Smart Campus: A Reliable Data Management System for MIT

Sabina Chen (sabinach), Joseph Torres (jmtorres), Justin Yu (justyu)
Recitation Instructor: Tim Kraska
Mon/Wed 1-2 and 2-3pm
Word Count: 6,656

April 22, 2018

Contents

1 Abstract

Smart Campus is a highly scalable and resilient framework for implementing data transmission capabilities to a network of electronic devices on MIT's campus. The primary use case for this data will be to improve the quality of life for MIT's occupants, with the added benefit of reducing energy costs. Therefore, Smart Campus follows a similar level of priorities. In order to do so, Smart Campus imposes stringent fault tolerance without sacrificing cost efficiency (as otherwise potential cost-saving benefits are lost).

The system is constrained to follow certain properties regarding bandwidth and path availability, and propose a design which amortizes the cost of adding and handling failed devices on the network. In doing so, we permit an amorphous network that can fit most campus architectures. Although this compromises simplicity in implementation, Smart Campus is far more cost-efficient and scalable than alternatives which strictly prioritize fault tolerance. When implemented correctly, Smart Campus is an optimal choice for modernizing a campus to support smart devices.

2 Project Summary

MIT's campus spans 166 acres and is over a mile from end to end [1]. The devices of interest (15,000 thermostats, 15,000 motion detectors, and 1,000 video cameras) are spread across the campus in an unpredictable pattern. This complexity is furthered by the unique architecture of MIT's buildings, e.g. the Stata Center. In order to address these difficulties, we employ an amorphous network topology (a hierarchical model which uses an extended star topology) that minimizes the total number of gateways by subjecting our network to the following constraints:

- All network devices can handle worst case throughput (i.e. the size of buffers of network devices are weakly decreasing).
- Each smart device has more than one gateway it can connect to and more than one path to the same gateway.
- When the hop length between the closest gateway to a smart device and that device is too large, we place another gateway.

This architecture prevents network congestion under normal operation and allows for a number of use cases (such as system-wide updates, crisis mode, and more). The network topology also simplifies the process of routing, as the connections between a gateway and its reachable smart devices create an implicit graph which can be used in routing protocols. Although drastic changes to a single gateway's connected devices may necessitate modifications to the network on the gateway level (which is costly and requires setup time), amortized costs for adding and removing individual network and smart devices are minimal.

Each network device communicates with the Facilities Centralized Server (FCS) in a unicast manner (one-to-one), and vice versa. Therefore, each network device simply stores the next hop to the FCS for incoming FCS traffic and the

next hop to all smart devices it can access for outgoing FCS traffic. In order to keep track of and update routing tables, the FCS stores and uses the implicit graphs that each gateway is the centroid of.

The FCS maintains multiple threads to concurrently process incoming and outgoing information. It also leverages its local file system and relational databases to store long-term information and manage device failures. Additional logic is installed in the FCS to utilize the network in the cases of system updates, crisis mode, failures, and more. We project that the overall cost of the project will be roughly $1,260,000 (discussed in Section 7), with minimal long-term costs due to the energy cost savings this system enables.

The major contribution of Smart Campus is a fault-tolerant yet scalable system that does not sacrifice cost efficiency. Section 3 defines how the various system components, including smart devices and network devices, are set up and utilized. Section 4 discusses the placement of network devices and how information travels along the network. In section 5, data processing on the FCS, as well as information storage, is explained in greater detail. Section 6 explains how the system operates under a variety of use cases. Lastly, although care is taken to evaluate and justify components as they are introduced, section 7 contains additional details regarding the system's performance and security considerations.

3 System Components

3.1 Device Identifiers

Both network and smart devices have 32-bit identifiers and 32-bit timestamps that are stored in the 64-bit packet header (Figure 1). Device identifiers encode type, location, and unique identifier info. Device types are encoded in 3-bits, as there are 3 types of smart devices and 3 types of network devices. With a total of around 200 buildings, 8-bits will encode the building [1]. This leaves 21-bits to encode a unique identifying serial number, enough for well over a million devices. This enables 1-to-1 communication between the FCS and any device.

Byte	0								1								2								3							
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Content	Device type			Building Number																												

Byte	4								5								6								7							
Bit	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Content	32 Bit Timestamp																															

Figure 1: Device packet header specifications

3.2 Smart Devices

3.2.1 Thermostats

Thermostats send packets containing current room temperatures to the FCS every 5 minutes through a series of BLE and BLE+ repeaters. The FCS has data on each thermostat that is no more than 5 minutes old.

3.2.2 Motion Detectors

Motion detectors send packets to the FCS every 500ms with data regarding the last time motion was detected, which is then used to turn lights on and off as well as adjust room temperatures.

3.2.3 Video Cameras

As video cameras provide the most sensitive data, it is important for these devices to have redundant paths to the FCS, especially during crisis mode. Each camera captures 5 frames per second, one of which is sent to the FCS. Frames are broken up into 1,400 packets due to size constraints on an individual packet. As specified, the most recent week's worth of data for every camera will be stored on the FCS. With 1 frame being sent a second, this will result in at most 17 TB[1] of stored frames on the FCS at any given moment. The cameras themselves also have a 4 GB buffer which will be used to store data in the event of a network failure. When storing frames on the buffer, we avoid saving duplicates by using `is_equal(frame_id_1, frame_id_2)` with each new frame on the most recently stored one. If it's false we do not save the frame, whereas if it's true we simply make a pointer for that timestamp to the previous frame. When deleting frames, if the most recent frame is a pointer to a frame we are deleting we will replace that pointer with a copy of the frame that will be deleted (in order prevent null pointers).

3.3 Network Devices

3.3.1 BLE and BLE+ Repeaters

BLE repeaters are cost-effective and the first step in getting data from thermostats and motion detectors to the FCS. Similarly, BLE+ repeaters will always be the first point of contact for video cameras (since BLE+ repeaters have persistent storage which is useful in the case of device failures). The branching factor[2] of BLE+ repeaters will be small in order to improve redundancy, so it will often be the case that for each BLE+ repeater there will be another one nearby that covers the same set of devices as it. Stacking BLE+ repeaters in this way ensures higher system reliability as more routes are made available for smart devices.

3.3.2 Gateways

Gateways are placed sparingly due to their high cost. Similar to repeaters, gateways also have decreased signal strength through walls. When considering vertical reach, gateways are placed on every nth floor, where n is the average hop count between a smart device and the FCS. Based on the system design, the average hop count for Smart Campus is roughly 5 hops (discussed in more detail later), so vertically co-linear gateways will be placed every 5 floors (there may be multiple gateways per floor, or one gateway on each floor but on different

[1] With 1,000 cameras each sending 28 KB frames a second, and $7 \cdot 24 \cdot 60 \cdot 60 = 604,800$ seconds in a week, $1,000 \cdot 28 \cdot 604,800 = 16,934,400,000$ KB stored per week. This is equivalent to 16.9344 TB, or 17 TB.

[2] "In computing, tree data structures, and game theory, the branching factor is the number of children at each node, the outdegree[5]."

verticals). Gateways are placed in regions central to densely populated devices, but are placed while keeping the aforementioned constraints in mind.

3.3.3 FCS

The FCS is located in the basement of MIT's Building 7 due to its centralized location on campus. This reduces the length of the longest wired connection from a gateway to the FCS.

4 Network Architecture

4.1 Properties

The topology of Smart Campus' network is constrained using a number of useful properties that will assist in evaluating the network's performance and simplifying routing. Therefore, we create graphs for each gateway where that gateway is the centroid and its graph follows the listed properties. In the following properties, let b^* be the maximum bandwidth a network device supports and n be the maximum number of connections to that device. Let N be the set of all network devices on campus, S be the set of all smart devices on campus, and k be the graph sparsity coefficient where $0 < k \leq 1$ (discussed in a moment).

Property 4.1 (Bandwidth Bottlenecks) *Given that the network is arranged in an extended star topology, then $\forall d \in N$, it holds that the bandwidth of an outgoing connection to the FCS from d is limited by:*

$$\frac{b^*}{n} > k \sum_{i=1}^{n-1} B(x_i)$$

where x_i is a connected device contributing to incoming traffic and $B(x_i)$ is the bandwidth of device x_i.

Property 4.1 takes into consideration the fact that overloading a network device with many components may congest the outgoing bandwidth to the FCS. If devices are connected while guaranteeing Property 4.1, then network device buffer sizes will be weakly decreasing and throughput to the FCS will always be less than the bandwidth of that connection. This will allow us to vastly improve redundancy as this nearly eliminates congestion from the network.

Property 4.2 (Path Redundancy) *$\forall s \in S$ there exist at least 2 gateways s may connect to, as well as at least 2 paths to get to each of these gateways.*

Property 4.2 is an essential component of Smart Campus' resiliency. Although employing this property introduces a financial cost, we maintain that this is one of the smallest hits one can make to cost efficiency while still emphasizing fault tolerance. In order to ensure backup paths will not congest the network, we introduce k as a means to making the graph more sparse. We will initially set k to $\frac{1}{4}$. This sufficiently enables backup paths to handle device failures, while still permitting room for additional devices. As devices are added, k will increase. Once k reaches $\frac{1}{2}$, we will place another gateway in the area (while still maintaining our properties) to set k back to $\frac{1}{4}$; occasional, expensive gateway additions are offset by fast and simple network and smart device additions, resulting in amortized scalable network modifications.

Property 4.3 (Latency Limiting) *If any device has a hop length greater than or equal to 10 to its nearest gateway, another gateway will be placed near.*

The last major property covers edge cases where isolated devices may be far away from a gateway; this can cause them to suffer from high latency. To remedy this, we will place a gateway close to any device which has a sufficiently large hop count; this in turn provides better scalability and fault tolerance to the existing network (we will also ensure that Property 4.2 is held).

4.2 Topology

There are three general location types where smart devices are placed: small rooms, hallways, and large rooms (such as common areas and large lecture halls). Small rooms primarily include thermostats and motion detectors, whereas hallways and large rooms contain all three smart devices. The network will be a hybrid between a star topology and an extended start topology to accommodate the campus layout. Small rooms (Figure 2) and large rooms (Figure 3) will use BLE and BLE+ repeaters as centroids in a local star topology, whereas larger clusters in the extended star topology use gateways as centroids. Hallways will contain BLE+ repeaters in order to connect a centralized gateway to cameras (which are typically more common in hallways) and isolated rooms (Figure 4).

Smaller stars are connected to an extended star in order to form a graph connecting all smart devices to gateways (Figure 5). This way, the system is more flexible and scalable to arbitrary building architectures, as we can extend our resilient and cost-efficient design to fit any shape using the above three location types. The primary edge case for this design would be long, fringe hallways. In this case, we use BLE repeaters to chain the isolated devices to a gateway while still maintaining Property 4.3. BLE repeaters are used to chain as their batteries never fail and they are cheaper, making them a better option for both cost and fault tolerance.

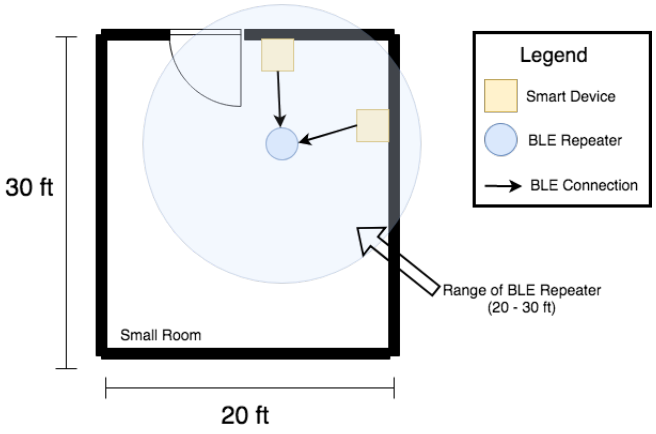

Figure 2: Placement guidelines for a local star network topology of a small room. Smart devices communicate with BLE repeaters in order to forward data.

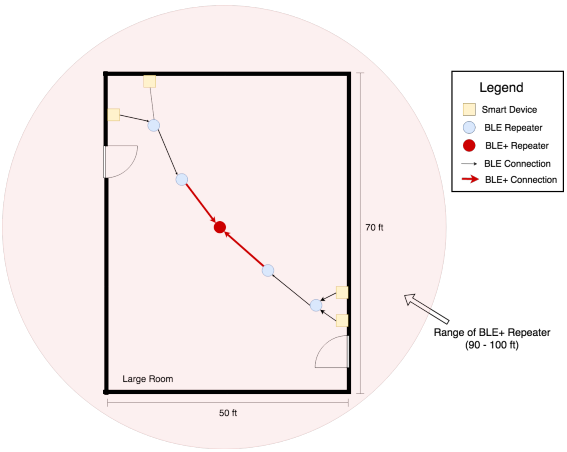

Figure 3: Placement guidelines for a local star network topology of a large room. Smart devices communicate with BLE repeaters, which then forward data to BLE+ repeaters.

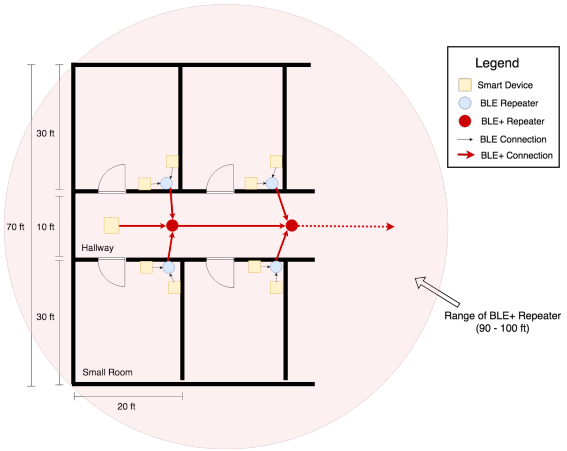

Figure 4: Placement guidelines for a local star network topology of a hallway. BLE repeaters communicate with BLE+ repeaters, which then forward data to other BLE+ repeaters, creating a chain of BLE+ that eventually connect to the gateway.

4.3 System Initialization

4.3.1 Network Device Placement

Placing gateways given the proposed constraints is not trivial. In order to accomplish this task, it is important note that system initialization considers a set of already existing smart devices. As such, we will employ a bottom-up hierarchical clustering algorithm[6] to determine dense areas where gateways can be placed. The properties outlined in Section 4.1 will be used in the metrics and linkage criteria of defining clusters. Once gateways are placed, we follow

the rules outlined by our topology and network devices to place enough network devices to ensure our properties, and thus redundancy, are met.

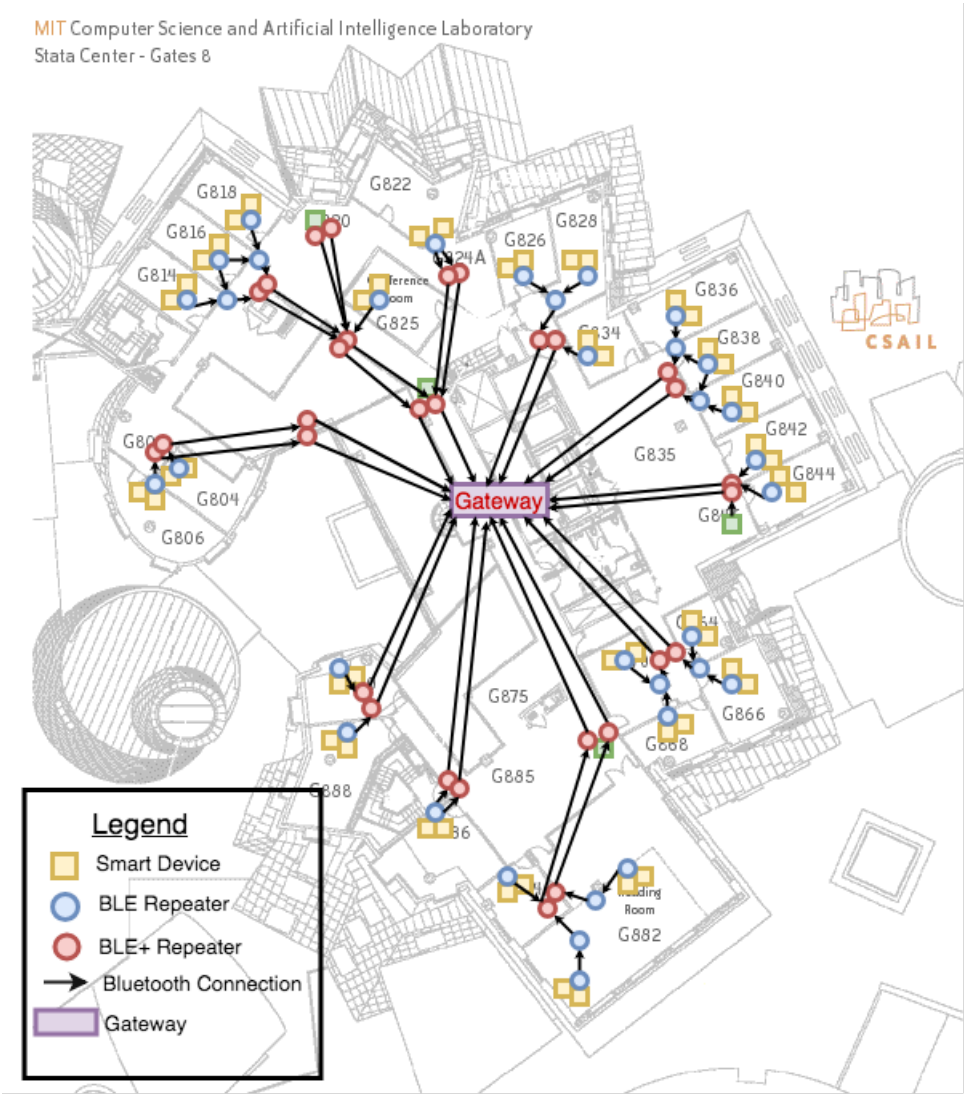

Figure 5: Placement example for an extended start topology on a real MIT building (ie. Stata Center[8]). Smart devices communicate with BLE and BLE+ in local stars, which connect to larger clusters in the extended star topology that use gateways as centroids. This extended star topology allows for a system that is both flexible and scalable to arbitrary building architectures.

4.3.2 Node Discovery

Node discovery is a two step process. First, we will consider the case of system initialization. At the time of system startup, the FCS will begin a broadcasting protocol starting from each gateway that will eventually lead to the creation of

routing tables. Each gateway will broadcast to connected devices using breadth first search (BFS). The closest smart devices that fall within the defined gateway graph properties will be considered. When network devices which connect only to smart devices are encountered, they will broadcast a path composed of 32-bit UIDs (the device and its own) upstream; this path is stored in the body of the packet. This propagates upwards as network devices append their own ID to the path until a hop length of 10 is exceeded or a gateway is encountered (network devices will perform this check). The FCS will aggregate all paths for each gateway and store them locally in a relational database table. Additionally, we will also make two extra tables: one which serves as a routing table, mapping every smart device to the gateway which provides the shortest path (determined via a distance-vector routing protocol), and another which maps every device to a set of all graphs which it is a member of. All of these operations should be linear with respect to the size of the input, except for finding the shortest path in each graph, which is polynomial in the size of the input (and operates on many small, sparse graphs). This information is then distributed to all network devices in order to update their routing tables.

The second step of node discovery is when a device is added. In an existing graph, the addition of a new device is fairly trivial. Since smart devices can only broadcast their 32-bit UID, it will simply connect to the closest available network device. If it cannot connect to a nearby device (because it is saturated with connections or isolated), then additional devices will be added as well. In this case, we broadcast from the new device and an existing network device will forward this to the FCS (using its own routing table), where the FCS will append the entry to its tables and respond with an updated routing table. If adding a device causes a gateway to be too saturated, we repeat the first step to distribute the load (and potentially add new gateways if needed). In the case where some devices are not done initializing in step 1 (i.e. first setup), then we simply keep updating the graphs until the network is stable. This method of node discovery allows for simplified scalability since the network is designed to support adding new devices with ease.

4.3.3 Misconfigured UIDs

Smart and network device UIDs are 32-bit strings composed of the device's type, location, and identifier. Device UIDs are established at system initialization or during the addition of new devices. When a new device is added to the network, the FCS performs various checks to ensure that the smart device has the correct UID format configured. Since the specifications state that only smart devices may be misconfigured, we will assume that network devices are always configured correctly. These corrections will ensure the system is correctly configured and less prone to failure.

Incorrect UID setups of a smart device are detected by the FCS. A UID with an incorrect device type is detected when the smart device sends data incongruent to its declared type (e.g. `get_time()` is not specified on a thermostat). Incorrect location bits are detected by comparing the location of the new smart device with its nearest BLE or BLE+ repeater. In general, a smart device will connect to a network device that is in the same building or an adjacent one (if

this is not the case, then the device is misconfigured). The identifier at the end of the UID is simply to designate device uniqueness, and as the specification ensures unique UIDs even after misconfigurations, this is not a concern.

4.4 Routing

Once devices are set up, routing is straightforward. There are two directions to consider for communication: upstream and downstream. In either case, the protocol is unicast, so a smart device which wishes to send information to the FCS needs only to know the next hop it should take to get to the closest gateway (there may be entries for multiple gateways in each smart device's routing table, but only one hop for each gateway). Similarly, network devices only know the next hop to connected gateways. Downstream, on the other hand, is initially concerned with which gateway the FCS must use to communicate with a smart device. After performing a lookup in its table (mentioned earlier), it sends the packet to a gateway, which contains information on the next hop for each device which is a member of its graph. As information trickles down, the routing tables get smaller as network devices only contain information regarding the next hops needed to communicate with a smart device (Figure 6).

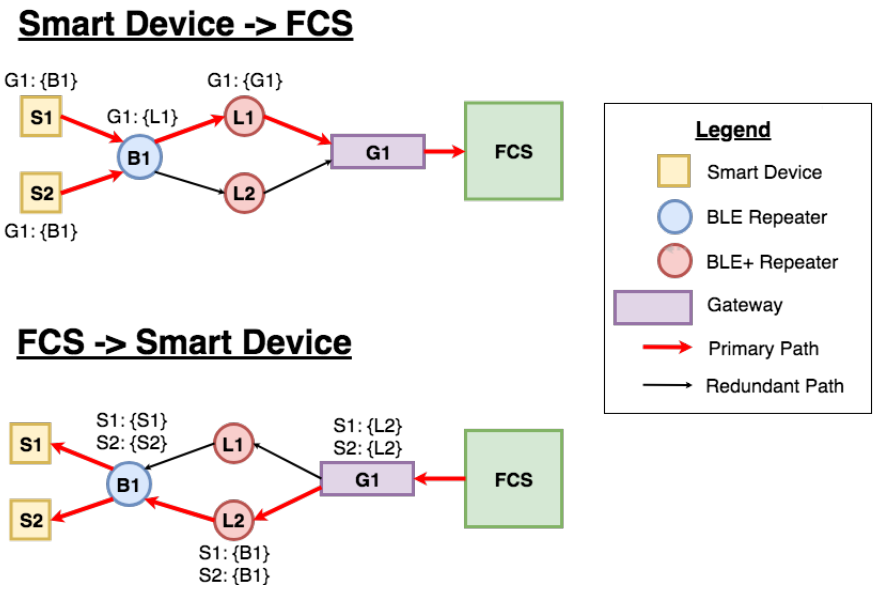

Figure 6: Routing table examples for each smart and network device. From smart device to FCS, devices contain the next hop to the nearest gateway. From FCS to smart device, device contain the next hop along the shortest path to the specified smart device.

4.4.1 FCS Routing Storage

As mentioned previously, there are three database tables on the FCS which will assist in routing. The first is a table which maps each gateway to a single graph of devices for which it is responsible for. The second is a mapping of each device

on the network to the set of all gateway graphs it is contained in. The last is a routing table for every smart device. All of these tables are updated upon adding and removing devices from the system (Figure 7).

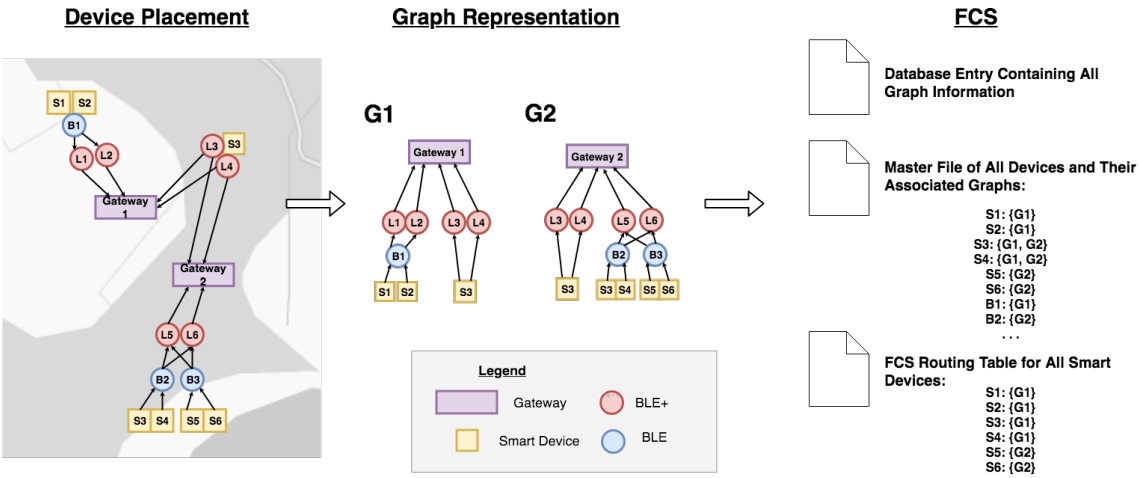

Figure 7: FCS file system for storing routing and graph information.

4.4.2 Routing Tables

Routing tables are set during initialization and only updated during device failures or device additions. Therefore, routing tables mostly remain static and act as a look up table during network routing, decreasing computation time and allowing for quick look ups. Because there exists a path from each smart device to at least two gateways (as specified by our implicit graph constraints during system initialization), the routing tables include the next hop to the primary gateway, and the next hop to the secondary gateway. The size of the routing tables is weakly decreasing as a device gets farther away from a gateway (in order to preserve space and improve performance).

4.4.3 Device Failures

When a device is added or removed from the network, affected devices are able to dynamically update their routing tables with help from the FCS. As each device stores a routing table containing both the next hop to the primary gateway, and the next hop to a secondary gateway, if a device along the primary path goes down, affected devices along the route simply switch to using the secondary route (Figure 8).

Whenever data arrives at the FCS, it is stored in a timestamped entry in a table (sorted by time) in a relational database. A thread exists on the FCS which reads the oldest entries in the table every minute and pings any device which has stale data. If no response is received, we assume the device is down and update all routing tables accordingly (and notify facilities to inspect the device). Note that if a network device is an articulation point[7], then the entire tree will be considered down; this will be noticed by Facilities and can be

handled accordingly. According to the specifications, a BLE repeater will never fail, and a BLE+ repeater will only fail if its battery dies (which occurs every 6 months to 1 year and takes 5 minutes to replace). Gateways last between 1 and 5 years and take 1 hour to replace. Our system can reroute packets in most cases, and will only fail if all gateways a device can connect to are down in the same period (which is extremely unlikely); even in this case, data will simply buffer and eventually be sent in the required amount of time.

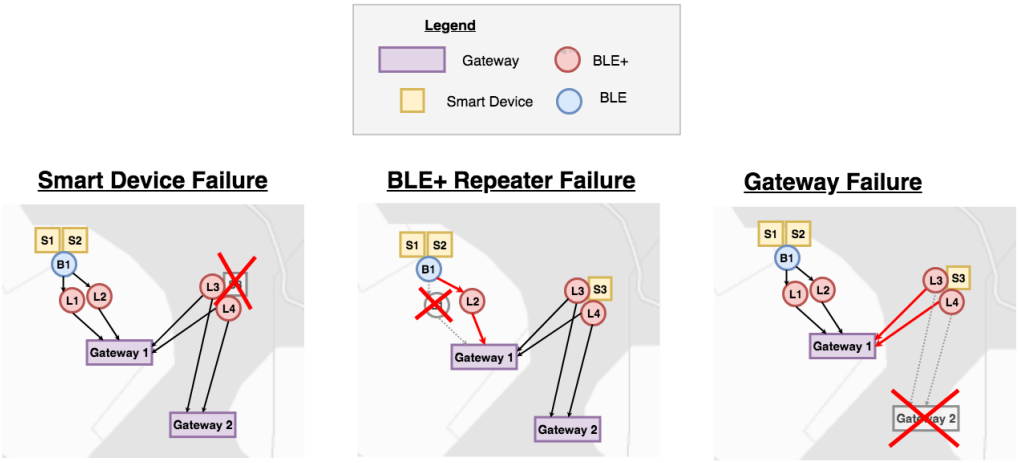

Figure 8: Examples of each type of device failure: Smart Devices, BLE+ Repeaters, and Gateways. As specified by Property 4.2: Path Redundancy, devices are connected to at least 2 gateways and at least 2 paths to each of these gateways. If a device along the primary path goes down, affected devices along the route simply switch to using the secondary route.

4.5 Protocols

4.5.1 Protocol 1: Simple Data Fetching

The FCS retrieves data from thermostats and motion detectors by calling `get_temp()` and `get_time()`. Each packet contains a 64-bit header—32 bits containing the timestamp of the reading and 32 bits containing the UID of the device.

Thermostats send a 32-bit temperature reading every 5 minutes to the FCS. Motion detectors instead send a packet which includes a 32-bit timestamp of the last time motion was detected. If the light is off and there is motion, it is quickly detected since updates are sent every 500ms. To reduce the amount of stored data however, the FCS stores 1/600 data points to keep data with 5 minute granularity. The rapid detection of movement in a room assists in keeping fresh data on the occupancy in the room. This enables resilience as there is a quick detection of false positives. Cost-efficiency is also enforced as a lack of occupancy is detected faster (and thus the FCS is notified sooner that it can lower temperatures, thus lowering energy costs).

4.5.2 Protocol 2: Distributed Frames

For videos, an individual frame is 28-kbytes which is distributed across 1400 20-byte packets when sent. The packet header uses 64 bits — 32 for a time-stamp of the frame and the rest for the UID of the device. The body uses 11 bits for the order of the frame and the rest for data. Frames are reassembled and processed at the server. One frame per second is sent (by storing the frame from `get_latest_frame()`), but in order to not clog bandwidth, it is not guaranteed that all frame data is sent at once. Instead, data is sent intermittently to the FCS over a 5 hour interval, using the cameras' buffer as a temporary storage for frame data. Response from the server are reconstructed similar to how they are constructed. Entire frames are sent before sending other ones (i.e. resend dropped packets, unlike in simple data fetching). However, not all frames will be transmitted entirely on first attempt. Packet losses may occur in the network. If the FCS does not receive an entire frame within the maximum 5 hour period, it will ping the video camera and request any missing packets. With 4 GB buffer size, video cameras can store up to 39 hours of frames at one frame/sec granularity, giving ample ability to retrieve previous frames.

4.5.3 Protocol 3: Prioritized Channels

In the event of a critical incident that requires crisis mode, the system grants priority traffic to cameras in zones which have crisis mode enabled. To do so, BLE and BLE+ repeaters prioritize traffic from crisis cameras, which increases the rate at which they send frame packets (thus necessitating other cameras to buffer more video data on local storage). This is done by assigning a priority bit at the beginning of the packet body, which is then checked at each network device. The data from other smart devices will not cause a bottleneck in this case as the system is underloaded and network device buffers are weakly decreasing in size.

4.5.4 Protocol 4: Reliable Packets

In most cases, devices prioritize the nearest BLE repeater, which in turn prioritizes the closest BLE+ repeater. If a prioritized packet is encountered at a repeater, then it is pushed to the front of the buffer queue. In the event a queue is full, simple tail drops are performed (prioritized packets will take place of a non-priority packet).

5 Data Processing and Storage

5.1 Threads

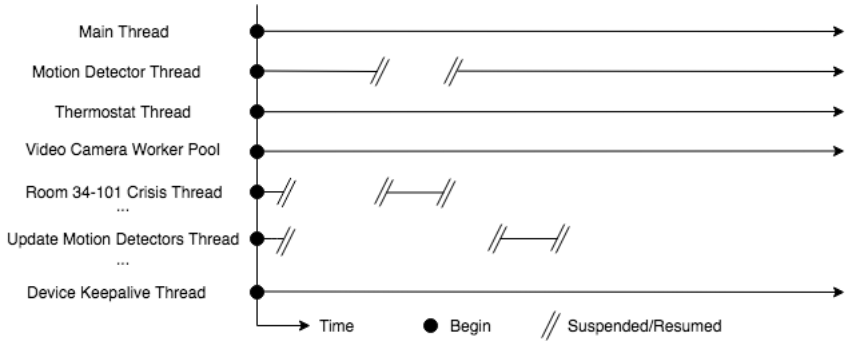

Figure 9: Thread Examples on the FCS

5.1.1 Main Thread

The main thread runs on the FCS and handles incoming data. Depending on the type of incoming data, the information is passed to different threads. Metadata in the header of each packet is checked upon arrival. For motion detectors or thermostats, a message is passed to a designated Motion Detector and Thermostat thread, whereupon an atomic write of the data is performed to the database after it traverses a queue. If conflicting data arrive simultaneously from the same sender device (with the same timestamp, indicating an error), one packet is picked at random to maintain overall performance. For video data, we hold a worker pool of about 25 threads. We assign an idle thread to process a new frame at a certain timestamp. When a frame packet arrives, it is added to a self-sorting queue (based off of order) that exists on its corresponding thread. Once the queue fills (its set to the expected length in packets of a frame), the system writes to the local file system and atomically writes to the database after which the thread is returned to the pool. If there are no idle threads, we append the packets to a queue.

5.1.2 Device Keepalive Thread

For all devices, an additional thread runs which executes a *cron*[3] job that examines the oldest entries for each device in the database. In doing so, if it detects a device has been idle for too long (e.g. longer than 5 minutes for all devices except gateways, which take 1 minute), then it will ping the device requesting a response. If no response is received within a minute, the device is assumed to be dead, routing tables are updated, and facilities is notified.

[3] *Cron* is a time-based job scheduler in Unix-like operating systems. *Cron* is used to schedule jobs (commands or shell scripts) to run periodically at fixed "times, dates, or intervals [3]."

5.1.3 Crisis Thread

Each room has a paused thread known as a crisis thread. When crisis mode is enabled, the thread resumes and all cameras in that room are sent a request for `enable_crisis(camera_ID)`. This does not change how the main thread processes video data, but instead increases the rate at which frames are received (which is handled by the network) and modifies frame packets to also contain a priority bit. The FCS will simply have a larger load to handle. When crisis mode is over for a room, `disable_crisis(camera_ID)` is called for each camera in that room and the thread is paused.

5.1.4 Update Thread

For each type of device, there is a paused thread for updates that resumes when `update(device_type, software_binary)` is called. While this thread is running, the FCS prevents data from updated smart devices from being sent across the network until all devices of that type are updated. A queue on this thread that is the length of the number of devices that initially need updates is used. The thread pauses once all devices have sent an ack of their update. Afterwards, the thread is killed and normal function resumes.

5.2 Data Storage

FCS stores data using a relational database to improve lookup speeds. Two week's worth of thermostat and motion detector data are stored per device in a database. Video data is instead stored on the local file system, maintained in a hierarchy which is broken down by location (i.e. building number, followed by room, followed by a UID). The UID is a folder that contains frame data, where each frame is named with respect to its time stamp. The relational database has an entry for each camera's UID and the data stored will simply be a reference to the location of the frame data on the file system. Video data keeps frames for up to a week. Whenever data in the local file system or database is updated, the oldest piece of data is removed in favor of newer data.

5.3 Detecting Device Anomalies

Anomaly detection occurs as a background process. There is a fork of the main thread responsible for checking the database for anomalies. This program runs every minute. As a part of the main motion detector thread, if it is noticed that there has been no activity in the past 15 minutes for a given location, then a request to reduce the room's temperature by 2.5 degrees will be issued. A room will be allowed to vary from the desired temperature by a max of 7.5 degrees in either direction to ensure that when users enter the room, the temperature will not lead to excessive discomfort.

5.4 Inconsistent Readings

On average there is one motion detector and one thermostat per room (except for large rooms which may have multiple). In the case that motion detectors in the same room give inconsistent readings, such as if one reports that the room is in-use while the other does not, the system adjusts according to the motion

detector in-use, as the user may potentially be in an area that is only covered by one motion detector. In the case that the thermometers in the same room give inconsistent readings, the system averages the temperatures and adjust the HVAC based on this averaged value in order to account for varying temperatures between different sections of the room.

6 User Modes and Use Cases

6.1 Normal Operation

The majority of the system runtime is spent in normal operation, making it a priority over other modes. During normal operation, the system continues to monitor, process, and store data from the motion detectors, thermostats, and video cameras using the Main Thread on the FCS (as discussed in section 5.1.1).

6.2 Crisis Mode

The second most important mode is crisis mode, during which cameras in a room are set on priority to monitor a campus crisis. During crisis mode, the data generated is vital and is sent and processed by a corresponding Crisis Thread in the FCS (Section 5.1.2). This mode disables smart filtering in that camera using the `stop_filter()` command, and increases the number of video frames sent per second to the FCS, enabling video feed to be processed in real-time. Furthermore, using Protocol 3 (Section 4.5.3), frames from that camera will be marked as higher priority, allowing priority frames to bypass queues at repeaters. The redundancy of devices ensures that the network does not suddenly become congested by real-time video data. At the same time, the network still allows other non-priority smart devices to be processed concurrently using the Main Thread in the FCS (Section 5.1.1). After the crisis, smart filtering is re-enabled using the `start_filter()` command, and the number of video frames sent per second to the FCS from crisis cameras are decreased back to usual operating speed, restoring equilibrium to all video cameras' local storage.

6.3 Server Maintenance

Another low priority mode, due to its infrequency, is server maintenance. Before server maintenance begins, all cameras are set to automatically filter before buffering in order to reduce the number of frames captured per second. Furthermore, the cameras will also be set to automatically detect similar images using `is_equal(frame_id_1, frame_id_2)`. Frames detected as similar images will be deleted using `delete_frame(frame_id)` to filter redundant data. This procedure will enable more hours of camera footage to be stored in local camera storage during server maintenance. Older frames will be dropped as camera storage overflows. Additionally, motion detector and thermometer data will be buffered and stored in local BLE+ repeaters so that data is not lost.

6.4 Software Updates

Software updates is the last of the low priority modes and will be done on the FCS server as specified in the FCS Update Thread (Section 5.1.3). To guarantee

that every device gets updated, devices are turned off and queued to be updated via the FCS Update Thread, and turned on again after the software has been successfully updated.

7 Evaluation

With 15,000 thermostats, 15,000 motion detectors, and 1,000 video cameras, there is around 31 MB of data sent between smart devices to the FCS every second. The bulk of data sent comes from the 28 KB frames sent by each video camera as seen below, with the rest from 96 byte packets from thermostats and motion detectors.

$$28 \cdot 10^3 \cdot 1000 + 15000 \cdot 96 + 15000 \cdot 96 \approx 31 \text{ MB}$$

During crisis mode, the network traffic is not increased since packets from devices in crisis mode are simply prioritized at buffers. During server maintenance, cameras will buffer their images locally while thermostats and motion detectors continue to send traffic for BLE+ repeaters to store. Therefore, since most data is buffered at this time, network traffic during server maintenance will decrease to less than 3 MB/sec. During software updates, network traffic will be increased to account for 15 MB updates.

When each smart device is added, there is a period of discovery where the correct path to the FCS is established. With an average of 5 hops from each smart device to the FCS, and given that discovery requires one back and forth transmission from smart device to the FCS, discovery will take 1000 ms given 100 ms latency for each hop. By the same logic, the latency for data transmission between smart devices to the FCS is 500ms. On average, small hop distances not only reduces latency but also contributes to fault tolerance by reducing the number of components that can fail between a smart device and the FCS.

With all of this data, the FCS will eventually run out of storage. It has 100 TB available. With roughly 31 MB accumulated every second, it will take around 5.5 weeks to fill up the FCS. Since this is more data than can be stored and that is needed, older data will be discarded to make room for new data every week. Increasing the capacity of the FCS is not necessary because the oldest data required to be stored is one week's worth.

7.1 Scalability

If a large quantity of smart devices is added, our system can handle such an influx given appropriate additions of gateways and repeaters. For example, if 5000 new thermostats and motion detectors are added, there will be a .48 MB increase in network traffic across the network which can be easily handled.

$$5000 \cdot 96 \text{ bytes} \approx .48 \text{ MB}$$

With 500 video cameras added, there will be a 14 MB increase in the load across the network every second. Since gateways have 1 GB/sec throughput to the FCS, this increase can be sustained given the addition of repeaters to get the data to the gateways.

$$28 \cdot 10^3 \cdot 500 \text{ bytes} \approx 14 \text{ MB}$$

Dynamic rerouting of smart devices will adjust for the increased load by finding new optimal paths. This makes the system more scalable and also fault tolerant since more optimal paths reduce hop counts. For new buildings, the process will be similar to the beginning initialization process of gateway region partitioning and repeater distribution to reach the smart devices.

7.2 Failure and Update Response Times

In response to gateway failure, smart devices will be dynamically rerouted to another gateway. Our topology has assures such a second gateway to ensure fault tolerance. Gateway timeout is 1 minute, after which an alive check of the gateway is sent. A response or lack thereof is detected within 200 ms (2 hops). In the case of dynamic rerouting, the time it takes to reroute is dependent on the size of routing tables. In the worst case, the size of a routing table may be as big as 1 KB (as the routing table simply maps next hops and graphs are rather small, the size of this routing is also small). When something needs to be rerouted, it will take 500 ms to ping the server. The server will update its graphs, which will take at most 5 seconds (the time to search for graphs, rerun shortest path algorithms, and write to local storage). Propagating this over the network will then take roughly 500 ms as 1 KB is well withing the bandwidth of even the most bottlenecked connection. In total, dynamic rerouting will cost:

$$500 \text{ ms} + 5 \text{ s} + 500 \text{ ms} = 6 \text{ s}$$

The time for updates depends on the size of the binary. With an estimated update binary size of 15MB, and an average of 1 BLE, 3 BLE+ and a Gateway in between, the time it takes for an update is limited by the 2 Mbit/sec bandwidth of the BLE repeater. Given half of bandwidth in each direction, the bottleneck allows for 15 MB to be transferred roughly 120 seconds. Since updates are done for, say all video cameras, the data transmitted will be the same for all devices and thus will not require extra bandwidth as the number of devices of each kind increases.

7.3 Usability

Smart Campus has devised a storage scheme that makes it trivial for Facilities to query data. As all of our information is stored in relational databases, accessing information of varying complexity (from simple selects to more complicated filtering) is as complicated as executing lines of SQL is. In the case where facilities also desires an interface for accessing data, we can preserve cost efficiency by hiring a UROP to develop an interface that overlays the SQL commands. Fault tolerance, however, would not be guaranteed for this interface.

7.4 Outlook

As Facilities upgrades the infrastructure, higher requirements may be met. If the desired frame rate from cameras were increased, upgrades to camera buffer sizes to handle gateway failures would be needed to preserve fault tolerance. In addition, repeater bandwidth could be improved to adjust for the increased network traffic being sent in both normal and crisis modes. Another requirement of giving Facilities access to last minute of data before the start of crisis mode would only tax the network as cameras already store their past frames in their buffer comfortably. Thus, improved network bandwidth would be all that is required to send this additional data.

7.5 Hardware Components

With an average of BLE repeater per thermostat/motion detector pair in an area, a conservative estimate would be 16,000 BLE repeaters. With an estimate of 3 BLE repeaters funneled towards one BLE+ pair, and another BLE+ pair hop in between those and the gateway, the estimate of number is $\frac{4}{3} \cdot 16000 = 21,300$ BLE+ repeaters. At an average of 400 smart devices on each gateway there will be around 75 gateways. These components are placed in a branching pattern as described in section 4.2 and thus can handle buildings with unconventional layouts such as the Stata building as depicted in figure 5.

The cost of this design is the sum of the costs of the BLE/BLE+ repeaters and gateways.

$$16000 \cdot \$10 + 21300 \cdot \$50 + 75 \cdot \$500 \approx \$1.26 \text{ million}$$

7.6 Security

In order to restrict access to the system to Facilities staff members, a username and password system would be implemented. There would also be a hierarchical distinction where only Facilities managers will be able to delete data gathered, while general staff will have access to all other features of the system necessary for them to maintain it.

Due to computational constraints, smart devices cannot encrypt data. This is not problematic since thermostat and motion detector data do not contain sensitive information. Video cameras have more sensitive data, however due to their placement in generally very public areas or secure areas with little traffic, it is unlikely that this data would require encryption.

8 Conclusion

With a large campus, adding infrastructure to create an affordable, fault-tolerant, and scalable system requires careful design to maximize performance while minimizing costs. Smart Campus seeks to create a reliable network by making use of the different strengths of the various network devices given while minimizing the need for more expensive devices. Smart Campus puts into place devices

such as repeaters that provide increased throughput in normal conditions and redundant backup during device failures to ensure that the system always runs smoothly. This architecture allows for normal operation during the variety of situations it may encounter thus meeting Smart Campus' design goals. Overall, Smart Campus is a reliable, cost-efficient system that will allow for MIT to not only maintain a better infrastructure, but also to better serve its populace.

9 Author Contributions

Sabina Chen created the diagrams and caption used in this paper, as well as explained the various user modes and cases and their associated complexities. Joseph Torres created and designed the main aspects of the network architecture and FCS. Justin Yu created the bookends to the paper, as well as defined the system's interactions with smart devices and performed the evaluation. Every author contributed greatly to the editing of the paper.

10 Acknowledgements

We would like to thank our writing instructors Michael Trice and Michael Schandorf for helping us learn how to communicate technical content to various audiences. We would also like to thank Katie Sedlar for guiding us through the design project. Lastly, we owe a great thank you to Tim Kraska, our recitation instructor who helped us shape and fit our system design to where it is today.

References

[1] MIT. (n.d.). The Campus. Retrieved March 23, 2018, from `http://web.mit.edu/facts/campus.html`

[2] Cloudflare. (n.d.). What is Anycast? How does Anycast Work? Retrieved March 23, 2018, from `https://www.cloudflare.com/learning/cdn/glossary/anycast-network/`

[3] Indiana University. (2018, January 18). What are cron and crontab, and how do I use them? Retrieved March 23, 2018, from `https://kb.iu.edu/d/afiz`

[4] Associated Press. (2005, November 4). MIT maps wireless users across campus. Retrieved May 6, 2018, from `https://web.archive.org/web/20060905081952/http://senseable.mit.edu/news/on_us/CNN4November2005.htm`

[5] Branching factor. (2018, April 07). Retrieved from `https://en.wikipedia.org/wiki/Branching_factor`

[6] Stanford University. (2009, April 7). Hierarchical agglomerative clustering. Retrieved May 7, 2018, from `https://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html`

[7] Ruzzo, L. (2004). Graph Algorithms: BFS, DFS, and Articulation Points. Retrieved May 7, 2018, from `https://courses.cs.washington.edu/courses/cse421/04su/slides/artic.pdf`

[8] MIT. (2012, April 12). Index of /stata/floorplans. Retrieved May 7, 2018, from https://projects.csail.mit.edu/stata/floorplans/