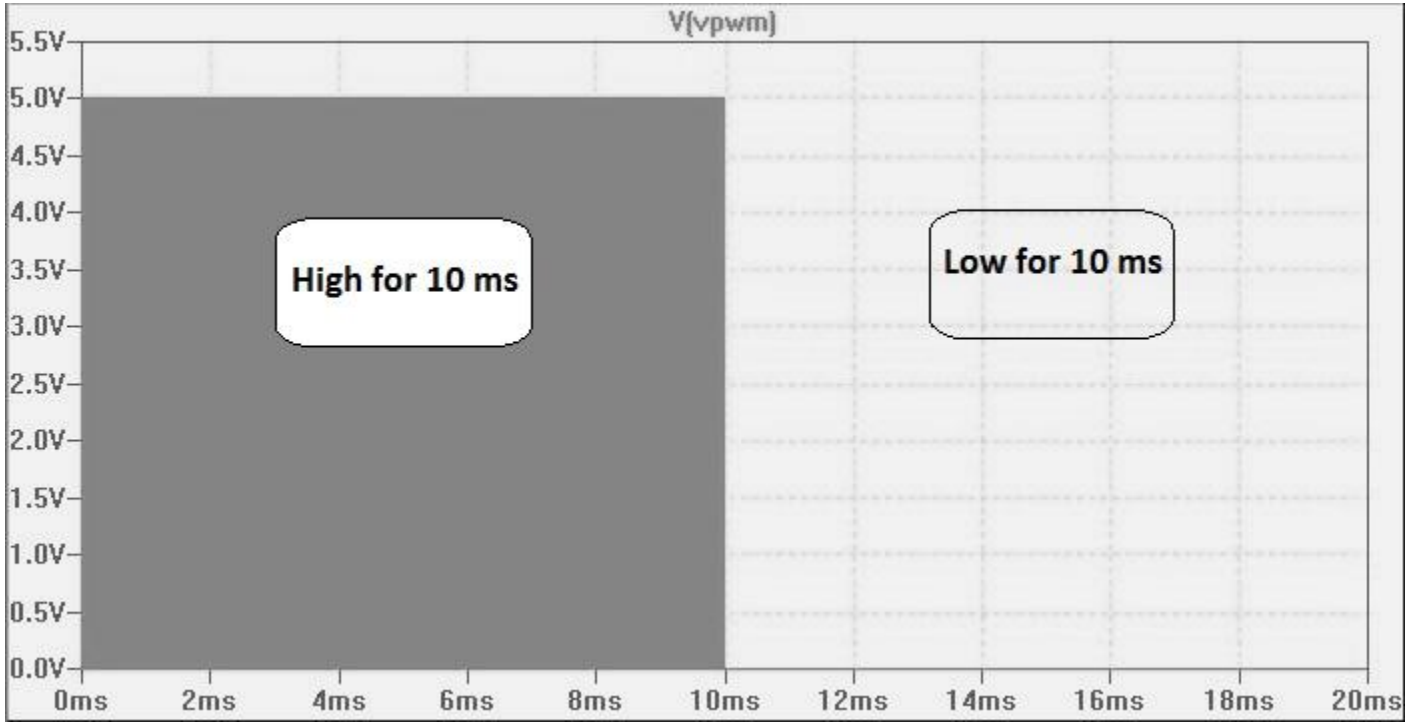
- Most IR receivers are made with a Bandpass Filter tuned to 38 kHz



Visualizing Preamble

Current Preamble:

6 PWM Cycles



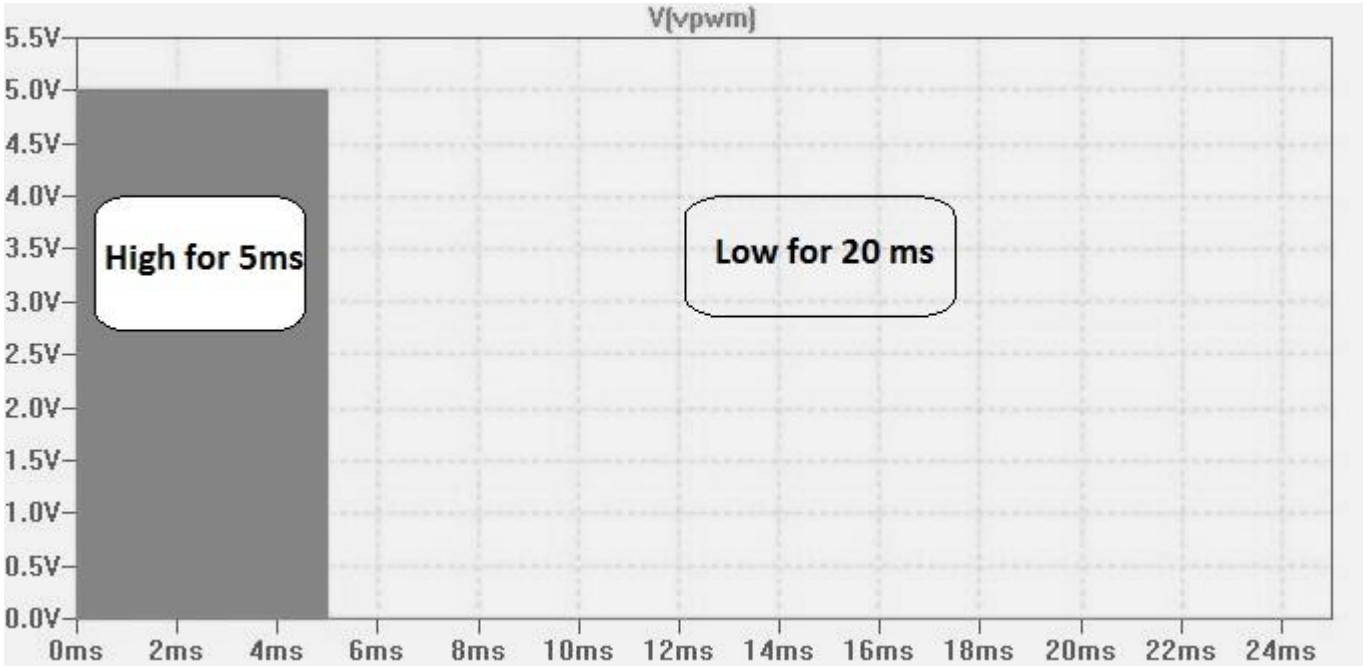
Transmitting Preamble

```
void send_preamble_sequence(int preamble_duration,
    int i = 0, j = 0;
    for(i = preamble_duration; i > 0; i--){
        //equal durations
        mraa_pwm_write(pwm, duty); // high
        for(j = PREAMBLE_DELAY; j > 0; j--);

        mraa_pwm_write(pwm, 0); // low
        for(j = PREAMBLE_DELAY; j > 0; j--);
    }
}
```

6 preambles

Visualizing 0 Bit

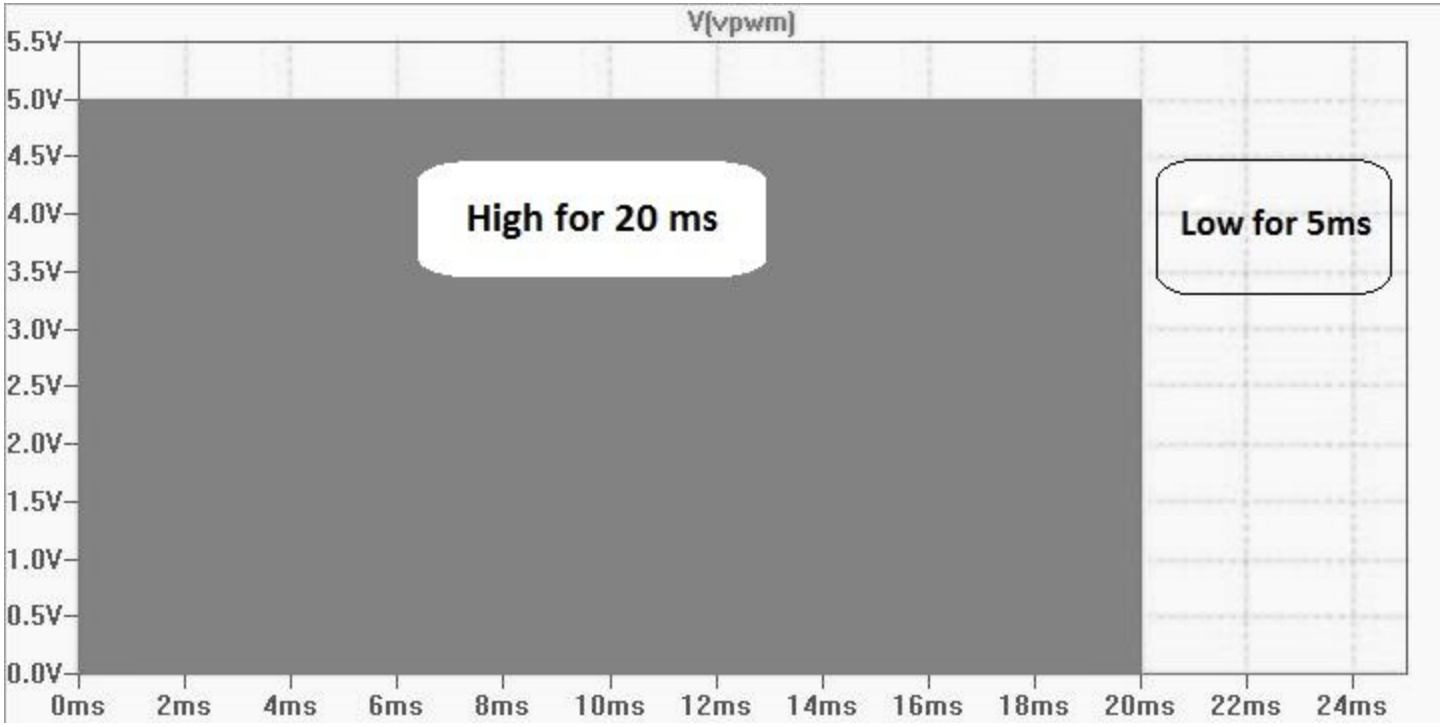


Transmitting 0 Bit

```
void send_low_bit(mraa_pwm_context pwm, float duty) {
    int i = 0;

    mraa_pwm_write(pwm, duty);   // 5ms
    for(i = SHORT_DELAY; i > 0; i--);
    mraa_pwm_write(pwm, 0);      // 20ms
    for(i = LONG_DELAY; i > 0; i--);
}
```

Visualizing 1 bit



Transmitting 1 Bit

```
void send_high_bit(mraa_pwm_context pwm, float duty) {
    int i = 0;

    mraa_pwm_write(pwm, duty);  //20 ms
    for(i = LONG_DELAY; i > 0; i--);
    mraa_pwm_write(pwm, 0);     // 5ms
    for(i = SHORT_DELAY; i > 0; i--);
}
```

Transmit Message

```
void send_message_bit(int edisonID, int irEmitterID, float dut

    // Preamble - Signals the Receiver Message Incoming
    send_preamble_sequence(PREAMBLE_DURATION, pin, duty);

    // Sending Edison Board ID # - 2 bits, MSB then LSB
    switch (edisonID){
```

argv/argc parameter

```
    // Sending Edison IR Emitter ID # - 2 bits, MSB then LSB
    switch (irEmitterID) {
}
```

1 ID/ each IR emitter

IR Transmit: Technical Challenge

Issue: Establishing the Proper Timing

Symptom: PWM pin was not being driven high and low as expected.

```
//equal durations
mraa_pwm_write(pwm, duty); // high
for(j = PREAMBLE_DELAY; j > 0; j--);

mraa_pwm_write(pwm, 0); // low
for(j = PREAMBLE_DELAY; j > 0; j--);
```

IR Transmit: Technical

Debugging: Translation from for-loop counts to nanoseconds

```
root@edison:~/ee202cProject# ./time
First time stamp is 1446667589.209627093
Second time stamp is 1446667589.220286244
Time difference in nanoseconds = 10659151
 root@edison:~/ee202cProject#
```

IR Transmit: Solution

Breakthrough: PWM timing

Solution: Lower the PWM switching speed

IR Receiver Driver Code

Algorithm -

1. **Sample** the signal output from the receiving LED

   Store the samples (0's or 1's) in an array **a[ ]**

2. Count the number of consecutive 1's and 0's in the array **a[ ]** and store the counts in **values[]** array

3. Detect the position where the **preamble** ends in the **values [ ]** (The preamble is known to be - 20 1's followed by 20 0's - this pattern repeated 6 times) .

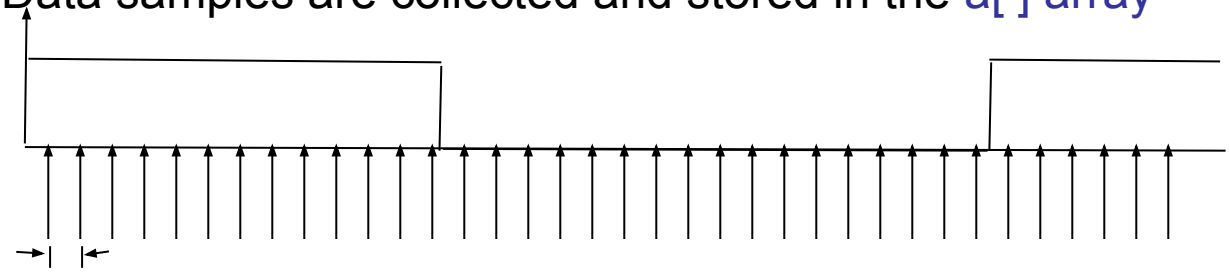
IR Receiver Algorithm

4. Read the 8 values from the **values[ ]** array, after the preamble and infer the transmitted bits from these values. A 10 followed by a 40 corresponds to a bit **0** and a 40 followed by a 10 corresponds to a bit **1**

5. From the 4 bits inferred , the first 2 correspond to the **Edison ID** and the remaining 2 correspond to the **LED ID**

Sampling Data

- Data is read from the gpio pin every 500 µs
- The sampling interval has been implemented using an idle for loop
- 3000 Data samples are collected and stored in the a[ ] array



Sampling Interval ~ 500 µs

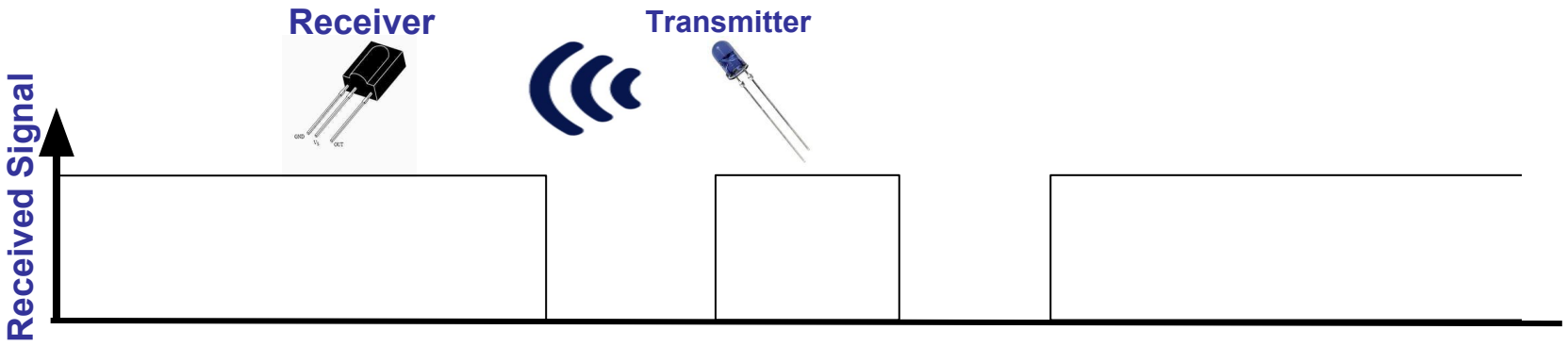
a[ ] = 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1

```
while(count < 3000 ) {
        irSig = mraa_gpio_read(gpio);
        a[count] = irSig ;
        for (j = 0 ; j < SAMPLING_RATE ; j++);
        count++;
    }//end while loop
```

← Read from the gpio pin

← idle For loop to implement the sampling interval

Consolidating a[ ] array and populating the values array



Populated values[ ] array

22 21 23 17 21 18 22 19 20 22 19 21 37 8 12 42 . … . .. .

Preamble **1** **0**

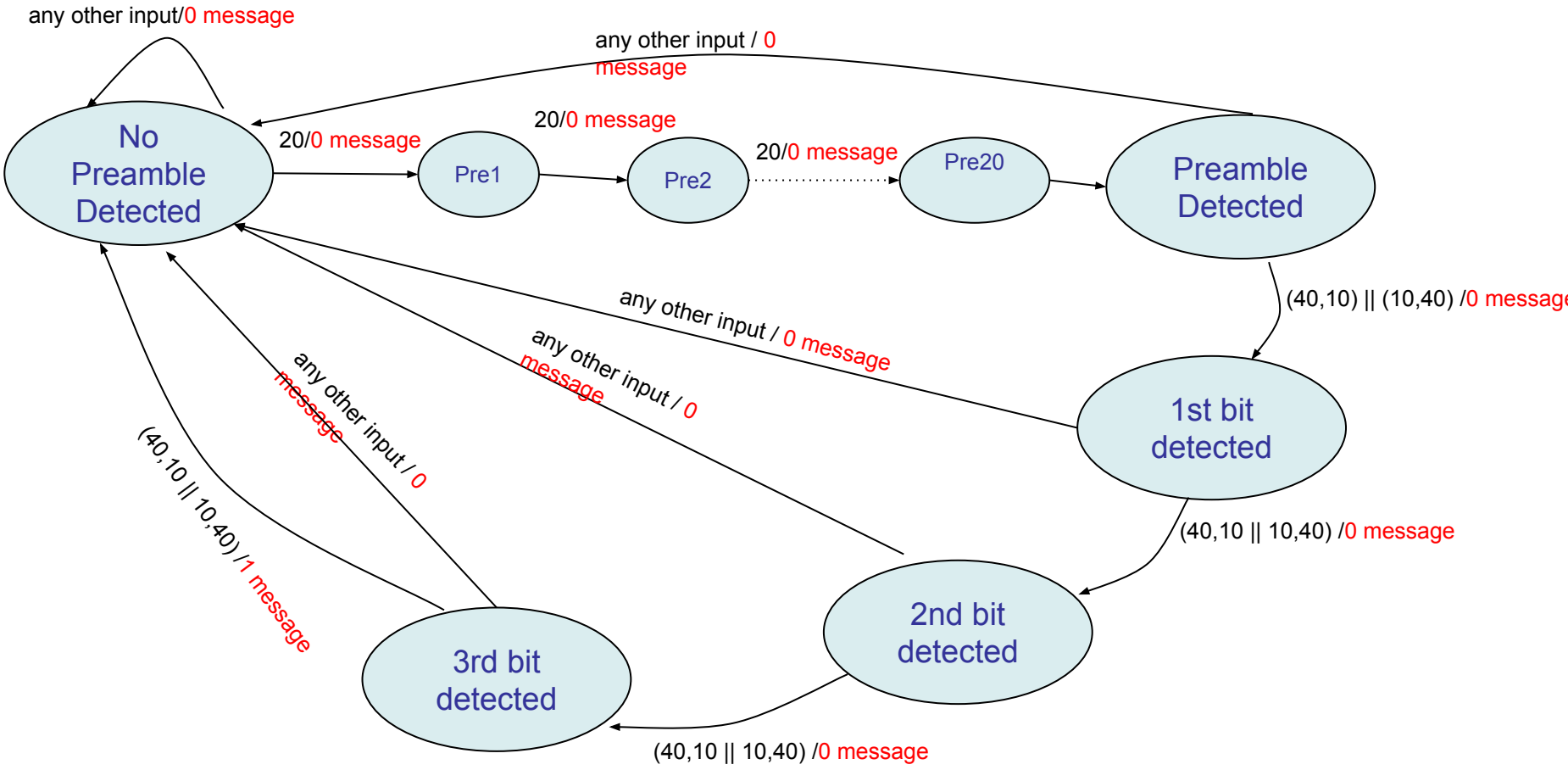
Preamble - 20,20,20,20,20,20,20,20,20,20,20,20

1 - 40,10

0 - 10,40

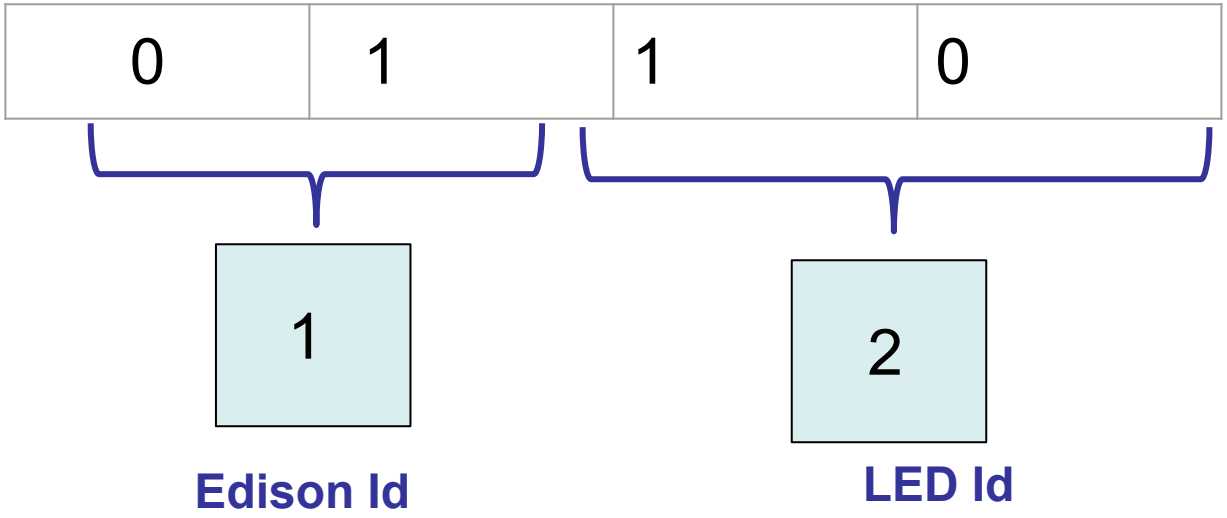
- But the value could be off by +/- 5 ! eg. a 20 could be between 15 to 25
- This variation is due to variation is pwm response time, variation in delay of idle for loop and various other factors

Mealy FSM detecting preamble and the value bits



Inferring the Edison ID and LED ID

One Received Message from an LED(4 bits)



Key Challenges in IR reception

1. **ISR didn't work !**
   - Initially we thought about setting an Interrupt Service Routine (ISR)
   - Edison ISRs are actually software simulated interrupt and hence take unpredictable amount of time
   - Hence we resorted to sampling the receiver pin at every 0.5 ms

2. **Variability in Idle For Loop delay**
   - Idle For loop also has variability in delay !
   - Developed a test program to figure out delay of idle for loop

3. **Figuring out the correct Message Duration !**
   - Had to increase duration from initially planned 5 ms to 220 ms
   - This helps in reducing the impact of the variations in delay of the idle for loop and also the variability in the response time of the PWM pin
   - Took us considerable time to figure this out and make the transmission work !

Key Challenges in IR Reception

4. Threshold for Detection

- Detection thresholds for bits 0 and 1 had to determined through experimentation

- eg. a 10 corresponds to anything in between 5 and 15

```
////////////////////////////////////////////////////////////
#include <stdio.h>
#include <mraa/gpio.h>

// this is the rate for the "for loop" which waits
#define SAMPLING_RATE 32000

#define UPPERBOUND_PREAMBLE 24
#define LOWERBOUND_PREAMBLE 16

#define UPPERBOUND_LOWVALUE 15
#define LOWERBOUND_LOWVALUE 5

#define UPPERBOUND_HIGHVALUE 45
#define LOWERBOUND_HIGHVALUE 35
```

What's next!

- Integration:
  - Localization - Server - GUI
- Services:
  - Tracking everyone
  - Contamination detection - Encounters
  - Unauthorized access
  - Patient zone
- More Security:
  - Encrypt the data on the tags

UCLA

Thank you!

Q & A

Test Plan

Action Points	Test Activity	Status
1.	Smart tag identifies ID of one IR IED	DONE
2.	Smart tag identifies ID two IR LEDS connected to two different Edison Anchor nodes	DONE
3.	One Edison anchor node drives 4 IR LEDs in a time-multiplexed fashion and smart tag detects each LED's ID	DONE
4.	4 Edison Anchor nodes drive 4 IR LEDs each . The IDs of all the 16 LEDs should be detected correctly by both the smart tags	TO DO
5.	Send smart tag location to the server (Client-Server testing)	DONE
6.	GUI Testing - gui should correctly display location of each of the smart tags	TO DO
7.	Handle multiple requests to Server from different Tags	DONE
8.	Examining the case when server goes down	DONE
9.	Integration	TO DO
10	Testing of localisation based services	TO DO

Team and Responsibilities
Salma

- Responsibilities:
  - Developed the secure communication algorithm of the project:
    - Examined/Developed several communication solution (HTTPS/SSH/SFTP) to choose the most convenient.
    - Developed the server/client code to handle the communication
    - Maintained the location data received in a database on the server for future queries.
- Future Responsibilities:
  - Maintain more security:
    - Encrypt the data stored in the file by a public key of a server to ensure the track information is kept secure if the server went down.
  - More Services!

Team and Responsibilities Raymond

- Security/Server:
  - Designed/Developed the GUI in python that will display the locations of the tags in real time, and will serve as the interface to all eventual implemented system features.
- Localization:
  - Designed the IR emitter driver circuit.
  - Developed the initial C code for IR emitter/receiver communication.
  - Contributed in developing the final IR communication code.

Team and Responsibilities
Pranjal

- Technology Selection for Localisation:
  - Evaluated various localisation technologies like Wifi, visible light Leds , Zigbee and Bluetooth and ultrasound transmitters and receivers
  - Recommended IR data transmission using 38Khhz modulation as the key technology to be used for the project
  - Figured out correct IR components and placed orders for the same

- Infrared LED driver code:
  - Designed/developed algorithm for receiving the Infrared signal. This includes first sampling the signal generated by the receiver and then processing the samples to infer the id of the transmitting anchor led from the sampled values
  - Developed the code for achieving the same . Collaborated with the team members for testing the same

Team and Responsibilities Anthony

Security/Server:

- Configured sMAP archiver to act as database for location data and time. Configured sMAP client to transmit data to archiver.

Localization:

- Contributed to IR Transmit/Receive code, resolving PWM timing issues
- Scaled IR Transmission/Receive code from 1 Edison, 1 LED to multi-Edison, multi-LED

Task Management

- Emails - Group Chat (Skype, Google Hangouts)
- Minimum 2 meetings each week (after class)
- Shared folder on UCLA Google Drive
  - Code
  - Presentations
  - Reports
  - Readings/Tutorials

Resources and References

- MRAA Library
  http://www.i-programer.info/programming/hardware/8744-exploring-edison-mraa-gpio.html?start=2
- IR Transmit Receive Tutorials (Sparkfun)
- SFTP Tutorials
- Python GUI Tutorials
- PWM Debugging
  - https://communities.intel.com/message/265937