IntelliNet: A Campus-Wide Internet of Things

Anelise Newman, Christopher Wang, Eric Luu, Giancarlo Delfin

March 23, 2018

Contents

1 Introduction

The MIT campus contains many devices such as thermostats, cameras, and motion detectors that monitor and control the state of MIT's buildings. Currently, these devices lack persistent storage and the ability to intelligently respond to real-time events. MIT Facilities wants to modernize this system by creating a network of "smart" devices connected to a central server (the FCS) that is capable of aggregating data, issuing commands, and responding to campus crises. This network will allow Facilities to automatically adjust the temperature of vacant rooms, collect crowd data for space allocation projects, detect the failure of individual devices, and perform other intelligent tasks.

To meet these goals, our proposed system collects video, temperature, and motion data generated by the smart devices through a network of Bluetooth nodes and Internet-compatible gateways. Our modular network design combined with a hierarchical routing protocol provides a simple and robust system for routing data between devices and the FCS. Upon receiving this data, the FCS can analyze it, dispatch appropriate instructions to smart devices, and reliably store it for future analysis. The sections that follow will specify our proposed network topology, communication protocol, and FCS procedures, as well as how they achieve our main design goals of simplicity, performance, fault tolerance, and scalability.

2 Network Topology

We propose a modular, campus-agnostic network topology that is both simple and extensible. It divides the campus into hierarchical domains that provide the basis for our routing protocol.

The basic unit of our network is a $42' \times 42' \times 42'$ cube. In the middle of the cube are one or more BLE+ repeaters. We choose a side length of 42 feet so that every point in the cube is within 30 feet of the repeater, which is the maximum range for a smart device (in the absence of walls). We place enough BLE+ repeaters in a cube to maintain connections with all the devices that fall within the cube. Each device connects to a single "parent" BLE+ repeater.

These cubes are composed into 3D blocks of variable dimensions each containing between 20 and 30 BLE+ repeaters. Each block contains two gateways. These blocks can be arranged to cover the entire campus.

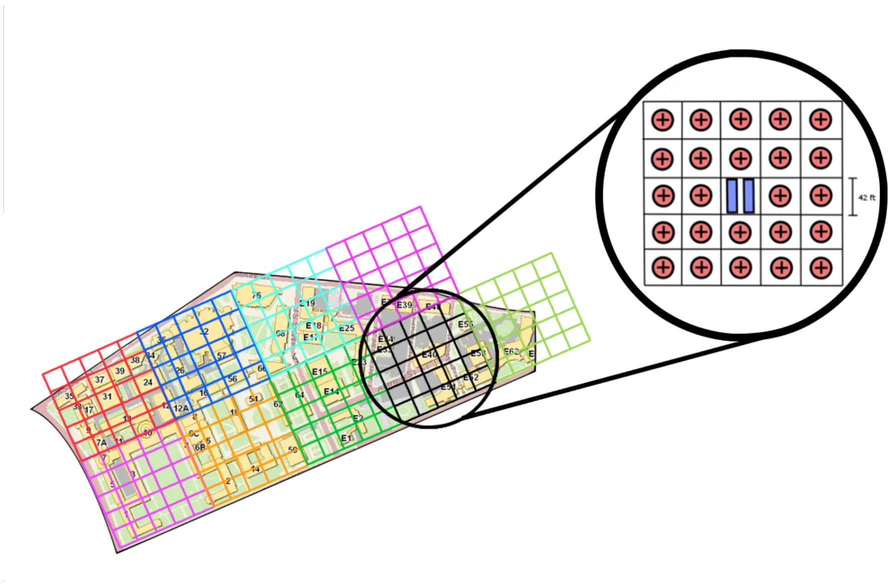

Figure 1: *An example of a flat block consisting of 25 cubes arranged in a* $5 \times 5 \times 1$ *grid. Note that there is also a BLE+ repeater in the center square, which has been omitted for clarity.*

Walls limit the range of smart devices and may put them out of range of their parent BLE+ repeater. To resolve this problem, we also place chains of normal BLE repeaters, where required, to extend the range of smart devices.

Our topology makes no assumptions about the layout of MIT's buildings. It ensures network coverage across campus and can be extended to future buildings or other institutions.

3 Network Protocols

Our hierarchical network protocol provides fault tolerance and performance while maintaining simplicity.

3.1 Naming

Each device in our network has a unique 48 bit ID. The first 10 bits specify which block the device is in. The next bit specifies one of the two gateways in the block. The higher bits designate the parent BLE+ repeater and the device itself (Figure 2). If the device is a gateway or a BLE+ repeater, all higher bits corresponding to lower levels in the hierarchy are set to 0.

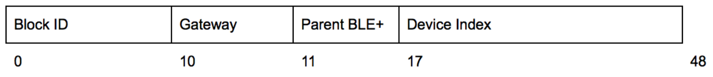

Figure 2: *ID breakdown*

3.2 Network Protocols and Routing

Packets in our network are routed in a hierarchical manner that reflects the structure of our network.

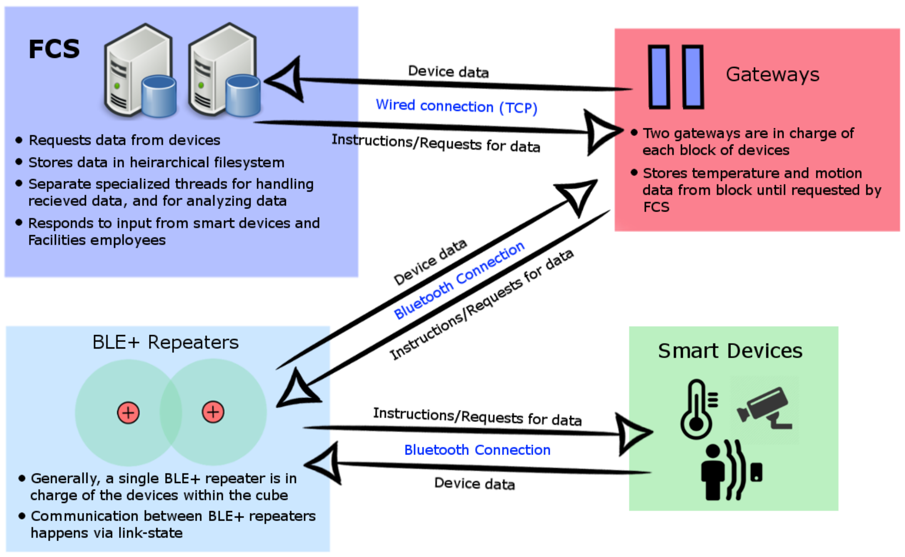
FCS
• Requests data from devices
• Stores data in heirarchical filesystem
• Separate specialized threads for handling recieved data, and for analyzing data
• Responds to input from smart devices and Facilities employees
Device data
Wired connection (TCP)
Instructions/Requests for data
Gateways
• Two gateways are in charge of each block of devices
• Stores temperature and motion data from block until requested by FCS
Device data
Bluetooth Connection
Instructions/Requests for data
BLE+ Repeaters
• Generally, a single BLE+ repeater is in charge of the devices within the cube
• Communication between BLE+ repeaters happens via link-state
Instructions/Requests for data
Bluetooth Connection
Device data
Smart Devices

Figure 3: *Our system diagram. Requests for data are sent from the FCS to one of two parent gateways per block, where they travel through the grid of BLE+ repeaters to get to the device. Device data follows a similar path to the FCS, which analyses and aggregates the data and responds with device-specific instructions.*

3.2.1 Hierarchical Routing

The path a packet takes to get between a smart device and the FCS is illustrated in Figure 3. Packets from a device to the FCS are routed first to a device's parent BLE+ repeater and then to one of the two gateways in its block. If a device is in range of its parent BLE+ repeater, it sends its packets directly; otherwise, its packets pass through a chain of BLE repeaters first. Once packets reach the BLE+ parent, they are routed through a mesh of link-state-connected BLE+ nodes until they reach one of the two parent gateways which can contact the FCS. The process for getting from the FCS to a device is symmetric.

3.2.2 Network Layer Routing

A smart device communicates only with its parent BLE+ repeater or any BLE repeaters that are in range. Within each block, gateways and BLE+ repeaters transfer packets between each other using a link-state routing protocol. Gateways communicate with the FCS via the Internet. There is no communication between devices in different blocks.

The system balances simplicity and reliability. Although link-state routing adds complexity, it is only used to route between the 20-30 BLE+ repeaters and two gateways within a given block. Within this limited scope, the flood overhead required by link-state routing will be a constant factor, and will not scale with the size of the overall system. The flexible routing between BLE+ repeaters means that alternate paths can be found if a BLE+ repeater goes down.

In cases where BLE repeaters are necessary to extend range, these repeaters will accept connections from any device and broadcast packets that they receive to anyone in their range. This protocol is simple but may generate some extra congestion. However, since the repeaters have a very limited range (30 feet) and are only used in cases where range is already limited, we do not think this will cause significant slowdown in the network as a whole.

The network layer protocol is best effort. Since repeaters and devices are inherently unreliable, we rely on the transport layer to enforce reliable communication.

3.2.3 Announcements

Periodically (every second), devices broadcast their presence to all neighbors within range. Devices only accept connections from their parents, their children, and range-extending BLE repeaters.

3.3 Transport Protocols (TCP)

Our transport layer uses TCP to ensure that messages are delivered reliably in spite of unreliable hardware. Our packet headers will include the source and destination ID of the message, as well as a packet ID (to be used for sending ACKs).

3.4 Temporary Data Storage

Data accumulates in our network until explicitly requested by the FCS. This allows the FCS to load balance and manage congestion and allows for controlled data accumulation if the FCS goes down temporarily.

Temperature and motion data are stored on the two gateways in each block. As soon as smart devices generate data, they send it to one of their parent gateways. Video data is stored on the camera itself. Gateways and cameras wait for an FCS request to upload their saved data.

3.5 Fault tolerance

Our design is resilient to gateway failure. If a gateway fails, its sibling in the same block will take over its load until it is repaired. In the absence of a gateway failure, packets will be routed by randomly setting the "gateway bit" in the destination address to 0 or 1. If a problem is detected in one of the gateways, all packets are routed through its sibling.

Regarding BLE+ repeater failure, our design balances resiliency and simplicity. If a BLE+ repeater fails, connection will be lost to its immediate children, but all other data will be sent to the FCS as normal because link-state routing allows for finding alternate paths from any live BLE+ repeater to the gateway. Since BLE+ repeaters take only 5 minutes to repair, we feel that the simplicity gained in terms of naming, routing, and topology from having one parent per device justifies losing connection to the children for this short window.

4 Data processing

As depicted in our system diagram (Figure 3), the FCS has three main jobs for data collection: requesting data from smart devices, processing incoming data, and responding to input from smart devices and Facilities employees. The FCS's processes are designed to be as modular and fault-tolerant as possible.

4.1 Data Requests

Our system of request-driven data collection allows the FCS to intelligently coordinate data flow through the network to reduce latency and congestion, leading to a more performant system.

4.2 Temperature and motion data.

The FCS requests temperature and motion data from gateways every two minutes, well within the 5-minute interval requested by Facilities. Requests to the various gateways are staggered throughout the 2-minute window to balance the load on the FCS. When a gateway receives a request from the FCS, it sends the data in its backlog, starting with the most recent data, up to a threshold. This same procedure applies to both crisis and normal mode.

4.3 Camera data

By default, the FCS will collect camera data from one camera per block at a time and will cycle between the cameras in a block. This prevents excessive camera data from overwhelming the network. In normal mode, the FCS requests all data in the camera's backlog.

In crisis mode, the FCS requests data from only the crisis cameras in each block and only requests real-time data, meaning that old data frames are kept in the backlog. Given that the non-crisis cameras can buffer up to 40 hours of data at one frame per second, delaying collection from them should not cause data loss in most circumstances.

4.4 Data Ingestion

All data enters the FCS via the same FIFO queue. Thus, it is necessary to split up the incoming data by type and pass it to the appropriate handler.

The FCS's main thread spawns child threads that pop data from the input queue and distribute it to several specialized queues, each with their own workers. We designate special queues for temperature data, motion data, video data from non-crisis cameras, video data from crisis cameras, and update confirmations. Each queue's handlers take care of calling the appropriate methods to store the data for subsequent analysis. In addition to storing the collected data, these handlers call a method that updates the timestamp of the most recently received data for the smart device, it's BLE+ parent, and the gateway it routed through.

This design is robust to failure because even if one thread fails, having multiple workers allows data ingestion to continue until the main thread can restart it. It is scalable because the amount of workers can be increased to handle throughput, and if multiple processors are required, the modularity of having specialized queues means that work can be split between processors.

4.5 Data Processing

Certain information needs to be handled by the FCS in real-time: changing thermostat temperatures in response to motion data, responding to crises or update requests, and keeping track of failures. Special threads are assigned to each task.

Threads for data analysis are separated from threads for data ingestion: they read from data storage as opposed to directly from the input FIFO queue. Thus a failure of the analysis module will not disrupt data ingestion. Also, this division of labor means that these two tasks can be run on different servers if Facilities decides to scale in the future.

4.5.1 Thermostat updates

Thermostat temperatures must be adjusted depending on whether there has been motion in the room in the past two hours or not. Every five minutes, a thread is awakened that determines if temperature adjustments are needed and sends temperature-update commands to appropriate thermostats.

4.5.2 Responding to a crisis

Handlers that read from the crisis video data queue wait on the event `start_crisis_processing` to begin operation. The event `enable_crisis(camera_ID)` has two effects: `camera_ID` is added to a global set storing the ids of cameras in crisis, and the event `start_crisis_processing` is emitted. `disable_crisis(camera_ID)` results in `camera_ID` being removed from the set of cameras in crisis. Once all cameras have been removed from this set and the crisis queue has been cleared out, crisis queue handlers will resume waiting.

4.5.3 Responding to an update

On the event `update(device_type, software_binary)`, a thread awakens that sends the binary to devices of the correct type. This is done in a staggered way so that only a fraction of the devices within each block receive the update at once. Particularly, the two gateways in each block should not be updated concurrently. The thread returns once all devices have responded with an update confirmation message (processed via the update queue) or a timeout has expired, in which case the devices who did not respond are reported to Facilities as having failed.

4.6 Clearing out data and checking for malfunctions

Once per every five minutes, a process runs that deletes all stale data (video data more than a week old and temperature data more than two weeks old). This thread also checks the timestamp of the last message received from each smart device. If the message is more than ten minutes old for a smart device or 5 minutes old for a gateway/repeater, the device is marked as broken and this list is returned to Facilities for maintenance. If a gateway is broken, this is considered a serious threat to network efficiency, so the FCS slows down data requested from the corresponding block by asking cameras to send their video data at 3 instead of 5 frames per second until the issue is marked as resolved by Facilities.

5 Data Storage

The FCS's data storage unit stores data in the file system and makes use of backups to protect data.

5.1 Data Organization

The FCS will store its data in a hierarchical directory structure. The top level has three directories: `Temperature`, `Motion`, and `Video`. Each device category is separated out by room number and then device ID. Data received from smart devices is stored as `.txt` files. This simple arrangement allows Facilities to access any piece of data by traversing three directories.

5.2 Data Backups

The FCS has 100 TB of storage to store a week's worth of camera footage, two weeks' worth of thermostat data, and various motion detector timestamps. Its hard drive will be divided into two partitions: one with 90 TB for normal data and one with 10 TB for backups. The 90 TB partition will store a week's worth of video data recorded at five frames per second. It will also store two weeks' worth of thermostat data and motion timestamps, which together require less than 0.00025 TB of storage. The 10 TB partition will store camera footage compressed using the HEVC standard, which reduces size by a factor of up to 1000, along with duplicates of thermostat and motion detector readings, since their storage requirements are minimal.

6 Design Goals

6.1 Simplicity

Our network topology is made of composable blocks that can be independently reasoned about and easily extended. Our block-based, hierarchical routing protocol lets us leverage the flexibility of link-state routing while limiting its complexity. Finally, modularization of functions on the FCS means that each thread only has to worry about a specific task instead of coordinating with a host of other processes.

6.2 Performance

Our system was designed to smooth out data flow between the network and the FCS, which will reduce congestion and latency. Furthermore, the FCS is highly parallelized and can quickly process large quantities of data.

6.3 Fault Tolerance

Our network handles gateway failures and limits the damage from BLE+ failures. We limit the amount of device data stored in the network itself to prevent loss from device failures, while maintaining the capacity to buffer some data if the FCS goes down. Finally, the modularization of the FCS means that the death of one thread does not interrupt the others.

6.4 Scalability

Both our network layout and our processing pipeline scale easily. Our network topology is campus-agnostic and easily extensible and the modularity of the FCS means that it would be easy to divide its functionality between multiple machines if more processing power were required.

7 Conclusion

Our proposed system fulfills the three main goals of MIT Facilities: collecting data for future project planning, reducing temperatures of inactive rooms to save energy, and automatically detecting infrastructure failures to facilitate faster repairs. Our system ensures reliable communication of commands and requests for data, even when individual machines fail. Our system also has mechanisms to prevent data loss if the FCS fails. Designed with simplicity, performance, fault tolerance, and scalability in mind, our system will provide the infrastructure needed to improve the quality of campus life.