Facilit-ease: An Optimized System for MIT Facilities

Authors:
Libby Aiello (libbya@mit.edu)
Kunal Tangri (ktangri@mit.edu)
Bobby Rauch (brauch@mit.edu)

Recitation Instructor:
Sam Madden
12:00 pm – Libby and Kunal
1:00 pm - Bobby

May 7, 2018

Contents

1 Introduction

MIT Facilities currently has thousands of electronic devices around campus, like thermometers, motion detectors, and video cameras. These devices successfully perform their designated tasks, but lack "smart" functionality. MIT Facilities is seeking a system installation which would enable data from these devices to be sent to a main server for analysis. This collected data would be utilized to speed up MIT Facilities' response to failures, save MIT funds, and improve our campus environment. The system proposed in this document utilizes MIT Facilities' current network of devices around main campus, and adds a system of organized, network notes to make the devices "smart." Our design prioritizes system performance in emergency situations, simplicity for the system administrator, low cost, and both modularity, and fault tolerance in all scenarios. This proposal outlines our system design by describing the placement and communication protocols of networked nodes, and programs run on the main server. These provide the delivery and analysis of smart data, and allow MIT Facilities to improve our lives on main campus.

2 Overview

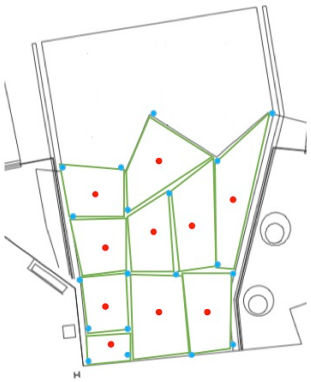
2.1 Network Topology

Our system consists of a network of nodes, which together are capable of collecting, analyzing, and responding to data gathered from many smart devices. Some potential use cases include intelligently monitoring temperature, detecting device failures, and providing a video security system. To support these uses, our network topology is designed to adapt to each of them in a reliable, and fault tolerant manner, while keeping cost and deployment time low.

2.1.1 Layout

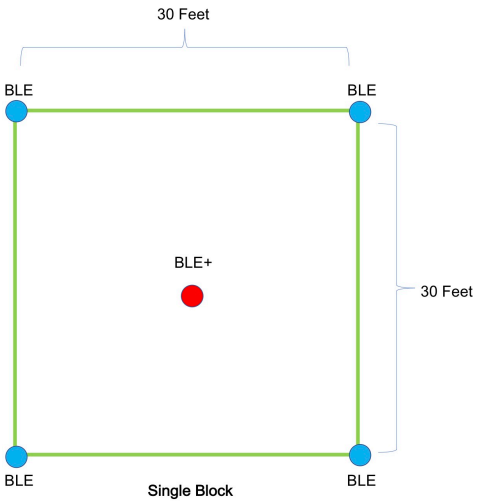
Blocks:

The system can achieve its desired functionality through its reliance on a simple network module called a block. Each block consists of four BLE (Bluetooth Low Energy) repeaters placed at the four corners of a 30ft by 30ft square[1], as well as a BLE+ repeater placed in its center [See Figure 2]. The BLE repeaters are the primary communicators with the smart devices, receiving data from all devices within their block area and transmitting it towards the nearest gateway. BLE+ repeaters serve as a fall back mechanism in case the four BLE repeaters cannot handle the bandwidth flowing through them. Since the majority of network traffic flows through the BLE repeaters, BLE+ repeaters generally have low traffic and low latency. Therefore, they are solely responsible for the transmission of crisis mode video to the nearest gateway. Our network takes advantage of these blocks by connecting them in a grid to collect data from smart devices and push it towards the nearest gateways. Thus, our system requires that all smart devices are contained within a grid block. Additionally, these grids are an abstraction, and, in practice, they are fitted to the space they are required to cover [Figure 3].

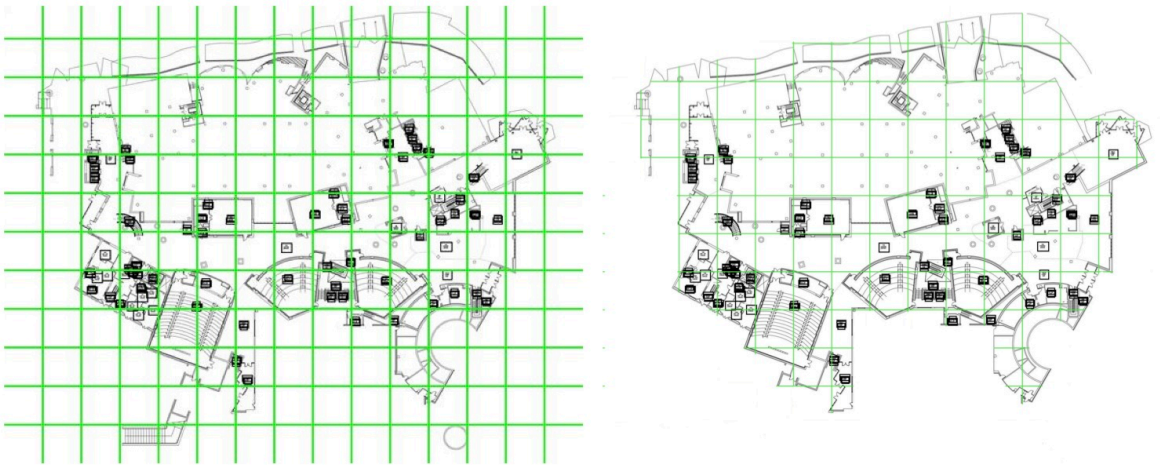


[Figure 1]

[1] The actual placement of these BLE repeaters is flexible. However, since BLEs have a communication range of 30 ft., each block must contain its four BLE repeaters within that 30 ft. by 30 ft. range to ensure that communication can still occur. For this paper, we will use the perfect square placement of BLEs for simplicity, but it is possible, in practice, to have flexible blocks [Figure 1].



[Figure 2]



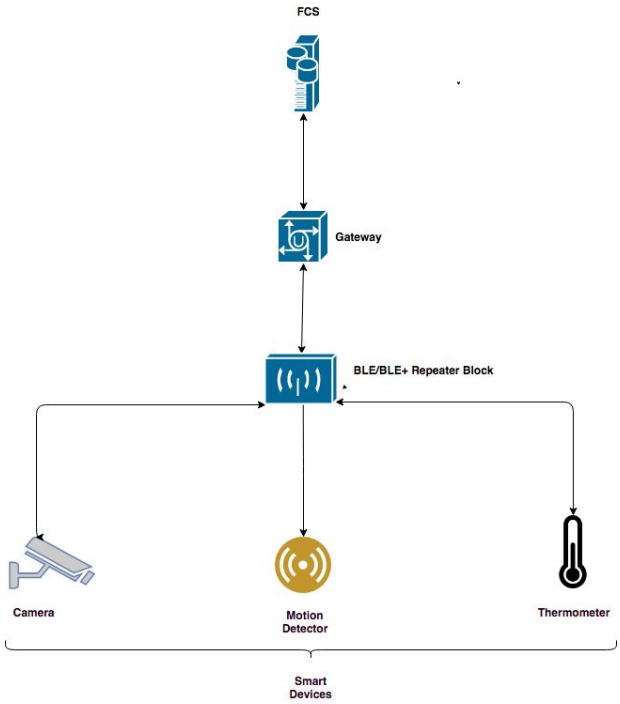
Grid Abstraction over Stata Center

*Grid blocks not true to size/scale and without flexible fitting

[Figure 3]

High-Level Data Flow:

Under this grid-based topology, the flow of the data is as follows. First, smart device data are relayed to one of the BLE repeaters in the block that contains the devices[2]. Next, the repeaters forward the data they've received towards the nearest gateway - this consists of passing data through the repeaters of other blocks that are one step closer to a gateway. Finally, from the gateways, data are transmitted to the Facilities Central Server (FCS) via the internet. [Figure 4]



MIT Facilities System Diagram

[Figure 4]

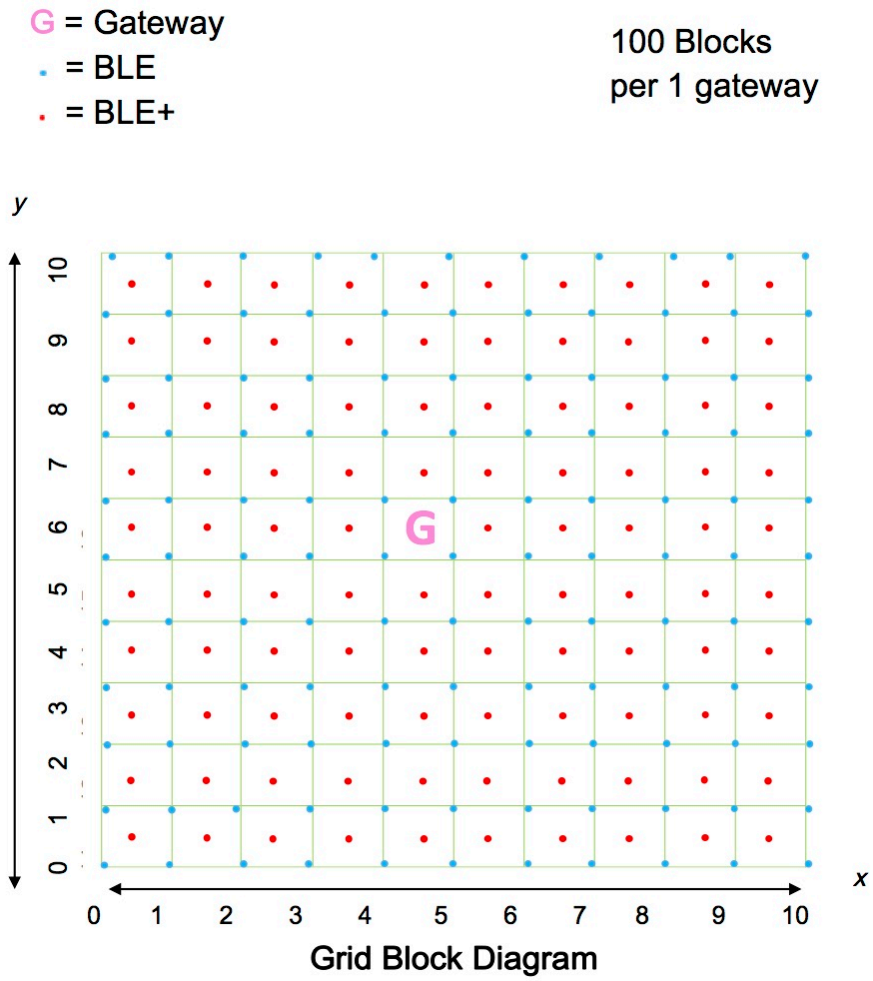
[2] or a BLE+ repeater if the BLE repeaters are operating at full capacity.

Gateways:

All data pass through gateways, which transmit smart device data to the FCS. The number of gateways in our system is determined to minimize congestion, and the system administrator can decide to increase the number of gateways depending on whether they are aiming to minimize cost or maximize reliability. Adding more gateways makes the system more expensive, but results in a shallower network - data have to travel through fewer steps to reach the FCS, thus incurring less latency. Ideally, the system administrator should place a gateway in the middle of a 10x10 grid of blocks [Figure 5]. In areas of main campus in which a perfect 10x10 grid cannot be fitted, the system administrator should attempt to place gateways so that they have about the same number of blocks on each side, with no more than five blocks on each side of every gateway. Additionally, the gateways should be placed such that they serve roughly equal numbers of blocks. This placement of gateways is beneficial for two main reasons:

1. Maximizes the number of paths that data can travel to the gateways, so that there is a more even distribution of data flow across paths. This prevents any single path from becoming overwhelmed with traffic.
2. Data are never more than 10 hops away from the nearest gateway. As a result, data are sent to the FCS within one second of transmission from smart devices[3].

[3] In the worst case, all smart devices send out their data in a given second. Therefore, data should be able to pass from smart device to the FCS in less than a second (10 hops takes one second) to prevent congestion from occurring.


G = Gateway
= BLE
= BLE+
100 Blocks
per 1 gateway
y
x
G
0 1 2 3 4 5 6 7 8 9 10
Grid Block Diagram

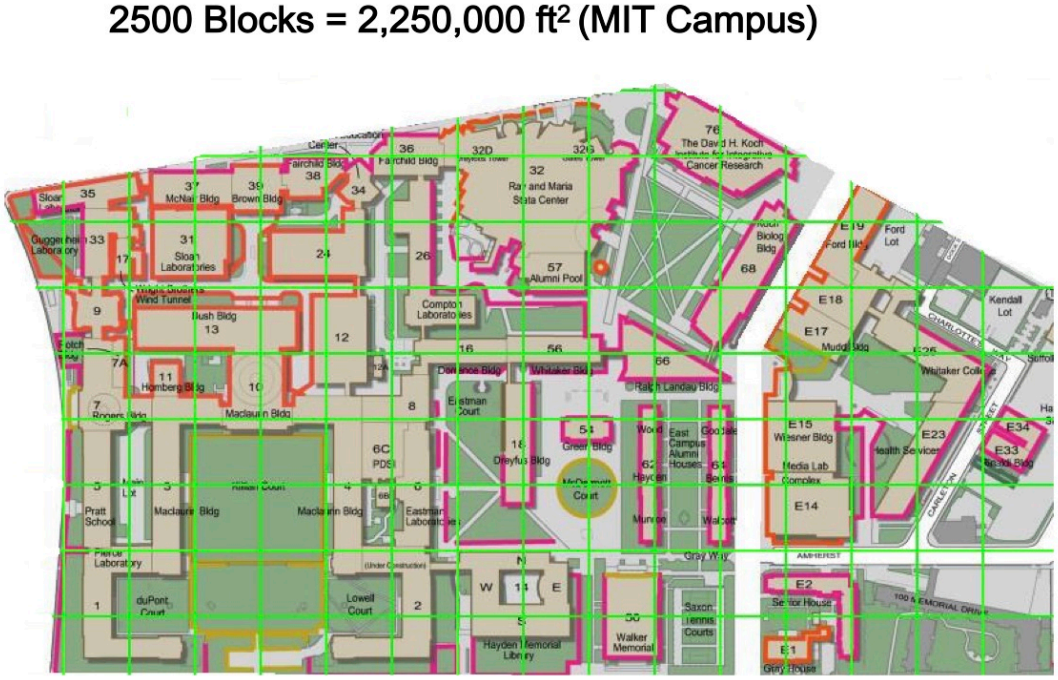
[Figure 5]

2.1.2 Features

Our grid-based network is scalable and allows for simple calculations of the number of repeaters and gateways that are needed to support an arbitrary quantity of smart devices. There are two constraints faced when making these calculations: the maximum amount of data that can be transmitted from all the smart devices in one second, and the amount of area that needs to be covered. We'll present a tailoring of our system to the specifications of MIT.

MIT:

MIT has 15,000 motion sensors, 15,000 thermostats, and 1,000 cameras. Additionally, we estimate that our network needs to cover about 2,250,000 square feet [Figure 6]. Given these specifications, we calculated that the maximum amount of data that can cumulatively be transmitted per second from all the smart devices is 28,120,000 bytes [Appendix Eq. 1], and that 2,500 blocks are needed to cover the total area of campus [Appendix Eq. 2]. Our system requires 25 gateways to support 2,500 blocks with the ideal structure, but may require more depending on the system layout. Using these hardware specifications, we can calculate that network congestion is not an issue [Appendix Eq. 3]. With the number of blocks and gateways in our ideal structure, our total cost will be $163,510 for MIT's campus [Appendix Eq. 4].



- *Blocks and # of gateways not true size/scale as pictured*

[Figure 6]

3 Design

3.1 Communications Protocol

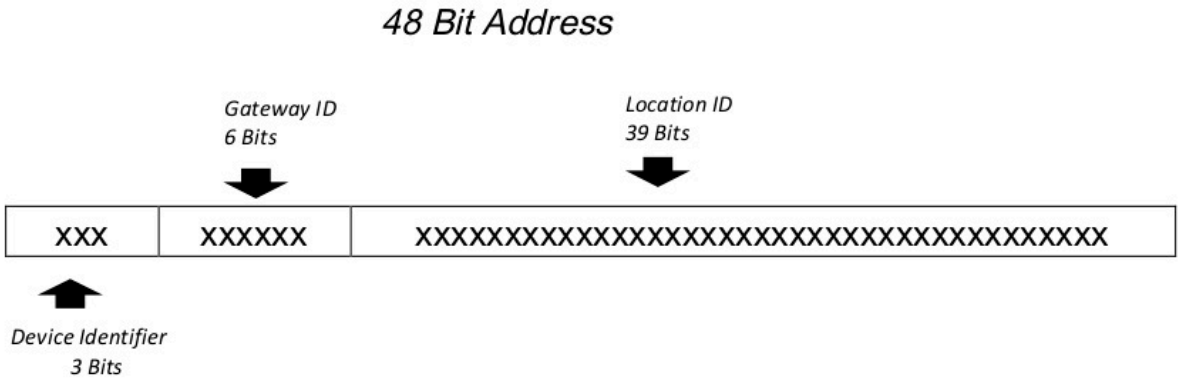
The layout of our network, combined with an intelligent communication protocol, ensures that data packets are correctly relayed to the appropriate devices. Networked systems can have high deployment costs and become quite unreliable when device naming is up to the system administrator. As a result, we have devised a self-naming scheme which contributes to the reliability and fault-tolerance of our network layout.

3.1.1 Naming

Our naming convention for gateways, repeaters, and smart devices serves as the basis for our communications protocol. Important routing information is encoded in each 48-bit device identifier as follows: a three-bit device type identifier [Figure 7], a six-bit gateway identifier, and a 39-bit location identifier. The three-bit device type component allows the system to distinguish between different types of devices for routing purposes, and the six-bit gateway identifier represents the ID of the closest gateway. Our system uses the 39-bit location identifier to route data between smart devices and the FCS. At the system's initialization, only the three-bit device identifier needs to be assigned. Both the gateway identifier and location identifier are assigned once the system completes its self-naming process. The resulting 48-bit self-named identifier prevents misnaming of devices and allows for easy setup for the system administrator. [Figure 8]

Device Type	Number in Bits
Gateway	000
BLE+	001
BLE	010
Camera	011
Motion Detector	100
Thermostat	101

[Figure 7]



[Figure 8]

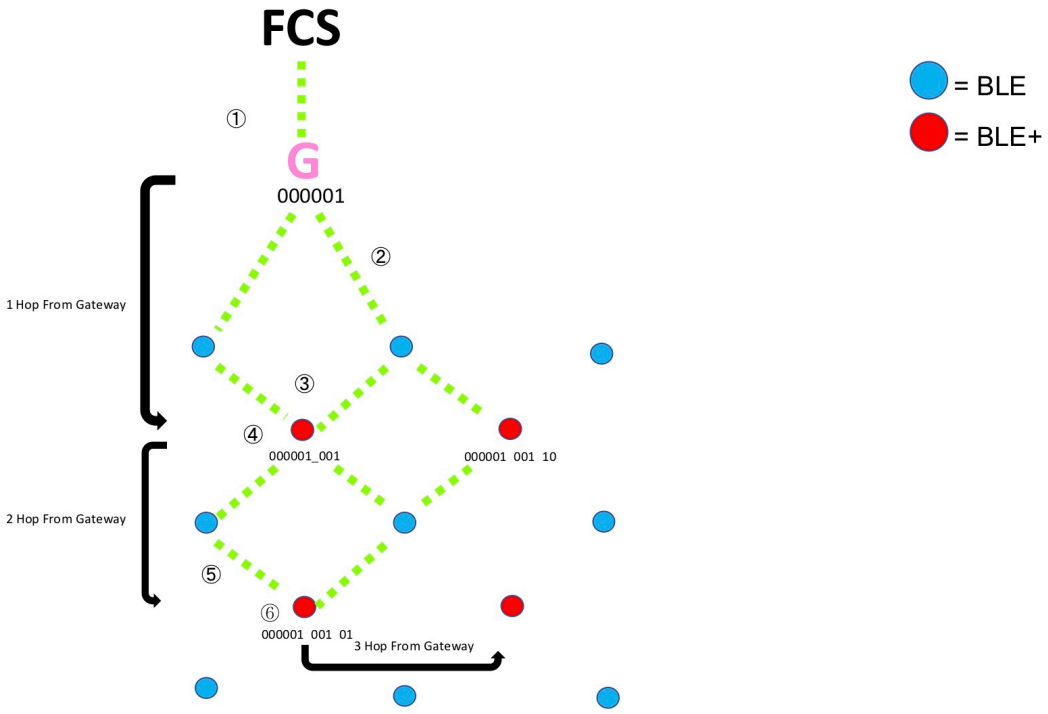
3.1.2 Self-Naming Process

Our self-naming process is run after devices are installed, and assigns location IDs to repeaters based on how many hops it takes to get to the nearest gateway. There is a 39-bit space allocated in the device ID for the location ID. However, all 39 bits are not necessarily utilized - the length of the location ID depends on the number of hops a device is from a gateway. As a result, most location IDs are padded with additional zeros to fill up the remainder of the 39-bits. For example, if a device is four hops from a gateway, its location ID will be 9 bits and the rest of the allocated space will be filled with 30 zeros. Our system uses this location ID property to route data to and from the FCS. The network initialization and location ID assignment process follows these steps [Figure 9] and [Figure 10]:

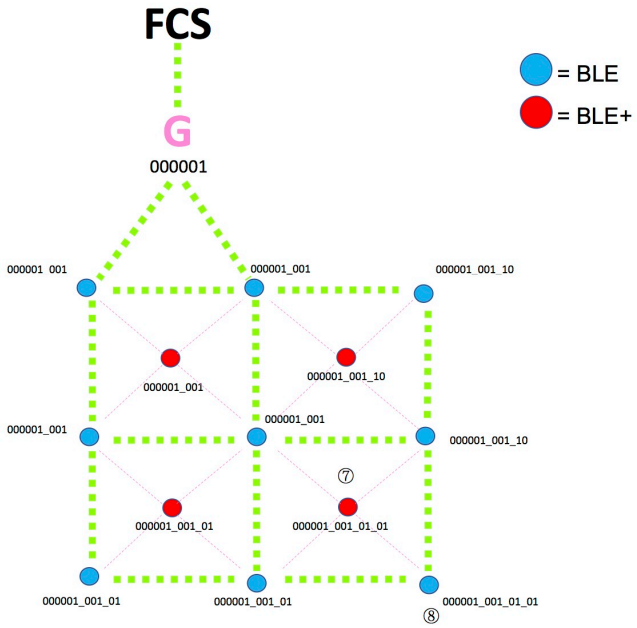
1. Gateways connect to the FCS and are each assigned a unique six-bit gateway ID.
2. Gateways broadcast their IDs, and connect to any BLEs they receive a broadcast back from.

3. Gateways connect to the BLE+s in their directly adjacent blocks (using BLEs as intermediaries to enforce adjacent connections). The gateway can connect up to four BLE+s, and, as a result, three bits are needed to assign four BLE+s each a non-zero[4] unique identifier.
4. Gateways assign their six-bit gateway ID and a 39-bit location ID to the BLE+s they've connected with (discovered).
5. BLE+ repeaters that have been discovered repeat this process, connecting to their adjacent BLE+s (using BLEs as intermediaries to enforce adjacent connections). Similar to step 2, BLE+s connect to these intermediaries by broadcasting their IDs, and connecting to BLE repeaters they receive broadcasts back from.
6. Upon connecting to an undiscovered BLE+, the previously discovered BLE+ assigns the new BLE+ the same gateway ID and location ID that it holds. Then, the old BLE+ assigns two new bits to the new BLE+s location ID to indicate that it is one additional hop away from the gateway.
7. Repeat from step four until all BLE+s have been named, and do not rename any BLE+s that have already been discovered (BLE+s that have been discovered are distinguished because they have already formed connections)
8. BLE repeaters assume the gateway and location IDs of the BLE+s closest to their primary gateway and disconnect from all BLE+s besides their primary (i.e. the BLE+s with the smallest number of hops in its location id)

[4] These must be non-zero because if a BLE+ could have an all-zero location identifier, it would be impossible to tell the difference between the location ID and padded zeros.



[Figure 9]



[Figure 10]

3.1.3 Device Communication

After running the initial self-naming process, the following connections are in place:

- Gateways are connected to directly adjacent BLE+s and BLEs (up to eight connections)
- BLE+s are connected to directly adjacent BLE+s and BLEs (up to eight connections)
- BLEs are connected to directly adjacent BLEs and their primary BLE+ (up to five connections)

The final step in the system's initialization is to connect all of the smart devices to the network of repeaters and gateways. Since the smart devices have random IDs, apart from their device type identifiers, our system needs a mechanism to determine their location in the system. This location is necessary for a few system functions, including our reliability protocol and sending device updates. To achieve this, each smart device chooses a default BLE repeater to communicate with, and relays this new connection info to the FCS and its primary gateway. By storing a mapping between the smart device ID and BLE ID, the FCS and gateways can route packets to individual smart devices because BLE IDs contain location information. In short, the smart device initialization step works as follows:

1. Each smart device broadcasts its identifier and forms a connection with the first BLE repeater to respond to it - this is the device's default repeater
2. Each smart device then sends an initialization packet, containing its ID, to its default repeater
3. Each BLE repeater, if receiving a message from a smart device, adds its own ID to the initialization packet, and forwards it towards the FCS. Additionally, we use the acking protocol defined in section 3.4 to ensure that these initialization packets aren't lost. If the BLE repeater receives a message from another

BLE (i.e. initialization packets from other blocks) it continues to forward it to the FCS

4. When an initialization packet reaches the FCS, the contained smart device and BLE IDs are stored in a table, mapping from smart device ID to BLE ID

3.1.4 Routing

Our naming convention allows for easy routing between the FCS and smart devices. As a result of our naming convention, device location IDs grow 2 bits longer with each hop away. When routing from the FCS to a smart device, first the FCS determines the default BLE for the smart device. Then, the FCS sends a data packet marked with the BLE and smart device ID. That data packet then propagates through the grid, trying to match each hop of the target BLE's ID. For example, when a data packet is being routed to a location ID of 001_10_11_10[5], it will initially look for a repeater with a location ID that begins with 001. Once it finds an initial match, it attempts to match the bits from the second hop. So, from a repeater that begins with 001, this packet will be routed to a repeater that has 10 as the next hop. Then, from a repeater that begins with 001_10 the packet will route to a repeater that contains 11 as the next hop, and so on. The packet will know it has arrived at the target smart device's default BLE, when it fully matches its location ID. Then, the packet will be sent from this BLE repeater to the smart device ID that the packet contains the identifier for. When routing packets from smart devices to the FCS, the hops can be followed the other way, by finding devices with shorter location IDs. So, if we were routing in the opposite direction in the example above, the packet would travel from a location ID of 001_10_11_10 → 001_10_11 → 001_10 → 001 → Gateway.

[5] Underscores are used to highlight the bits added at different hops.

If at any point, all directions contain the same length ID, the path randomly chooses one repeater to continue to, and then goes on from there. Given the setup of the system, it is guaranteed that one of the BLE repeaters in each block is also adjacent to a BLE repeater that is either closer or further away, so this random selection always progresses the packet closer to the destination.

Initially, all data attempt to flow through the network of BLE repeaters to leave BLE+ repeaters available for other purposes, and because BLE repeaters do not fail. However, if a smart device attempts to send a packet to a BLE repeater and it is dropped due to bandwidth issues, the smart device automatically disconnects, and sends its data through the BLE+s instead. Our fallback mechanism from BLE repeaters to BLE+ repeaters provides an extra layer of reliability to help prevent any data loss and congestion from occurring.

3.1.5 Messaging Protocol

In our system, messages are sent from video cameras once every second (one frame per second default), from thermostats every four minutes, and from motion sensors whenever motion is detected.

Messages that consist of video data frames are fragmented into pieces that satisfy the max packet size and have the following components:

1. Video Camera Device ID
2. Crisis mode bit (1 for crisis, 0 otherwise)
3. Direction of flow bit (1 towards FCS, 0 towards smart devices)
4. Frame ID
5. Fragment ID
6. Frame Data

Messages that consist of thermostat or motion sensor data include:

1. Device ID
2. Direction of flow bit (1 towards FCS, 0 towards smart devices)
3. Timestamp at time of data collection

4. 32-bit data (for thermostats)

The crisis mode bit is included so that our system can respond with urgency to packets sent from cameras in crisis mode. If the crisis mode bit is set to one, then the first BLE repeater to see this packet forwards it through BLE+ repeaters towards the nearest gateway - reducing the probability of bandwidth issues and latency. All packets also include a direction of flow bit. This is required for the system to determine if it is following the routing protocol towards the FCS, or the protocol to route back to smart devices. The device IDs, timestamps, and Frame/Fragment IDs for cameras, are included to increase the reliability of the system. We are aware that 0.0001% of packets are dropped over the Bluetooth connections in our network, so our system incorporates a confirmation mechanism to ensure that packets are received. This confirmation mechanism operates as follows:

1. BLE repeaters receive data from smart devices within their block and from other BLE repeaters routing their data towards the nearest gateway.
2. For data collected from smart devices within the BLE repeater's block[6], the BLE transmits the data towards a gateway and makes a backup copy on the nearest BLE+ repeater. If the device type of the device that sent the data is not a smart device, the BLE repeater continues forwarding the data towards the FCS.
3. The original data will most likely reach the gateway, and when it does, the gateway sends a confirmation message back to the BLE+ repeater of the block that the data originated in (easily located through the mapping of smart device ID to BLE ID, which, in turn, matches the ID of the nearest BLE+).

[6]A BLE can tell if data is coming from a smart device within its block by looking at the device type of the data's sender. If the device type is a smart device, then the data is coming from within the BLE's block, otherwise the data originated from a different block, and is just flowing through this repeater to get closer to the gateway.

4. This confirmation includes the device ID, and either the Frame ID (for video data) or timestamp (for motion sensors and thermostats). We only confirm camera frames as received once the FCS receives all of its fragments.
5. When BLE+ repeaters receive confirmations, they delete the stored backup copy from memory, as it has been received by the FCS.
6. If no confirmation message is received within a minute,[7] then the BLE+ resends that piece of data from its copy, and continues to keep that copy in storage until it is confirmed.

3.2 FCS Processing

Facilities uses a single centralized machine, called the FCS, to process the data that are sent from the sensors, repeaters, and gateways. Packets sent to the FCS arrive in a FIFO queue and wait to be processed.

3.2.1 Main Thread

The main thread is continuously run on the FCS, processing data in the FIFO queue and storing it accordingly.

· Video frames: The system administrator can choose to store all video frames or run the filtering function to find duplicates. If the frames should be analyzed, the Video Analysis Thread is run and if some of the frames are the same, then only one frame is passed to data storage via the Data Storage Thread.
· Motion detector data: Any motion data in a room is treated as a sign that the room is currently in use.

[7] System administrators can modify this if needed.

Thus, when motion detector data is received, the main thread spawns a new thread to query the motion data hash table and determine whether the room was assumed to be used or not, based off of the most recent motion timestamp in the hash table. These threads must acquire the lock for the given entry in the hash table, given that the hash table is protected with locks If the room was assumed to not be used, this thread tells the Temperature Updating Thread to increase the temperature to the desired temperature. After, it updates the most recent motion timestamp in the hash table to that in the received motion detector data.

· Temperature data: The Data Storage Thread is called to input this data into the hash table.

· Malfunction data: The User Interface Thread is called to display this information as an error to the Facilities staff members.

While the threads above run in response to received data, the following threads run constantly in the background:

· Room Activity Thread: Runs over the motion data hash table to determine whether any rooms have not seen motion for two hours. If so, it spawns the Temperature Updating Thread to decrease the temperature to ten degrees cooler.

· The Historical Thermostat Analysis Thread: Can also be used by the system administrator with their desired specifications to determine whether any of the thermostats are acting dysfunctional.

· Gateway Activity Thread: Is used to determine whether all gateways are still active. Sends a request for an acknowledgment every X seconds, and if that ack is not received within Y seconds,[8] it concludes that the gateway has gone down. If a gateway is determined to be down, it spawns the User Interface Thread to alert Facilities, and also reruns the self-naming process.

3.2.2 Crisis Thread

A Facilities staff member uses the user interface to declare crisis mode on a room, which resumes the thread for that room in the FCS processing unit.

---

[8] These X and Y values are to be set by the system administrator.

This thread sends packets to each camera within that room which request that the `enable_crisis` function be run, and that they send their video frames to BLE+ repeaters, rather than through normal routes, for faster transmission. Our system is well equipped to handle the bandwidth and latency requests, and we have calculated that five fps video would easily arrive within five seconds to the FCS using this modification because all data travels from smart devices to the gateways within one second. A Facilities staff member can disable crisis mode through the same user interface, and packets are sent to each camera within that room which request that the `disable_crisis` function be run, and that packets are sent through normal routes again.

3.2.3 Update Thread

A Facilities staff member can commence a software update on a specific type of device (thermostat, motion detector, or video camera) by entering this in the user interface, which resumes the Update Thread. This thread sends a packet to each device of the given type that requests that the update function be called. When a device has successfully updated, it sends a packet back to the FCS to indicate that it has finished. The Update Thread at the FCS waits until it has received a confirmation packet from each device before returning. If, after a certain amount of time designated by the system administrator, there are still confirmation packets missing from devices, the FCS sends packets to those devices again to request that they be updated. Because of the low-latency setup of our system, the packets should arrive at the devices very quickly. If specific devices should be given high priority for updates, the Facilities staff may also indicate this on the user interface, and these packets can be sent via BLE+ repeaters so they begin even sooner.

When Facilities wants to update the entire FCS, they can select a different function in the user interface. This function instead sends packets that tell all the BLE+ repeaters and gateways to temporarily

store the data that they receive, rather than send it. Then, when the FCS is back on, it sends new packets and the BLE+ repeaters and gateways slowly begin sending their stored data to the FCS.

3.3 FCS Data Storage

A modern database-management system (DBMS) houses the data on the FCS server and provides simplicity. It has a user interface that allows the system administrator to query data, access the camera video, and be alerted of faulty thermometers. The database schema within the modern DBMS is straightforward. There are four separate hash tables within the DBMS. The first hash table contains thermometer data, and maps each thermometer ID to a linked list of last recorded temperatures. The second hash table contains motion detector data, and maps each motion detector ID to a linked list of timestamps of recorded motion. The third hash table contains camera data, and maps each camera ID to a linked list of frame data which that camera captured. The fourth hash table maps smart device id to BLE ID as mentioned in 3.2. This setup provides easy data access for both the system administrator and the main thread. Not only is our design user friendly, but also space friendly, as our system utilizes only a fraction of the total storage of the FCS. The default mode of the modern DBMS is set so data is overwritten every 2 weeks. This supports Facilities' access to one week of camera data, and two weeks of thermometer data. The system administrator can decide however to extend this time frame.

4 Evaluation

Transmission and Storage:

Our system theoretically should offer excellent performance, but it is important that we can prove this to the system administrator before it is actually deployed. Firstly, it is important to prove that the system can handle the communication overhead of thousands of smart devices

sending their data to the FCS at the same time. The maximum amount of data our network can handle in a single second is 1,900,250,000 bytes/sec. [Appendix Eq. 3] Per block, this breaks down to 760,100 bytes/sec [Appendix Eq. 3]. By dividing the number of smart devices by the total number of blocks, we calculated that each block on average contains 6 motion detectors, 6 thermostats, and ⅖ cameras. Together, these devices transmit 11,248 bytes/sec/block, so our system can easily handle typical data flow [Appendix Eq. 3]. It is also important that once a smart device comes online, it can be discovered, and find a route to the FCS in a timely manner. From the time of first initializing the system, it should take only 3.2 s for discovery and routing to the FCS [Appendix Eq. 5]. If the system is fully initialized, and the system administrator installs a new smart device, it should take only 1.1 s for the discovery and routing of that device to the FCS [Appendix Eq. 5]. Upon discovery, it only takes on average 1 second for data to be routed from a given smart device to the FCS [Appendix Eq. 5]. We also needed to check that our FCS could store all of this data flow from the thousands of smart devices. Our system can easily handle the minimum requirements, and could store potentially 11 weeks' worth of data, if the system administrator should want to.

Scalability:

Our system is highly scalable, and could handle the addition of thousands of smart devices, however, some tradeoffs would have to be made. Firstly, with the addition of many new smart devices, the number of connections per repeater could become constrained, as repeaters can only handle 8 connections. This can be solved by simply shrinking each grid block, which increases the total number of repeaters in the system. As a result, the number of devices per repeater decreases. The tradeoff here, is that the overall system cost would increase as the number of repeaters increase. Secondly, with the addition of many more devices, comes more network traffic, which could lead to congestion. Congestion can be easily solved by making a shallower network topology, and increasing the number of gateways.

As a result, the amount of data each gateway has to handle decreases. Once again, scalability is achievable, but the overall network cost due to purchasing more repeaters and gateways increases. Cost is the main limiting factor in the scalability of our system.

Additionally, our system can scale to cover a campus by creating separate grids for disconnected buildings, where the grids cannot be feasibly connected without installing repeaters outdoors. The only constraint when installing these grids, is that there is one gateway per grid.

Fault Tolerance:

Our system is fault tolerant, and can adapt quickly in the face of failure. If a gateway were to potentially go down, our system can respond in about 3 seconds, communicating with all other smart devices that the gateway went down, and that data should be routed to another gateway. When gateways fail, the FCS knows within X+Y[9] (FCS to Gateway Ping and Response Times) seconds because the message sent from the Gateway Availability Thread is not acknowledged. Thus, it decides to run the self-naming process again so that routing can easily continue without this gateway. Although a maximum of ten hops is desired to get to gateways, because the naming scheme allows for devices up to eighteen hops from a device to a gateway, this failure can be mitigated by reassigning the downed gateway's blocks to a different gateway. At the same time, Facilities will be made aware of this failure so that they can send an employee to fix the gateway.

In the case of BLE+ repeater failure, our system continues to operate as normal, as data primarily flow through the BLE repeaters.

---

[9] as mentioned in section 3.2.1, these X and Y values are set by the system administrator

BLE+ repeaters maintain connections with other BLE+ repeaters, and thus are able to tell when another BLE+ goes down because its connection is dropped. At this point, it sends a packet to the FCS with the ID of the downed BLE+. Our network structure ensures that all BLE+ repeaters are within range of several BLE+ repeaters. This means that there is enough redundancy that the system can still function as normal.

Simplicity and Flexibility:

This is precisely one of the strengths of our system; the ease of use for the system administrator. Our system minimizes the work for the system administrator and the grid block layout is simple and easy to understand. Our system responds rapidly and automatically to failures, it supports self-naming which saves the system administrator deployment costs, and the FCS running a modern DBMS supports easy access of data and quick notification in the event of failure. Our system provides great flexibility as well. If the system administrator decided they wanted to increase the frame rate for video camera data, the extra data load would be handled by the BLE+ repeaters which aid congestion. If the system administrator potentially increased the frame rate to a level which caused network congestion, more gateways can be added and then the self-naming process is re-run. Facilities can simply query the FCS using the modern DBMS interface with last minute video data.

Hardware and Costs:

Our system, for MIT's main campus, requires 25 gateways, 2601 BLEs, and 2500 BLE+s. These devices can be fit to a space of any shape using our flexible grid structure [Figure 1]. Given these hardware constraints, our system costs $163,510 [Appendix Eq. 4].

Security:

While the system has not been designed with security as the prominent design goal, there are certain security measures which can, and should be applied to our system. The modern DBMS on our FCS supports password protected, secure login for the system administrator, which should be utilized. This will prevent unauthorized access to all the smart device data. Additionally, each smart device should be set with an administrator password to prevent unauthorized access on a single device basis. Smart devices cannot encrypt data due to their low computational abilities. This is a problem which engineers are grappling with in the real world. As the internet of things expands, security professionals are increasingly concerned about the hackability of the internet enabled home through Bluetooth connected children's toys, webcams, refrigerators, and more.

5 Conclusion

Our system is reliable and fault tolerant, and fits the specifications necessary for temperature regulation, and security. By utilizing a grid topology built upon the small networks of blocks, the system is able to intelligently reason where to send data, quickly locate failures, and easily route around any potential network issues. Additionally, the use of a grid simplifies the calculation of system requirements and increases our system's ability to perform well even in high bandwidth conditions. Our system can also perform software updates on any part of the system without any noticeable changes in performance. The system administrator can easily monitor the state of the system through our intelligent FCS functions. There is also an added reliability mechanism which addresses the problem of Bluetooth reliability through a protocol of confirmations. With these design features in place, this system is tailored to be easily implementable and scalable, fault tolerant to protect users, and extremely reliable in the face of emergency.

6 Author Contributions

Each of the team members worked extremely hard brainstorming, designing, and editing our System Critique. While each of us contributed to all aspects project, the breakdown of writing sections was as follows:

- Libby Aiello - FCS Processing, System Failures
- Bobby Rauch - FCS Data Storage, System Diagrams, Evaluation, DP Document Formatting
- Kunal Tangri- Network Topology, Communications Protocol, Calculation Appendix

7 Appendix

1. Worst Case Cumulative Smart Device Transmission:
    a. 15,000 thermostats * 4 bytes/sec + 15,000 motion sensors * 4 bytes/sec + 1000 cameras * 28,000 bytes/sec = 28,120,000 bytes/sec
2. Number of blocks needed to fit MIT Main Campus:
    a. MIT Main Campus is 2,250,000 square feet, and each block can cover 900 square feet
    b. 2,250,000 / 900 = 2500 blocks
3. Network Congestion Proof:
    a. Total network throughput:
        i. A system of 2,500 blocks will contain 2,601 BLE repeaters and 2,500 BLE+ repeaters
        ii. In total, the network can support 250,000 bytes/sec * 2601 BLEs + 500,000 bytes/sec * 2,500 BLE+s = 1,900,250,000 bytes/sec
        iii. Each block can support about 1,900,250,000 bytes/sec / 2,500 blocks = 760,100 bytes/sec
        iv. Each block has an average load of 28,120,000 bytes/sec /2,500 blocks = 11,248 bytes/sec/block
        v. Since we ensure that data are propagated to the nearest gateway within a second, and each block can process 760,100 bytes/sec while the average load on each block is only 11,248 bytes/sec, we argue that out system will not experience any congestion from the aggregate amount of data running through the network
    b. End point congestion of gateway given 10x10 blocks:

i. Congestion can also be an issue at the repeaters closest to a gateway, as all data from the surrounding grid must pass through these repeaters to get to the gateway

ii. Our system needs to support 1,000 cameras and it consists of 2,500 blocks (⅖ cameras per block), and each gateway serves 100 blocks (40 cameras per gateway)

iii. Since each gateway is surrounded by 4 BLE repeaters, each BLE repeater is responsible for relaying data to the gateway from one quadrant of the 10x10 grid

iv. One quadrant will contain 10 (⅖ cameras per block * 25 blocks per quadrant) cameras on average → each BLE repeater surrounding gateways will have to handle 28,000 bytes/sec * 10 cameras = 280,000 bytes/sec on average

v. Each BLE repeater can support 250,000 bytes/sec, but with the BLE+ fallback mechanism in place for when BLE repeaters can't handle the amount of traffic, the total amount of data throughput one quadrant can utilize increases to 750,000 bytes/sec (BLE+s can handle 500,000 bytes/sec)

vi. Each gateway endpoint can handle 750,000 bytes/sec of data flowing through them, and, in the average case, there's 280,000 bytes/sec that need to be supported. So, our system is capable of preventing endpoint congestion, and still has plenty of additional space to support cases worse than the average

4. Total System Cost:
    a. $500 per gateway * 25 gateways = $12,500

b. $10 per BLE * 2601 BLEs = $26,010

c. $50 per BLE+ * 2500 BLE+ = $125,000

d. Total cost = $12,500 + $26,010 + $125,000 = $163,510

5. Time for Self-Naming Initialization Process:

    a. Initial Setup:

        i. It takes 200ms for Gateways to connect to BLE+s and for BLE+s to connect to other BLE+s - this is because IDs must travel two hops (through the BLE intermediaries) to connect, and each hop takes 100ms

        ii. Since our system enforces a ten-hop maximum from gateways to blocks (at initialization), it takes a total of 10 hops * 200ms = 2,000ms to establish connections with all BLE+s and gateways in the system.

        iii. It takes BLE repeaters 100ms to form connections with their primary BLE+ repeaters because the primary BLE+s are one hop from the BLE repeaters

        iv. It takes smart devices 100ms to form connections with their default BLE repeaters because the default repeaters are one hop from smart devices

        v. Next, the smart devices send their data initialization packets to the FCS. This process takes 1,000ms since the packet must travel ten hops to reach the FCS

        vi. The total initial setup takes 2,000 + 100 + 100 + 1,000 = 3,300ms

    b. Addition of Smart Device:

        i. A new smart device must first form a connection with its default BLE repeater, which takes 100ms

ii. The smart device then sends its initialization packet to the FCS, which takes 1,000ms to travel the ten hops

iii. The total time to add a smart device to the system is 100 + 1,000 = 1,100ms