Title

Hamilton: Flexible, Open Source $10 Wireless Sensor System for Energy Efficient Building Operation

Authors

California Institute for Energy and Environment (CIEE)
Andersen, Michael
Culler, David
et al.

Publication Date

2021

Hamilton: Flexible, Open Source $10 Wireless Sensor System for Energy Efficient Building Operation

Final Project Report
Review Draft June 28, 2021

Prepared for
U.S. Department of Energy's Office of
Energy Efficiency and Renewable Energy (EERE)

Prepared by
Michael Andersen
Prof. David Culler
Hyung-Sin Kim
Prof. Raluca Popa
Gabe Fierro
Sam Kumar
Moustafa AbdelBaky
Jack Kolb
Andrew Krioukov
Therese Peffer

In coordination with
University of California, Berkeley

2150 Allston Way, Suite 280
Berkeley, CA 94704-110
(510) 664-7462 | uc-ciee.org

Final Technical Report

Award Number:	DE-EE0007685
Recipient:	Regents of the University of California California Institute for Energy and Environment)
Project Title:	Hamilton: Flexible, Open Source $10 Wireless Sensor System for Energy Efficient Building Operation
Principle Investigator:	Carl Blumstein
Project team members:	Michael Andersen, Prof. David Culler, Hyung-Sin Kim, Prof. Raluca Popa, Gabe Fierro, Sam Kumar, Moustafa AbdelBaky, Jack Kolb, Andrew Krioukov, Therese Peffer
Report Submission Date:	28 June 2021
Reporting Period:	1 October 2016- 30 March 2021

ACKNOWLEDGEMENT:

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Building Technologies Office (BTO) Award Number DE-EE0007685.

Table of Contents

Executive Summary

Sensors for improving building performance are rapidly populating the market, driven in part by the drive to reduce greenhouse gas emissions resulting from energy production as well as improve the interior environment for healthy and more productive spaces[1]. UC Berkeley has led wireless sensor development over the past 25 years (e.g., Telos mote[2]). The Hamilton sensor was designed as a low-cost (named after Alexander Hamilton on the US $10 bill) high-performance sensor that is modular and interoperable. The objective of the Hamilton project was to create, evaluate and establish the technological foundations for secure and easy to deploy building energy efficiency applications utilizing pervasive, low-cost wireless sensors integrated with traditional Building Management Systems (BMS), consumer-sector building components, and powerful data analytics—and to demonstrate the effectiveness of this foundation on potential applications.

The project included iterative hardware design, incorporating a high-performance database (BTrDb, http://btrdb.io/), creating and iterating the development of secure data middleware (BOSSwave, WAVE/WAVEMQ), working with and pushing the development of an open-source tiny operating system RiotOS, and implementing and improving protocols such as Thread/OpenThread and TCP/IP. The hardware benefited from careful design to drive down the cost; the design included a System-on-a-Chip (SoC), chip antenna, single crystal and only five passive components. Careful design of the operating system created a low-power design to enable a long life with small batteries. The hardware included several sensors: temperature, radiant temperature (90degree cone), relative humidity, magnetometer, accelerometer, and light, with an optional occupancy (Passive InfraRed) sensor. The project was the basis of several applications, both internal to the research team and other researchers and professionals at other institutions. Several applications used the sensor hardware as the basis for other complex devices, such as an innovative low-cost ultrasonic anemometer (Arens et al., 2020), the prototype of a residential thermostat for the low-income sector (Alston-Stepnitz et al., 2020), personal comfort system desk fan and foot-warmer, and an occupant-counting device, Panasonic Grid-EYE Infrared Array Sensor[3]. Other applications used the sensors to improve building performance through interoperating with the building Heating Ventilation and Air-Conditioning (HVAC) system, such as using occupancy and/or distributed temperature sensing to reduce HVAC zone energy while still providing thermal comfort and to reduce peak loads in small commercial buildings. We developed WAVE drivers for three thermostats, two lighting systems, two plug load monitors, two electric vehicle chargers, an energy monitor, utility interval meter gateway, a photovoltaic system, and several weather stations. We demonstrated cloud-based energy analytics, implemented a schedule and a Model Predictive Controller in a small commercial building to optimize HVAC energy, occupancy and electricity price. This system has

---

[1] https://www.ledsmagazine.com/smart-lighting-iot/article/14180024/smart-building-sensors-provide-the-foundation-for-the-decade-of-data-magazine

[2] https://people.eecs.berkeley.edu/~culler/papers/spots05-telos.pdf

[3] http://www.andes.ucmerced.edu/research/occupancy.html

been deployed in multiple test beds: Soda Hall, Jacobs Hall, Sutardja Dai Hall, test sites for the California Energy Commission (CEC)-funded EPIC fans project, CIEE office in downtown Berkeley, as well as residential locations. Initial integration of these technological innovations was performed through the creation of execution containers containing the WAVE agent and various driver, proxy, or building system function logic.

The research added to the understanding of efficient sensor hardware, secure middleware, time-series data management (high performance database), efficient communication protocols, and interoperating with applications and building systems. The project funded several graduate students and post-doctoral scholars who produced 14 journal papers and conferences presentations based on this work. A startup company HamiltonIOT (https://hamiltoniot.com/) sold sensors and data aggregation and management services. The database BTrDb was released under an open-source license; Ping Things (https://www.pingthings.io/) commercialized BTrDb and is using it in their business with electrical utility monitoring and analysis. The researchers contributed to RiotOS and OpenThread. One researcher presented in a panel on security at the Energy Exchange Building Summit in Cleveland, OH in late August 2018. The WAVE and WAVEMQ secure middleware was used in multiple projects: the CEC-funded eXtensible Building Operating System (XBOS) project (https://docs.xbos.io/), FlexHP (LBNL-led project, funded by CEC) and OpenBOS-NY (TRC-led project, funded by NYSERDA). Another opensource tool supporting WAVE is JEDI (Joining Encryption and Delegation for IoT): end to end encryption for WAVE (github.com/ucbrise/jedi-protocol-go).

The project showed the technical effectiveness and economic feasibility of creating a low-cost, modular, and easy-to-deploy sensor. Through conversations with multiple end users, the research team discovered that many customers wanted data management and services in addition to the sensors. HamiltonIOT developed packages of sensors, border router, and data services to provide a seamless “plug-and-play” sensor deployment. Some customers were willing to pay for higher quality sensors (such as light); some customers wanted a robust enclosure that was waterproof or animal-proof.

Goals and Objectives

The objective of the project is to create, evaluate and establish the technological foundations for secure and easy to deploy building energy efficiency applications utilizing pervasive, low-cost wireless sensors integrated with traditional Building Management Systems (BMS), consumer-sector building components, personalized smartphone devices, and powerful data analytics—and to demonstrate the effectiveness of this foundation on potential applications.

The project actually created and evaluated these technological foundations—hardware, secure middleware, database—and integrated them with BMS and other components, evaluated with data analytics and demonstrated through applications. The project did not integrate with personalized smartphone devices.

Project Activities

The project was divided into three budget years with four tasks in each period: 1) low-cost wireless sensor system, 2) secure attack-resistant middleware tier, 3) application, components and services tier, and 4) technology to market plan. This section of the technical report discusses each budget period in turn

Budget Period 1

In the first budget period, the team developed the first version of the sensor (called Hamilton, after the person on the $10 bill), developed a secure information bus, and integrated the sensors into building systems to enable application.

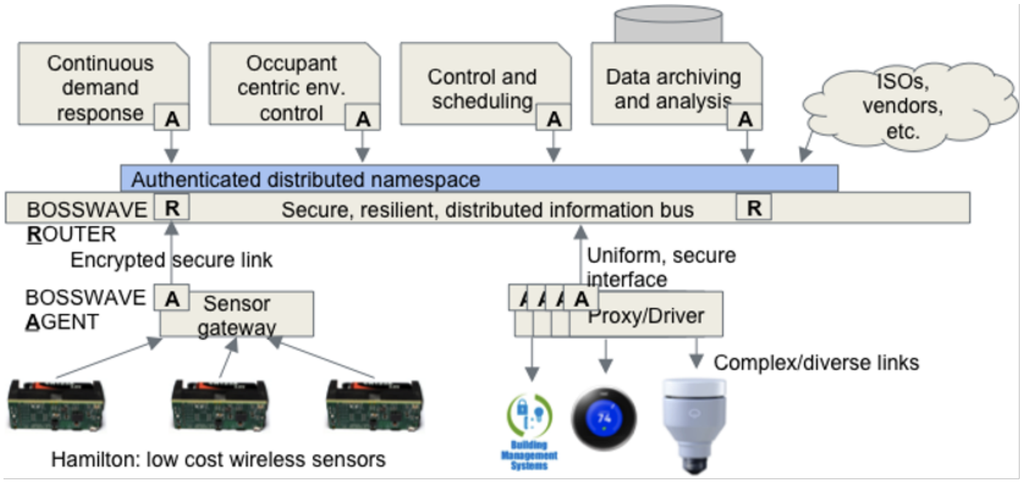


Figure 1: The initial vision of Hamilton sensors integrated into a secure building intelligence and automation system.

The first task, *Low Cost Wireless Sensor Systems*, sought to develop complete, long-lived networked sensors to provide situational awareness for optimization of environmental conditioning, lighting and other building functions at a total cost of manufacturing of less than $10 per sensor. Task 2, *Secure Attack-Resistant Middleware Tier* aimed to provide a resilient, secure, fully distributed information bus, structured into decentralized namespaces supporting a hierarchy of resources with authenticated communication and verified authorization on each action. The third task, *Applications, Components and Services* tier, intended to integrate these technological advances into a complete building-wide secure system utilizing pervasive sensing and fine grain authorization to spur innovation in building energy efficiency.

The first-generation hardware platform was designed and manufactured in sufficient volume to provide a sharp assessment of manufacturability, utility, cost, and cost trade-offs. An entirely US manufacturing process was utilized. We had manufactured a small run (100) of the preliminary (Hamilton 7) design, for which we had an overseas quote in large volume of $9.80 each; in low volume domestic production was $20 to permit early assessment of the Go/No-Go gap. Hamilton V.1 was designed and manufactured in quantity of 100 providing a sensor pack substantially beyond the $10 price point. Two models (described below) were manufactured in low quantity (60 V.1-3c and 40 V.1-7c with Passive InfraRed (PIR) sensor allowing a sensor design flaw to be detected. A design refinement was carried out and 1,000 Hamilton V2s (680

V2-3C and 320 V2-7C) were manufactured using an in-situ full test methodology, demonstrating high yield.

The Building Operating System Services Wide Area Verified Exchange (BOSSwave, or WAVE) was developed and implemented as Router and Agent software components that run on distributed computational resources local or remote to the building.

A Border Router was developed and produced to aggregate the data from multiple sensors. A total of 150 were manufactured in two batches, including a $34 variant of Hamilton, a Raspberry Pi, and robust enclosure.

This system was deployed in multiple test beds near the UC Berkeley campus: Soda Hall, Jacobs Hall, and the CIEE office in downtown Berkeley, as well as residential locations. Initial integration of these technological innovations was performed through the creation of execution containers containing the WAVE agent and various driver, proxy, or building system function logic (Figure 1 above). In the CIEE office, the Hamilton sensors were integrated with the building control systems.

Task 1 Low Cost Wireless Sensor System

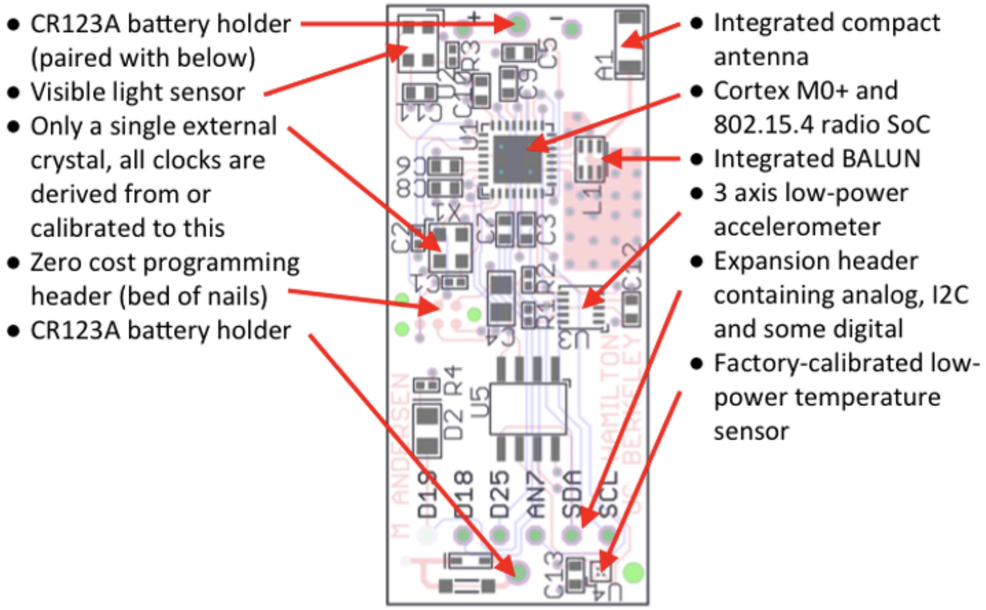
Sensor Hardware

This section describes the hardware design of Hamilton and embedded network design.

The goals for the low-cost sensor Hamilton platform are:

- Make the core mote cheaper by reducing assembly costs and leveraging integration
- Eliminate hidden costs:
  - Standalone, no carrier required
  - Power supply part of mote
  - Solve the enclosure problem (Enclosures can often cost as much as the mote, or more)
  - Mote has useful sensors by itself, doesn’t (always) require costly extension board
  - Mote includes antenna on board

Figure 2 provides a detailed description of the components of the preliminary version of Hamilton.



- CR123A battery holder (paired with below)
- Visible light sensor
- Only a single external crystal, all clocks are derived from or calibrated to this
- Zero cost programming header (bed of nails)
- CR123A battery holder
- Integrated compact antenna
- Cortex M0+ and 802.15.4 radio SoC
- Integrated BALUN
- 3 axis low-power accelerometer
- Expansion header containing analog, I2C and some digital
- Factory-calibrated low-power temperature sensor

Figure 2: Components of the first version of Hamilton, the 7.

Figure 3: Photo of Hamilton 7 with typical AA battery, at cost of $9.80.

Over the course of this period, the Hamilton 7 was improved to the current Hamilton V2 3C and 7C versions. The form factor and battery type was kept, but the board was extended by seven millimeters to enable easier programming (with the addition of a Button and LED (part of commissioning / key exchange)) and thermal isolation for the temperature sensors.

Figure 4 shows a schematic of Hamilton (3C/7C) and Figure 5 shows component details.

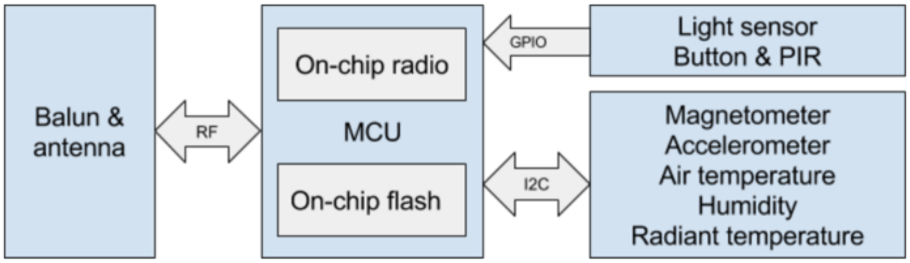


Figure 4: Schematic of Hamilton

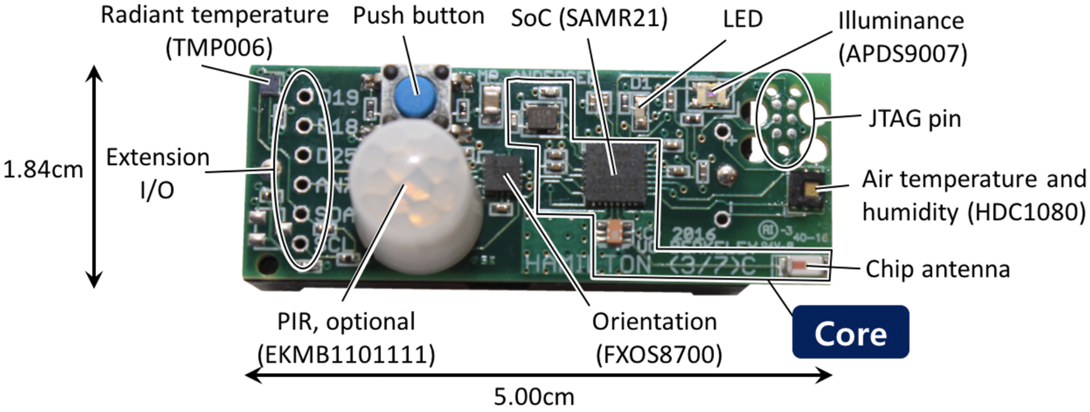


Figure 5: The second version of Hamilton: the (3/7)C with the push button, optional PIR sensor (7C), and separation for the air temperature sensor.

The programming connector has several large holes that allow the clip to hold itself to the board without requiring a component to be soldered on; that would increase cost. These holes, along with the requirement that the back of the board is clear behind the connector were leveraged to also improve the accuracy of the air temperature sensor that was moved between the programming connector and the antenna. This part of the board does not contain copper on the inside layers of the board, which is a requirement for the antenna but also prevents thermal coupling from the rest of the board onto the air temperature sensor. The drastic decrease in thermal mass surrounding the air temperature sensor, along with the change of the sensor to a higher precision device increased the air temperature accuracy and response time.

This version has the following sensors:

- High precision Temperature and Relative Humidity
- Radiant temperature
- High precision Accelerometer and Magnetometer
- Light sensor

The Hamilton 7C adds a Passive InfraRed (PIR) sensor to proxy occupancy detection. The estimated cost is $16 for the 3C at quantity and $36 for the 7C, with US manufacturer (our cost was slightly higher due to a last-minute rush order on a vendor part).

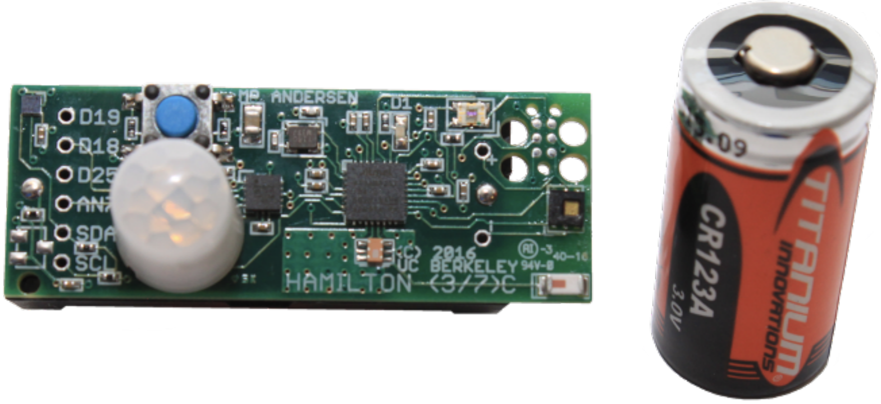


Figure 6: Photograph of the second version of Hamilton, the (3-7)C board; the 7C is shown with PIR sensor.

A border router was developed based on a single board computer, the Raspberry Pi 3. This device aggregates data from multiple Hamilton sensors via 802.15.4 and connects to the Internet. The features of the border router are:

- Stateless (can be power-cycled randomly)
- Rugged enclosure (can be kicked)
- Tamper detection (accelerometer)
- Wired Ethernet upstream
- All configuration conducted via USB flash drive (including network details like MAC address, IP, DHCP, hostname)
- Secure - e.g., no password-based SSH
- Total Cost ~ $70

Figure 7: The border router based on a Raspberry Pi 3 miniature computer plus Hamilton-derived 802.15.4 interface in robust enclosure.

The Hamilton sensor was demonstrated at BuildSys 2017 (Andersen, Kim, et al., 2017); the system architecture is described in (H.-S. Kim et al., 2018).

Sensor Software

In the interests of encouraging a wider adoption of the Hamilton platform, and in creating a software infrastructure for it that can be sustained by the community, we concluded that extending an already active and popular operating system would be preferable to creating a brand new operating system and attempting to create a community around it. As such, we evaluated TinyOS, RIOT OS and Contiki as possible candidates for extension.

While Berkeley originated TinyOS and supported a worldwide open source community for many years, the project and its open source community has fallen behind since responsibility for it shifted to Stanford in 2007. The TinyOS architecture is sound and extremely well suited to our goals. However, it is now difficult to get significant code contributions approved and merged (many pull requests are months old) and the alternative of creating an independent fork and clean the code base along with recreating the community is a large effort with risk. TinyOS also does not have support for the Atmel SAMR21 that Hamilton uses, and several important components have not been as actively developed in recent years.

Contiki is an active project with an active community that has existed for many years, but as a result has inherited a lot of technical debt in the code. In addition, there was no existing platform support for the Atmel SAMR21.

A relatively new operating system, RIOT-OS was evaluated, as it already had platform support for the MCU, has an active community (nearly twice the code activity of Contiki, and ten times that of TinyOS over 2016). The core developers (that approve code contributions) have done a good job of keeping the code clean, despite RIOT-OS's rapid increase in popularity, yet they are fast and responsive to code contributions. Ultimately this was our selection.

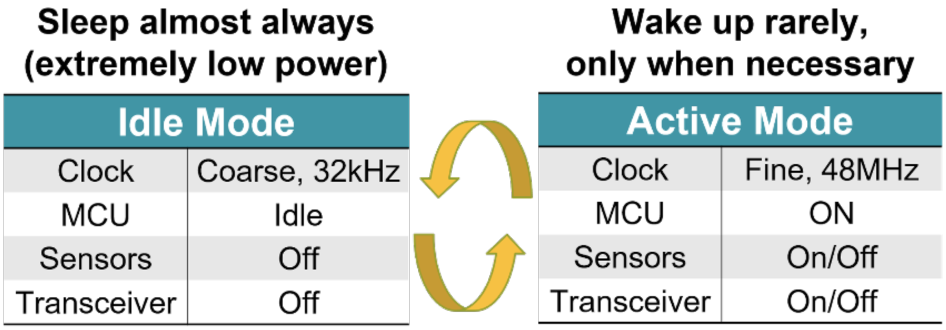
We developed an initial port of RIOT_OS for the Hamilton. Measurements of the power consumption, however, showed that the operating system was not properly using the low power features of the MCU. By working with the community, several critical changes were made in the operating system. By the end of Budget Period 1, some of these were in upstream RIOT-OS and some of these were maintained in the Hamilton-specific fork pending future upstreaming:

- Add support for low power modes on SAMD21 #2309 (we contributed testing)
- Support for system timers based on arbitrary precision oscillators #5608 (we contributed testing and bugfixes)
- Support for stdio over JTAG without a UART #5851 (our code, merged upstream)
- Support for Hamilton sensors (our code, upstream pending merge)
- Support for Hamilton platform (our code, not upstreamed yet)
- Fix for large I2C transfers#5832 #5833 (our bugfix, merged upstream)
- Add support for Hamilton's MCU #4719 (our code, merged upstream)
- Fix SPI power consumption #5868 (our code, merged upstream)
- Fixes to neighour discovery #4968 (our bugfix, merged upstream)
- Fixes to JTAG #4733 (our bugfix, merged upstream)
- Fixes to router advertisements #5006 (our bugfix, merged upstream)

Sensor Network Design

Within the embedded operating system described above, we developed an initial low-power wireless communications stack based on a listen-after-send duty-cycle protocol. This, and all the code for subtask 1.2, is available at https://github.com/hamilton-mote/RIOT-OS.

There are three main requirements when developing a robust and efficient communication infrastructure: 1) A node should be idle for most of the time (i.e., low duty cycle) 2) When a node is idle it should consume extremely low power 3) When a node is active, its operation is robust enough to support reliable packet delivery. To this end, we first define each mode operation: which Hamilton hardware components should be used or not for what operation mode, as depicted below. In brief, in the idle mode, we turn off all peripherals and use a coarse, slow (32 kHz) and low power oscillator. Whereas in the active mode, we allow any peripheral to be turned on and use a fine, fast (48 MHz) and high-power oscillator.



Sleep almost always (extremely low power)

Idle Mode	
Clock	Coarse, 32kHz
MCU	Idle
Sensors	Off
Transceiver	Off

Wake up rarely, only when necessary

Active Mode	
Clock	Fine, 48MHz
MCU	ON
Sensors	On/Off
Transceiver	On/Off

Figure 8: Design of two Hamilton operation modes: idle and active

Next, we tested if the idle mode was really idle. Our initial evaluation of Hamilton on RIOT-OS showed that the idle mode consumes more than 700 uA, which is much higher than our target (and design expectations from the datasheets). After analyzing RIOT-OS kernel, we found out that low power management functionality was not well-designed. Low power management of an entire platform, such as Hamilton incorporating sensors, radio, and MCU, is more complicated than that of a single hardware component. We implemented low power management functionality for Hamilton on RIOT-OS, which provides less than 9 uA of idle current consumption for all types of Hamilton motes. This is similar to the idle current consumption of TelosB (a representative 16-bit low power platform) on TinyOS (a representative low power OS) and lower than that of TelosB on Contiki. Our groundwork verified that the combination of Hamilton and RIOT-OS, which provides faster processing speed, more memory space, more sensors, and lower cost than TelosB/TinyOS, had the potential in terms of low power consumption.

Our next focus was to design an efficient active mode. Given that the active period governs Hamilton lifetime, we aimed to minimize the active period by thoroughly investigating RIOT-OS kernel and removing any redundant operation. We found that the established timer implementation of RIOT-OS was not efficient enough to support both the fast active mode and the low power idle mode. Specifically, it wasted considerable amount of time during the mode transition (i.e., when waking up and falling asleep). This is because a timer for a mode transition needs to be fed by the low-power but slow oscillator. To minimize that latency, we enabled the timer to use both fast and slow oscillators in a collaborative manner during the mode transition. **Figures below** show how the timer improvement impacts the active period. When a Hamilton wakes up and sleeps immediately, our improvement reduced the active period from 2.0 ms to 0.84 ms. When a Hamilton wakes up, sends a short packet, and sleeps again, the active period was reduced from 5.17 ms to 3.37 ms. Given that this contributes to lowering duty cycle, we see this as essential to enabling a robust, efficient communication infrastructure.

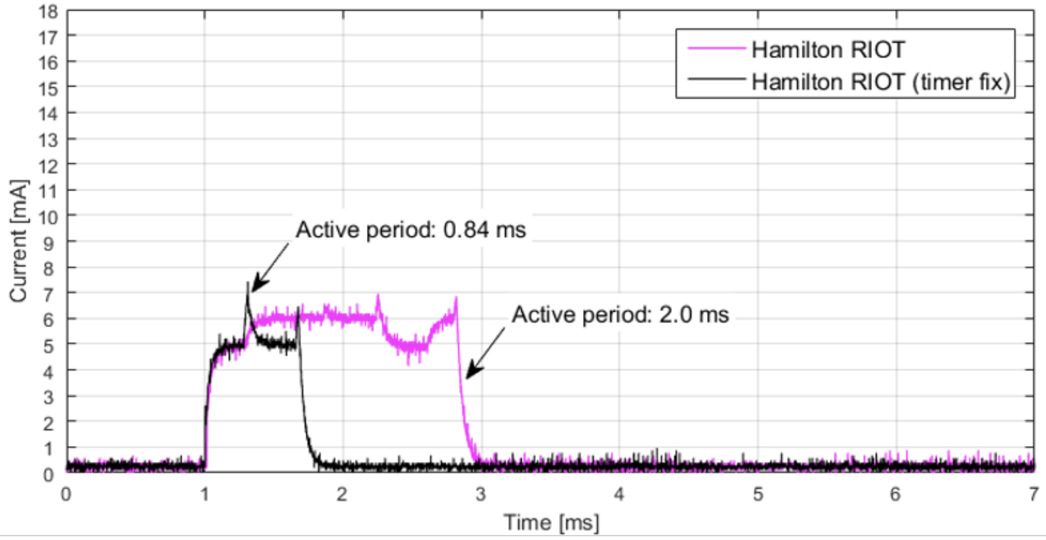


Figure 9: Impact of timer improvement on a CPU wake-up (1.16 ms reduction of active period)

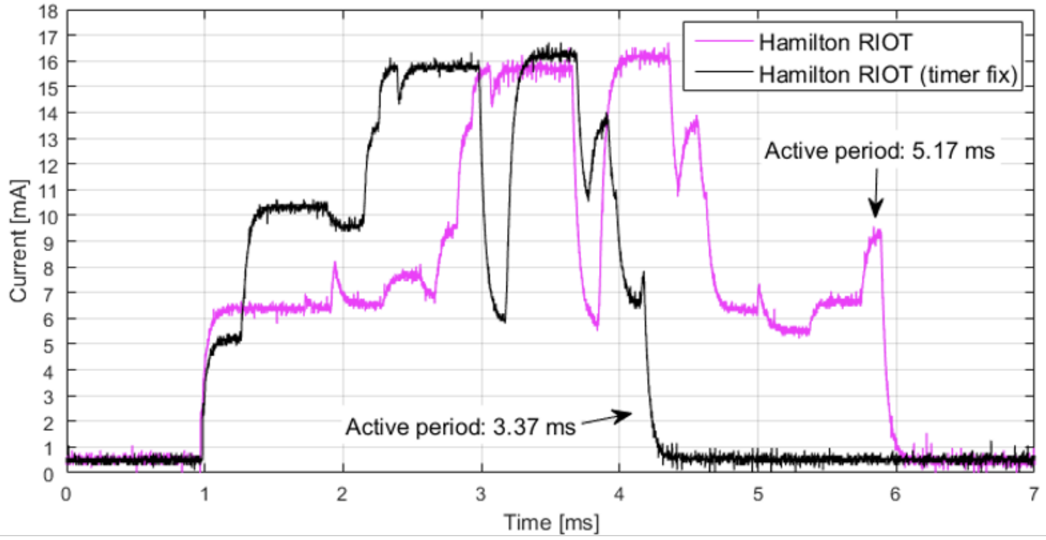
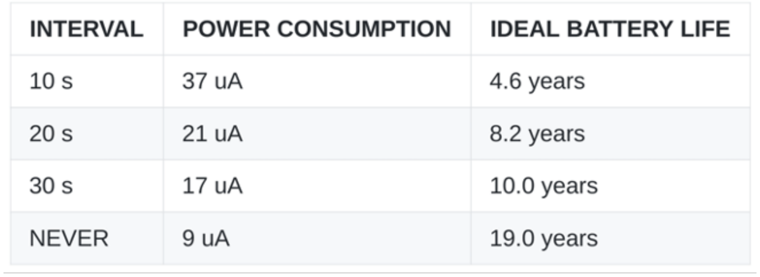


Figure 10: Impact of timer improvement on a radio wake-up (1.8 ms reduction of active period)

A full report on the power consumption of the low-power communication stack is at https://github.com/hamilton-mote/hamilton-3c-qualified-fw with summary here.

Using the controlled-source power measurement technique, the power consumption of the Hamilton-7C has been determined that sends a message once per interval, for various interval durations. The table below provides a summary of average power and expected lifetime for a range of configurations.



INTERVAL	POWER CONSUMPTION	IDEAL BATTERY LIFE
10 s	37 uA	4.6 years
20 s	21 uA	8.2 years
30 s	17 uA	10.0 years
NEVER	9 uA	19.0 years

Table 1: Power consumption and battery life at various data intervals

The configuration that we deploy most frequently is to send data every 20 seconds, so the average power consumption is 21uA, well below the threshold identified as the Go/No-Go target. The NEVER interval is the idle power of the system including the leakage of all of the sensors populated on the board.

The Hamilton 3C (without PIR) is strictly more power efficient than the Hamilton-7C (with PIR).

Our deployments used a topology in which the low-power sensor nodes are not responsible for network management or routing; they communicate to a Border Router using a newly developed low-power simplex protocol. A typical example is shown in the figure below. This is a cluster of star topologies that may overlap.

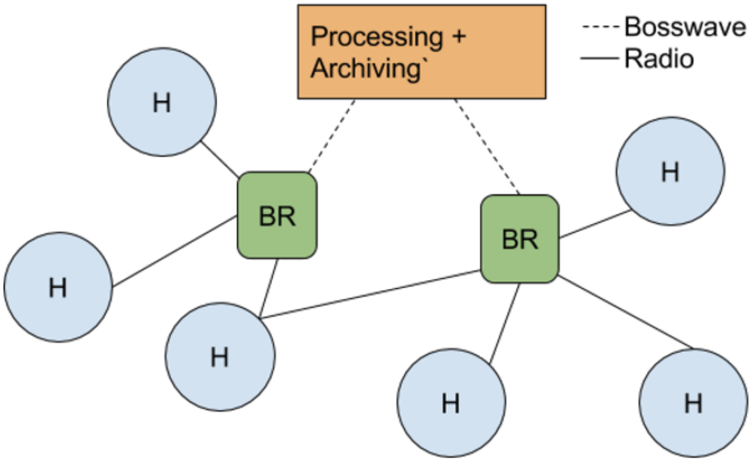


Figure 11: Typical overlapping connectivity of Hamilton low-power network

There are deployment advantages to having intermediate powered low-cost nodes between the Hamiltons and the border routers, extending the range of connectivity beyond border routers. The figure below depicts this network architecture. Nodes are battery-powered or plugged-in. Nodes can be plugged-in if there is a power source nearby and more so in indoor environments which have abundant power sources. Using plugged-in nodes efficiently can decrease energy consumption on battery powered nodes. Thread, a recently standardized IPv6 low power network protocol, adopts this concept and decouples battery powered nodes from route management. Google Nest opened the whole implementation of Thread, called OpenThread, on https://github.com/openthread/openthread and updates it. We used OpenThread as the network protocol for Hamilton.

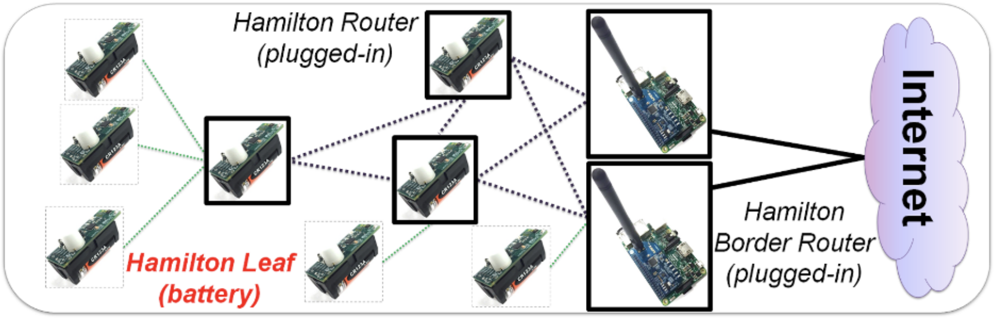


Figure 12: Hamilton network architecture. Only plugged-in nodes manage routing.

To support the Thread network, we designed a duty cycling protocol as depicted in Figure 13, enabling bi-directional communication between a plugged-in (always-on) node and a battery-powered node. When the duty cycling nodes has a packet to send to the always-on node (i.e., an upward packet), it immediately wakes up and sends to the always-on node. When the always-on node has a packet to send to the duty cycling node (i.e., a downward packet), it has to know when the duty cycling node wakes up to send the packet. To this end, the duty cycling node periodically wakes up and sends a beacon to the always-on node to notify its wake-up. Note that the beacon is only for receiving downward packets and the beacon interval is decoupled from upward packet interval. In most applications, data traffic is expected to flow mostly upwards for state monitoring, which enables to use a very long beacon period for a long lifetime.

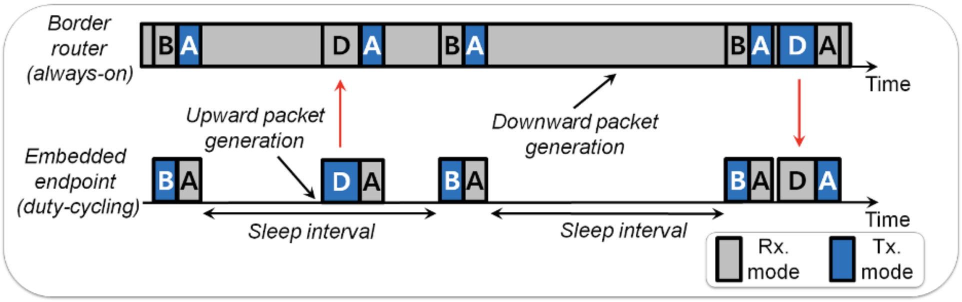


Figure 13: An example of listen-after-send-based duty cycling operation.

Milestone 1.1 Measure idle and active power consumption of a prototype implementation of Generation 1 and determine gap to Go/No-Go target.

The idle current of the full Hamilton 7C design, including sensors and operating system in their quiescent state, is under 9 uA, or 19.8 uW at 2.2v, 29.7 uW at 3.3v.

Active current varies depending what particular sub-circuits are enabled. For example, in sending or receiving a packet it reaches 17 mA (shown above) and with also sensing and computing it can be nearly twice that.

To measure current accurately over its four orders of magnitude variation, we developed a novel, high dynamic-range measurement apparatus, described below.

Milestone 1.2 Embedded OS schedules 10 millisecond (ms) task once per second at average current of 300 microAmp (uA)

Using the methodology described below (see Budget Period 1 Significant Findings below), we verified the milestone 1.2 target of 10ms/s active time (or 1% duty cycle) at below 300uA average. We constructed a Hamilton firmware code image that spins at full CPU utilization for 10ms and then sleeps for 990ms so that this 10ms pulse is a 0.1% duty cycle. Verification of the behavior of each phase is observed directly in isolation, but simply inferring average behavior from the independent measurements leaves too large an error, as the as the idle is so small when active is registered and spike is so narrow when idle is measurable.

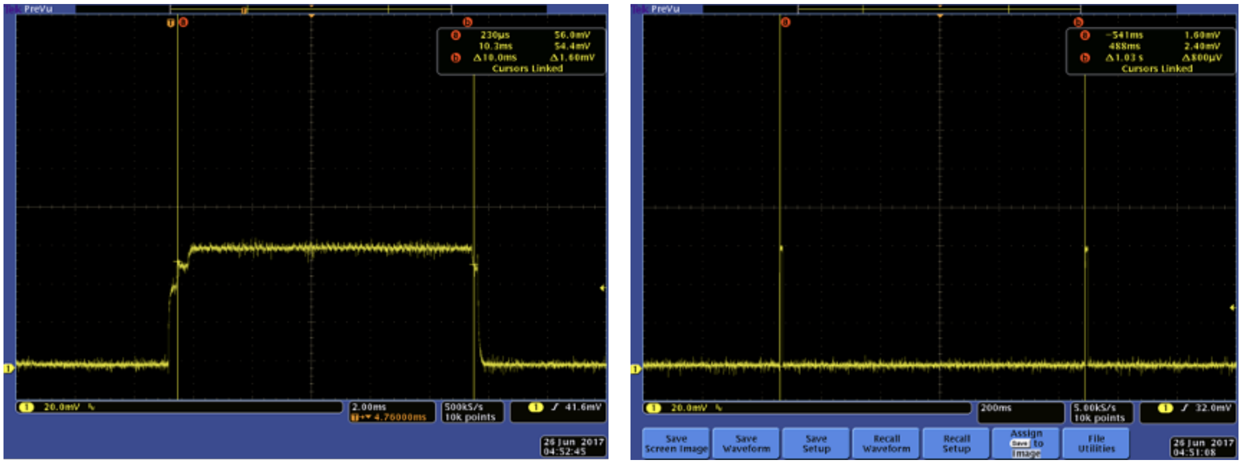


Figure 14: Direct measurement of the active mode (left) and idle mode (right).

To determine empirically the combined behavior, we connected this Hamilton to the power measurement jig and set the threshold to 300uA – verifying the milestone. We observed the following waveform:

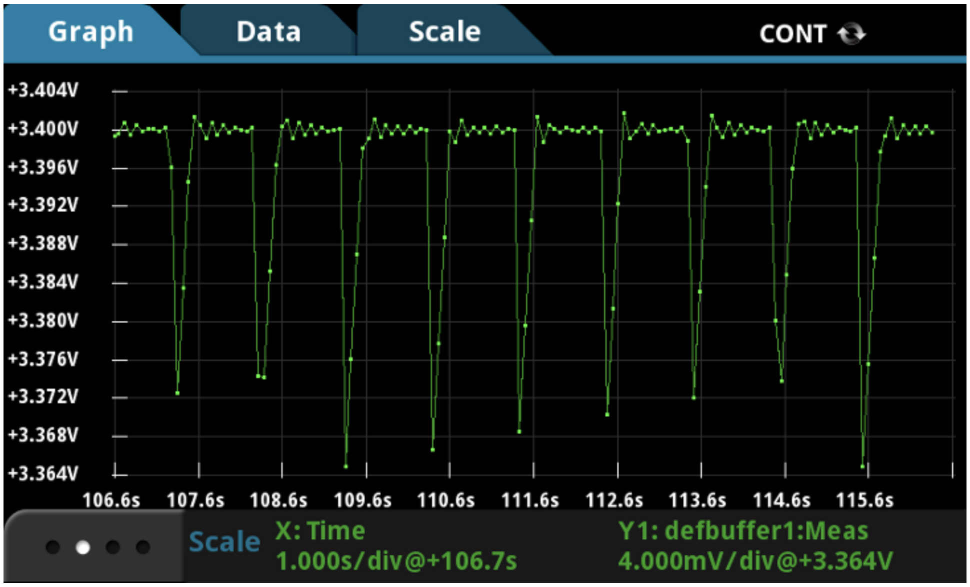


Figure 15: Current consumption search at 300uA target

The long flat periods indicate that we are operating well below the threshold. While this is sufficient to meet the milestone, to find the actual average current we conducted a search process, testing against a 100uA threshold and observed the following:

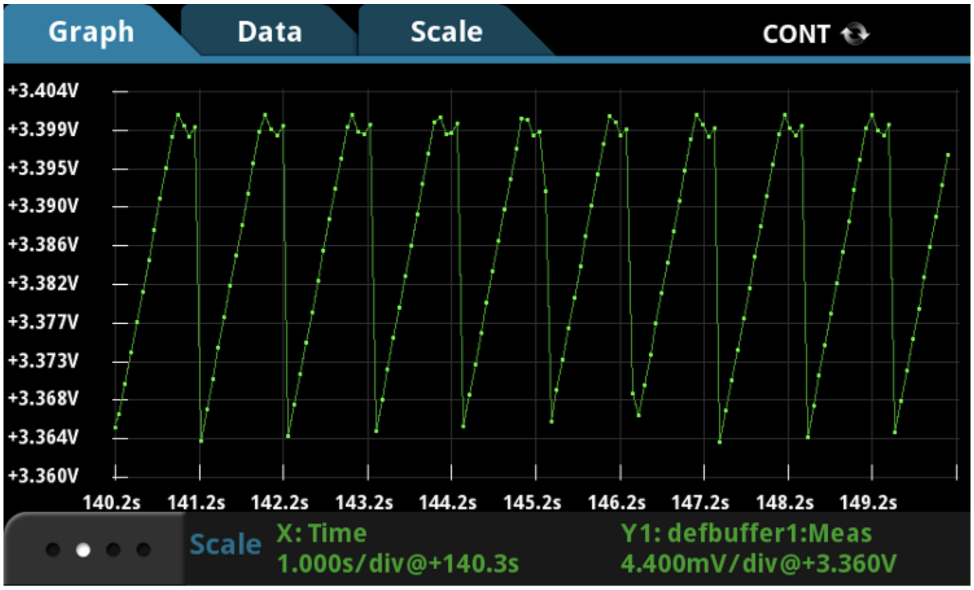


Figure 16: Current consumption search at 100uA target

The sharper points indicate that we are operating closer to the test threshold of 100uA. Testing against a threshold of 70uA we get:

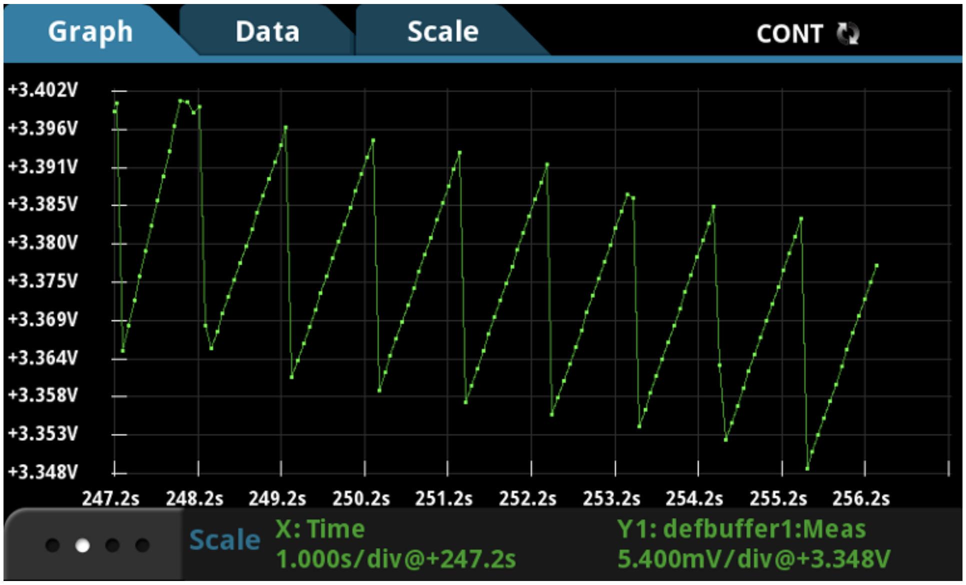


Figure 17: Current consumption search at 70 uA target

The sharp points and downward trend over time indicate that we are operating above the 70uA threshold.

We can therefore conclude that we have met Milestone 1.2: we have a Hamilton where the OS is scheduling a 10 millisecond task once per second with 70uA < average current < 100uA, well below the 300uA target.

Task 2: Secure Attack-Resistant Middleware Tier

This section describes the design and proof of concept of the BOSSwave (now called WAVE) router and agent, Spawnpoint container, and comfort application interface.

Secure information bus: BOSSwave

A full code release and documentation of BOSSwave is available at https://github.com/immesys/bw2. Its design and evaluation are presented in depth in a paper submitted to ACM Symposium on Operating Systems Principles and in one submitted to 4th ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys 2017). The basic structure of the router and agent software is shown below.

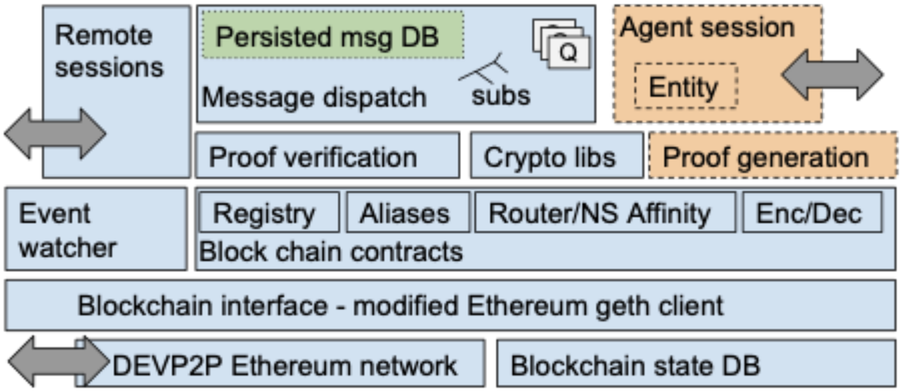


Figure 18: Layering of Software components for Secure Attack-Resistant Middleware

BOSSwave is implemented in Go and consists of two daemons: a **router** which typically runs on a server-class platform with a persistent internet connection, and an **agent** which runs on every platform using BOSSwave resources such as IoT devices, servers, laptops, phones etc. The authorization tier is implemented as four smart contracts on an Ethereum blockchain: the Registry contract stores the Delegations of Trust, as well as the public parts of Entities (such as their expiry date). DoTs and Entities are revoked by storing the revocation in the contract. The Alias contract stores a mapping from a sequence of human readable characters (e.g., "alice19"") to a 32-byte blob, usually the public key of an Entity in the registry. The contract ensures these aliases are globally unique and immutable. The Affinity contract stores the binding between a namespace and the designated router entity, as well as the binding from the designated router entity to a server's IP address and port. The Encoding/Decoding contract is a library used by the other three contracts that can validate Entity, DoT and revocation objects by checking they are well formed and the signature is valid.

To interact with BOSSwave, an application uses a language binding that connects to the agent. As the cryptographic libraries are embedded in the agent, the language bindings are trivial to generate. To allow the agent to perform BOSSwave operations on behalf of the application, the entity object is transferred by the application to the agent upon startup. The application can then send high level commands like "publish this payload to this resource" and the agent takes care of generating the proof, serializing the message, signing it, resolving and verifying the designated router and transferring the proof-carrying message. Similarly for subscription, the agent verifies every message and only forwards valid messages to the application. As a result of this architecture, applications or services using \wave\ are no more complex than those using legacy syndication such as MQTT or XMPP.

Note that there is no blockchain server. The blockchain exists without any server, authority or coordinator by consensus among the body of clients, all of whom are equal. So while, for example, the registry contract code is acting as an authority, it is autonomous and cannot be tampered with. This is trivializing a highly complex and fascinating system, please consult (Wood, 2014) for more information. An in depth presentation of the theory and realization of the cryptography and the Ethereum microcontracts is presented in (Andersen, Kolb, Chen, Fierro, et al., 2017).

The WAVE router and agent are designed symmetrically, so that while they are distinct roles, they utilize the same code paths.

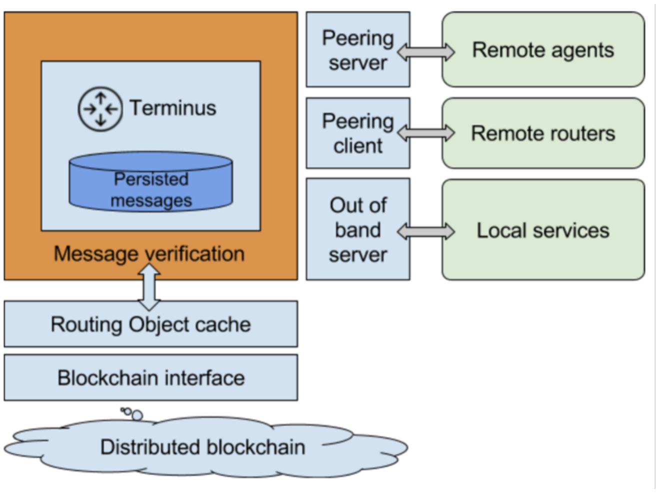


Figure 19: WAVE router and agent schematic.

The software is constructed in blocks with a clear interface between them to facilitate code auditing. The job of routing a given message to the set of matching subscriptions based on URI pattern matching is done by the Terminus block. All messages and subscriptions are assumed to be authorized and valid inside this block. All messages flowing into and out of this block flow through the message verification layer that will drop invalid messages. The messages come from or go to *adapters* on the right-hand side.

For messages going to a BOSSwave router from a service, the message will flow in through the out of band server adapter, through the message verification layer and terminus then into a peering client adapter to the remote router.

For messages going to a BOSSwave agent through a router, they will flow in from an agent through the peering server adapter, through verification and terminus and back out to subscribed agents through the peering server again.

Persisted messages (like metadata) are stored in a database in Terminus and are re-verified as they flow back out to clients, in case their permissions have been expired or revoked since they flowed into terminus and into the database.

Spawnpoint Container

The Spawnpoint system properly integrates with both Docker and WAVE and runs isolated microservices according to manifests published to it over BOSSwave.

Spawnpoint is a ground-up solution to the central challenges of embedded service lifecycle:

(1) How does a service obtain permissions to consume its input

resources and produce its output resources?

(2) How does a service obtain its initial configuration?

(3) How does a service get scheduled to run?

(4) How is the service isolated such that failure does not couple to colocated but unrelated services?

(5) How is a service monitored and how is authority to monitor it obtained?

(6) How is a service retired?

Spawnpoint's purpose is to deploy, run, and monitor the microservices, which together form an instance of the eXtensible Building Operating System (XBOS)(Gabe Fierro, 2021), within managed execution containers. Thus, Spawnpoint can be thought of as the "proto-service" combining Docker and BOSSwave upon which all other microservices are built. It begins with the realization that the resources required for persistent processes (e.g., CPU time) are the same as any other resources (e.g., sensor data) and can be managed using the same WAVE primitives. The ability to deploy and manipulate microservices is tied to the ability to publish commands on certain BOSSwave URIs, while the visibility of execution containers, as well as the computational resources that back them, is contingent on the ability to subscribe to messages on certain BOSSwave URIs. Spawnpoint enforces isolation between running services and also performs admission control to ensure that resources do not become oversubscribed.

We have the following active Spawnpoint namespaces:

- ucberkeley: used for UC Berkeley's Amplab and Center for the Built Environment (CBE) deployments
- mask: used for a residential deployment of XBOS
- garber: used for a residential deployment of XBOS
- scratch: used for testing and demos

The development of Spawnpoint is complete. Almost all of the active Hamilton deployments have their backend services deployed in Spawnpoint, processing tens of thousands of messages per hour. Code and installation guide can be found here: https://github.com/immesys/spawnpoint

Commercial building comfort application

We have conducted initial design discussions with Building Robotics on the adaptation of portions of the commercial COMFY product line to the XBOS API. We have also had design discussion with BuildingOS and examined the adaptation of their API to the open API. We have adapted the WAVE streaming API to the commercially and openly available influxDB (https://www.influxdata.com/) and thereby have built a variety of analytics dashboards with Grafana Labs open-source software for time series analytics over Hamilton deployments described below.

Milestone 2.1: Measure throughput of Router software component and determine gap to Go/No-Go

We conducted a functional implementation of the WAVE Router and Agent. We demonstrated the capacity to route the equivalent of 100,000 sensors at a 5-minute average reporting rate, which is 333 messages per second. Our tests showed a capacity that surpassed this rate, at approximately 1600 msgs/s on a single route. We expect this to grow with multiple producers and consumers. We have observed the average transaction propagation: Once a permission change is in the blockchain (which may take tens of seconds), a new transaction is observed in under 5 seconds when using our one router. We have tested seven routers operating concurrently. Our testing used 10,000 Entities successfully; we have executed permissions with 100,000 resources. We successfully showed the system's creation and maintenance of 1,000,000 delegations of trust (DOTs).

We conducted an emulation of a city-scale WAVE deployment involving nearly a million people and over a hundred thousand buildings using behaviors drawn from a statistical model. It issues commands to agents, which act on them as if they were real commands, putting the created entities and delegations of trust into the global state. Separately, 100 containers running in the cloud emulate participants in WAVE. To capture the effects of differing internet connections, the netem feature of Linux is used to emulate a spectrum of internet speeds and latencies. Bandwidth usage, CPU usage and memory usage statistics are streamed from each container to a central database for analysis.

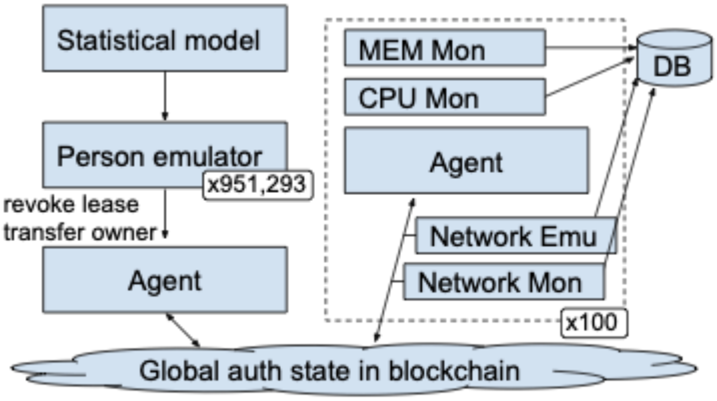
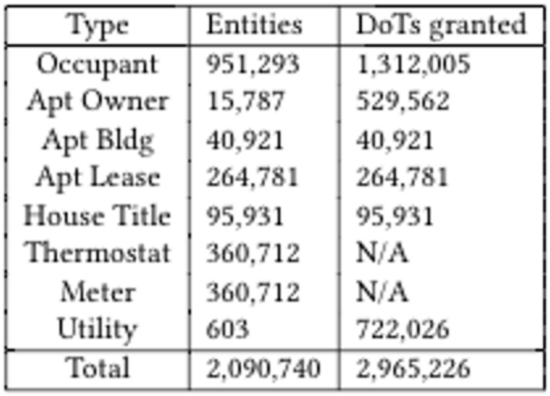


Figure 20: Emulation of WAVE router and Agent.

To set up the emulation, nearly a million distinct entities are created for people living in a city. Then an additional million entities are created for leases (and titles), apartment buildings, apartment owners, and common devices such as thermostats and meters as shown below. These intermediaries are created as distinct entities to capture the real hierarchy and delegation present in a city. An apartment leasee, for example, obtains permissions to the thermostat resources using a proof traversing their lease, apartment building and then building owner.



Type	Entities	DoTs granted
Occupant	951,293	1,312,005
Apt Owner	15,787	529,562
Apt Bldg	40,921	40,921
Apt Lease	264,781	264,781
House Title	95,931	95,931
Thermostat	360,712	N/A
Meter	360,712	N/A
Utility	603	722,026
Total	2,090,740	2,965,226

Table 2: Entities in the WAVE emulation.

At the peak of business hours, the city emulation causes roughly 150 changes of permission per hour. We separately characterize the capacity of BOSSwave at a rate of approximately 4000 changes of permission per hour, although this limit is driven by the blockchain and changes as the body of clients vote on what the bandwidth of the chain should be. By using the 100 instrumented nodes streaming the CPU, network and memory cost of the BOSSwave agent, we obtain a model of the cost of participation as the utilization of the global system changes. The most important takeaway is that the average cost of participation is reasonable: a constant CPU load of about 5% of a desktop-class processor core or 50% of an embedded-class ARM core. There is also an associated constant network bandwidth use of between 2KB/s and 12KB/s. For microservices deployed under a Spawnpoint instance, this cost is paid once and all the microservices share an agent (under different isolated sessions).

Task 3: Applications, Components and Services

This section describes the Building Application Components and Services: the design and implementation of secure general purpose scheduler component and secure controller, implementation of secure interface to a trust-limited high-performance time-series data management system, and design and implementation of zonal occupancy awareness application, fusing BMS data and sensor data.

Scheduler

Initial development of a domain specific language (DSL) and execution environment for simple schedulers and control loops is complete. The system's DSL is an implementation of the Lua embedded programming language with added support for BOSSwave interactions and task scheduling. The system so far supports the PUBLISH, PERSIST, SUBSCRIBE and QUERY operations for BOSSwave.

The DSL also contains operations for creating schedules. Schedules can be defined as periodic (e.g. "8am every weekday", "every day at 5:15pm", "every hour") or once-off (e.g. "at 11/30/2017 00:00:00").

We have used the system to implement the following programs, among others:

1. A simple schedule that uses PUBLISH to change thermostat setpoints over WAVE according to a weekday/weekend/holiday schedule

2. Creating an ad-hoc wireless light switch by binding the output of a Hamilton button to the state of a BOSSwave light

3. A traffic monitor that sends an alarm on a configured URI when the publish rate on a WAVE resource exceeds a threshold.

Beneath the DSL, the system provides an event-based execution environment.

The system execution environment is a single executable that can execute programs from local files, an interactive prompt (useful for prototyping), or BOSSwave URIs. Executing a program on a BOSSwave URI takes advantage of the BOSSwave permission model to only execute code published by trusted entities.

The execution environment runs in a Spawnpoint container and interacts with BOSSwave through a local BOSSwave agent.

The code and initial documentation can be found at: https://github.com/gtfierro/bodge

Secure Controller

The DSL above provides an environment for simple control algorithms and has been demonstrated as per 2 and 3 above.

Data management system

We have extended the Berkeley Tree DataBase (BTrDB) to provide two secure interfaces to the database. The first is a lightweight mechanism that permits direct access to the database by computers outside the cluster but verifies usernames and passwords. This runs within the BTrDB cluster, not within Spawnpoint. The data is accessed over a REST API and is very high performance. The security policies that can be expressed by this system match existing database systems and are quite limited (for example you cannot grant access to only a portion of time in a stream). Furthermore. the security policies are disjoint from the already-existing and very descripting permissions in BOSSwave.

To obtain additional security properties, a second system was developed that more tightly integrates BTrDB and BOSSwave. The system can run in a Spawnpoint container and permission to access data in BTrDB is governed by BOSSwave permissions. Access to the database is done over BOSSwave via an agent. This frontend also introduces a mechanism we believe to be novel in timeseries databases: punctuated data access. This extends the permission model from just resources to also include the chronology of permissions. As an example, if an entity is granted BOSSwave permission to a resource today, they can query archived data from the database but will only see data from today. A separate permission can be granted to see specific ranges of historic time, but the system is “secure by default” with respect to data privacy. This system is nearing feature-completion and we are in the performance-optimization stage of development. We developed support for richer metadata expression and querying, aiding in data discovery. https://github.com/gtfierro/PunDat

Occupancy

We have 40 sensors deployed in the Amp lab open office space in Soda Hall. This data, streaming over BOSSwave, is available for applications to consume. One such application, constructed as a prototype to demonstrate spatial presentation of dense sensor deployments, is a predicted occupant comfort map, shown below.

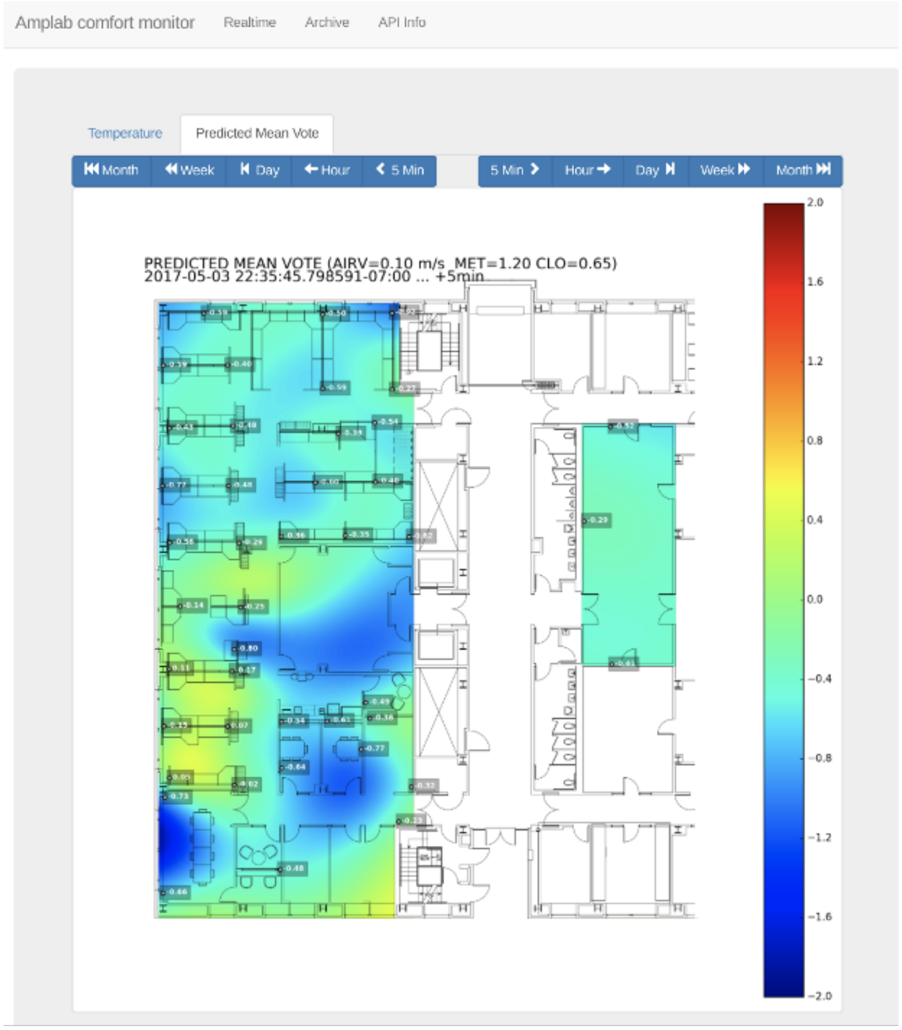


Figure 21: Using Hamilton sensors to produce occupant comfort map.

This occupant comfort is predicted using the Predicted Mean Vote equation, which utilizes the air temperature, humidity along with the estimated metabolism and clothing levels of occupants.

We worked on extending this application to incorporate occupancy representation and determination. We have been working closely with Andrew Krioukov of Building Robotics on new, effective methods of inferring occupancy. A direct approach is to utilize the Hamilton 7C with Passive Infrared to sense motion. Real world experience, however, shows that deploying sensors with the line of sight needed for PIR to work is not always possible. Furthermore, this is a binary result, one cannot easily count the number of occupants with this method, just if there are one or more occupants.

Preliminary results, however, suggest that by using absolute humidity, true occupancy can be predicted, including a fairly accurate estimation of how many people are in the room. This method works best in conference rooms or offices where there is a full partition. It does not appear to work well in open plan offices. There are numerous advantages to this strategy. Humidity is far cheaper to sense (the Hamilton-3C is half the cost of the Hamilton-7C), it consumes far less energy, it can be placed almost anywhere in the room without requiring line of sight, and it can give a count of the number of people.

Due to the promising results we have obtained so far, we have deferred producing the occupancy application (which we anticipated to be PIR based) until we have fully characterized the absolute humidity (or hybrid AH+PIR) occupancy method.

Both through direct BACnet access and, potentially, through interfacing with the Building Robotics BACNet interface we continue to work toward fusing BMS data. Broadening this to non-BMS settings, we have a deployment in CIEE, a 7500 sq ft office space in downtown Berkeley, with networked thermostats connected to roof top HVAC units, lighting, smart meter and plug loads and 16 Hamilton sensors. All devices report their state through BOSSwave, which is saved in BtrDB using Archive Requests. The deployment is a testbed for basic schedulers and controllers implemented over WAVE, and is a pilot site for future integrations with demand response signals and occupant-centered control.

Milestone 3.1: Insert over 333 values per second into time-series data management system.

There are two paths for data to flow into the time series database. One is directly (e.g., from microservices to the database via the BTrDB API) and the other is via Archive Requests.

Direct access has been shown to exceed the milestone by three orders of magnitude on several different setups. For example, we archived data from a city-scale electric grid simulation of 1000 synchrophasor devices, producing an aggregate load of more than a million points per second.

Archive Requests are a more powerful method of inserting data into the time series database and interact with the BOSSwave-aware database frontend. A participant with access to resources on BOSSwave creates a list of the resources they want archived along with some metadata and naming information, and then publishes this to a persistent BOSSwave resource. The archiver then observes this, and subscribes to the resources, renaming them as requested, and inserts the data into BTrDB. This method is extremely powerful because it supports offline-archiving. A participant does not need to be online to insert data into the database (unlike all existing timeseries databases) rather they persist their archive request and can go offline, knowing that the data will be archived on their behalf.

We have not yet performed stress testing to determine the limit of the Archive Request system, but our active deployments of Hamiltons, the Hamilton Anemometer and building operating system microservices, yields a rate of 998 points per second, which is above the milestone of 333 points per second.

Task 4: Technology to Market

Market Discovery

After a couple of presentations to industry in spring and summer (May 4, Center for the Built Environment (CBE) semi-annual Industry Advisory Board Conference, with over 100 industrial partners from leading engineering, architecture, and buildings-related manufacturers; August 17-18, Berkeley Energy Transportation Systems retreat), dozens of companies have expressed interest in Hamilton.

Hamilton IoT started formation in late August 2017 to sell Hamiltons, border routers, provide fulfillment, and provide data services (www.hamiltoniot.com).

Stakeholders include researchers, practitioners/consultants, and manufacturers. The table below reflects a few examples. Architecture firms seek added value in providing services, such as validated modeling. Engineering firms endeavor to improve the performance of HVAC system and controls through realtime sensing. Through this initial outreach we conducted a preliminary survey of needs and infrastructure deployment expectations of members of this segment of the market.

Stakeholder	*Application*	*Notes*
Architecture firm (Perkins and Will, LPA)	Environmental monitoring (assess building envelope performance)	Other sensors have underperformed Other sensors too expensive or difficult to install and obtain the data.
Engineering (Integral Group)	Environmental monitoring (assess equipment and improve system performance)	May want to own the data, use own server
Manufacturer of building components and materials (Sage Glass: windows, Armstrong: materials)	Environmental monitoring and Assessment of components	
Manufacturer of HVAC equipment (Price Industries, Ingersoll Rand)	Add value to product through improved environmental monitoring; Reduce labor costs of commissioning	Potentially incorporate sensors into products

Corporate building portfolios (Genentech)	Improved environmental monitoring on campus buildings	

Table 3: Industry interest in Hamilton applications

We received many inquiries from people wanting to purchase the Hamilton kits (sensors and border router) both in research and industry. These provide a starting point for broader market discovery.

In addition to these engagements, we have started a series of deeper engagements with industry partners representing a spectrum of go-to-market strategies.

- Alsen (http://alseninc.com/) working with founder Dr. Alberto Cerpa who is seeking to develop occupant-centered HVAC products. We assisted them with the design a custom Hamilton containing Panasonic Grid-EYE Infrared Array Sensor (https://industrial.panasonic.com/ww/products/sensors/built-in-sensors/grid-eye) and PIR for occupancy detection. We have consulted in getting boards built and provided 10 Border Routers (which will be reimbursed). They are performing an internal TinyOS port for the customized Hamilton.
- PingThings is supporting the BTrDB backend and using it for syncrophasor pilots with PGE.
- Chirp Micro Systems – we have developed a Hamilton-based ultrasonic anemometer utilizing their MEMS transceiver as part of a separate California Energy Commission project.
- Building Robotics continues to work closely with us in connecting technology out of this project with their evolving COMFY offerings.
- BuildingOS provides dashboards for over thirty thousand buildings. We explored with them low effort inclusion of technology coming out of this project.
- Hamony.ai is collaborating closely with us as they seek to re-segment the building commissioning market using Hamilton technology.
- Intel, through their IoT group, is supporting our efforts around low-cost insertion of our technology into buildings through automated metadata acquisition and continuous lifecycle metadata query processing. We are looking at the use of BlockChain technology.

We are also developing kits that would allow research and pilots to explore potential market segments. Collaborator Prof. Jeong Gil Ko is investigating its use in motion sensing from ICU beds at the Ajou University Hospital Trauma Center.

Cost Model

The development of Hamilton has considered materials and labor in design, including manufacturability and yield. Now with a dozen demonstrations, efforts are focusing on cost-benefit of different features, such as types of sensors, ease of installation, and data needs, such as aggregation, collection, visualization. Through several preliminary deployments, we are developing an understanding of deployment processes and how they can be streamlined to reduce the overall deployment cost profile.

We have the following active Hamilton deployments with reliable real-time data streaming:

Name of building/organization	*Type of application*	*Number of sensors*	*Notes*
Building Robotics	Office building	20	Environmental sensing
MASK home	Residential building	42	Environmental sensing
DEC home	Residential building	20	Environmental sensing
Jacobs Hall	University building, office/light labs	35	Environmental sensing
CIEE	Office building	16	Environmental sensing
Soda Hall	University building, office/light labs	40	Environmental sensing
Brower Hall	University building	10	Environmental sensing
Sutardja Dai Hall	University building	30	Environmental sensing
PR home	Residential building	10	Environmental sensing
Calibration jig	Laboratory	10	Environmental sensing
CBE laboratory	Laboratory	8	Environmental sensing plus basis of prototype air flow sensor (CEC-funded anemometer)
CBE laboratory	Insole warmer	1	Prototype of ARPA-E funded personal control system

Table 4: Hamilton sensors in demonstrations

A kickoff industry consortium meeting was held Feb. 8, 2017. The second industrial consortium meeting was held August 17-18 in Berkeley. Additional in-depth collaborations in Intel Corporation, Huawei, Bosch, Chirp Micro, Nest Labs and QuEST.

We worked on the near-term market discovery which includes outlining the stakeholders (e.g., engineers, architects), buying decisions, value made and context of installation.

Market Validation

Through conversations with multiple stakeholders we have outlined the value proposition in several building market segments. Commercial buildings and hospitals are obvious examples of targeted building market segments; less obvious are research labs that use animals (highly motivated to maintain a safe, secure, and healthy environment.) Corporations with greenhouse gas emission targets continue to pursue reduced energy consumption but maintain comfort. Manufacturers need sensors for validating and improving performance.

Commercial building comfort application Demonstration Site identified

We have deployed several Hamilton networks in two living laboratory buildings at UC Berkeley.

First occupied in 2009, Sutardja Dai Hall (SDH) is a seven floor, 141,000 square foot building on the northeast quadrant of the UC Berkeley campus. The building has state-of-the-art classrooms, conference rooms, café, light laboratories, and an auditorium as well as the Marvell nanofabrication lab. The building runs one of two 600 ton chillers (absorption using steam and centrifugal using electricity), depending on the time of year; 135 Variable Air Volume boxes with reheat supply conditioned air to the office portion of the building. SDH was deliberately built out as a living laboratory with multiple meters and submeters. The building has over 6000 sensing points from the Siemens Apogee Building Automation System (BAS) and Wattstopper lighting system accessible through the simple Monitoring and Actuation Profile (sMAP, developed at UC Berkeley), as well as additional sensors, such as temperature and flow sensors on the chilled water supply, discharge air temperature sensors at most of the 135 zones, and wireless indoor environmental quality sensors. The site can be used for investigations of control applications, thermal comfort, and indoor air quality of energy efficient space conditioning technologies. A separately funded CEC project is developing low cost air flow sensors that could also be installed as a result of this project. The building has been the site of many different research projects (e.g., DIADR—funded by DOE; CTR and XBOS-DR—funded by CEC).

The CIEE offices consist of a single floor of 7500 square feet in a concrete and brick three-story small commercial building built in 1934 in downtown Berkeley. The floor has high ceilings and windows on the north, east, and south facades; the space includes open and private offices, conference room, kitchen, bathrooms and a shower. As a result of a DOE-funded project (OpenBAS) and CEC-funded project (XBOS-DR), the space has been outfitted with a networked interval utility meter capable of real-time energy data at 10s intervals, five networked thermostats controlling packaged roof top HVAC systems, two advanced lighting control systems, plug load controllers, and network of wireless indoor environmental quality sensors. The data is aggregated, tagged and stored in a high performance and secure system that is easily and quickly accessible.

Initial comfort applications can easily be conducted in either building.

Budget Period 1 Significant findings, conclusions, or developments.

Several preliminary **deployments** were performed, proving out the over-all functionality, scalability, and ease of deployment of Hamilton/XBOS networked sensor systems.

Average power measurements were obtained by using a novel power measurement jig. Instead of measuring instantaneous current (as is measured by almost all setups), the jig measures the integral of the current, giving a very accurate average current measurement. This approach also has a much larger dynamic range than shunt-based approaches. The jig is accurate to 100nA while still being capable of providing up to 30mA for short bursts, with no range switching.

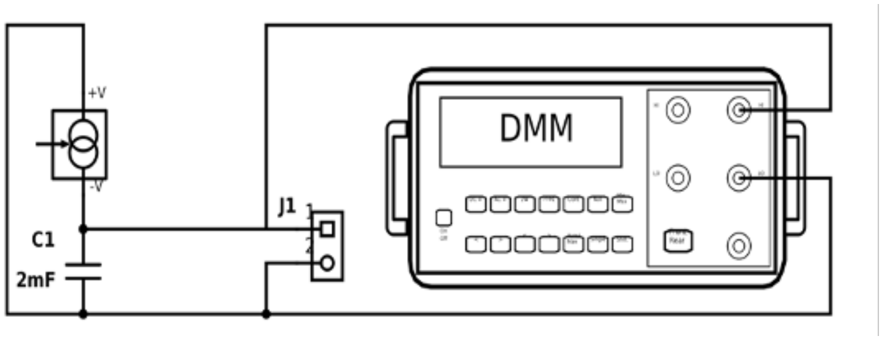


Figure 23: Schematic of measurement jig.

The sources of error in this approach are the current source, the stability of the value of the capacitor (not the value itself), and the leakage current of the capacitor. This approach has typically yielded poor results because most capacitors in the 2mF range either have poor leakage (ceramic) or a non-constant value that varies with past charge levels (electrolytic). We reached out to capacitor vendors and an engineer at Cornel Dubilier mentioned that if one were to use high voltage film capacitors (typically used for power factor correction at hundreds or thousands of volts), then the leakage current at low voltage (3.3V) would be negligible (order nA, much smaller than the uA we are measuring) and film capacitors have no “memory” effect, yielding a very constant capacitance value. With these, the only significant source of error is the current source, and it is possible to buy very accurate (pA calibrated) benchtop current sources that can be controlled by computer. We combined these to make a jig that automatically determines the average power consumption of a device under test, despite duty cycling between 5uA and 20mA.

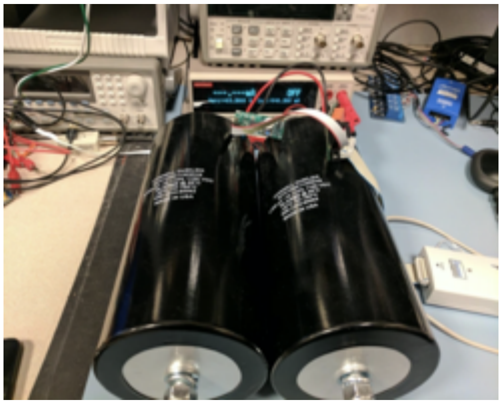
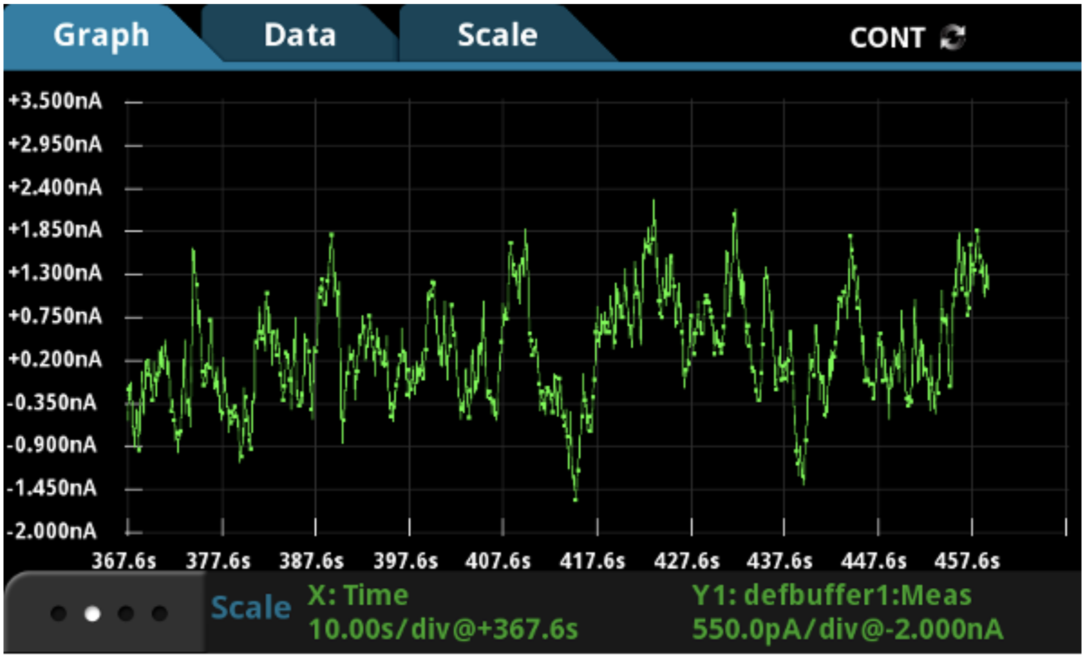


Figure 24: Photo showing large capacitors in measurement jig.

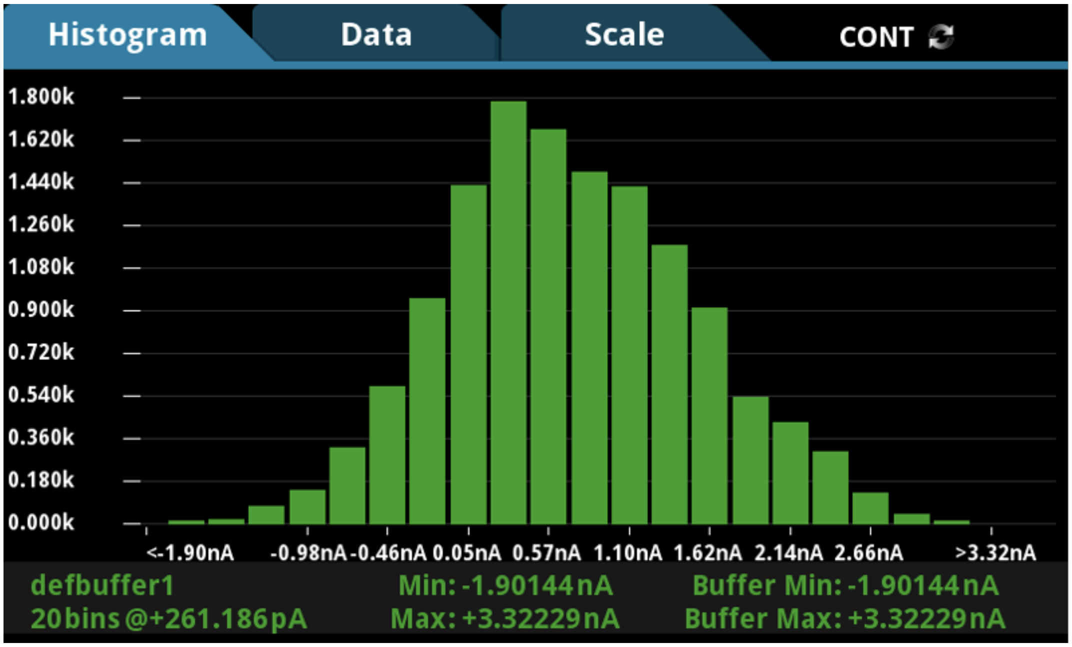
The following figure is a trace of the capacitor leakage over a few 100 seconds, showing the random nature of the leakage. Note that this is not always a positive number, because when capacitance fluctuates and increases, it manifests as the voltage of the capacitor increasing, so current flows out of the capacitor.



X: Time
10.00s/div@+367.6s
Y1: defbuffer1:Meas
550.0pA/div@-2.000nA

Figure 22: Trace of the capacitor leakage.

The figure below shows that the leakage is a distribution around 500pA with all samples falling below 3.3nA. As we are measuring quantities in the microamperes, this is definitely sufficient. It is encouraging to discover this, as it reinforces the merits of this measuring method.



defbuffer1
20bins@+261.186pA
Min: -1.90144nA
Max: +3.32229nA
Buffer Min: -1.90144nA
Buffer Max: +3.32229nA

Figure 23: Distribution of capacitor leakage

In this measurement methodology, a known current is supplied to the DUT in parallel with the capacitor. The current can be thought of as a threshold and we can accurately determine if the average power consumption of the device falls below that threshold or not.

The plot below shows the voltage on the capacitor over time with a test current of 50uA. The Device Under Test is a Hamilton-3C with 20 second sampling and encryption firmware (this is the image we have deployed in almost all deployments). We focus on the flat portions of the waveform at 3.4V (the target charge-to voltage). The longer those flat portions, the further below the target current the device is operating at. If there are no flat portions and the waveform is a slowly decreasing sawtooth, then the device is operating above the target threshold.

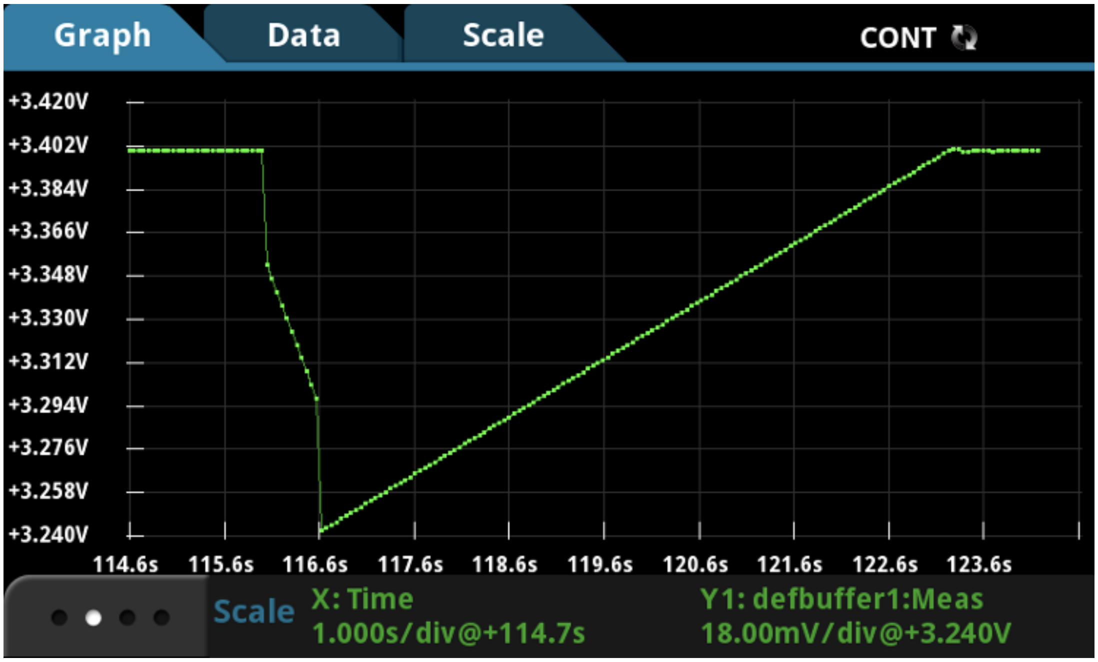
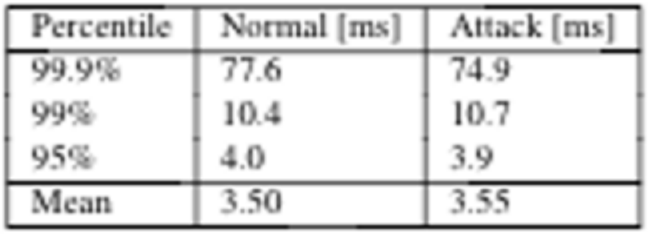


Figure 24: Hamilton 3C under 20s sampling test showing operation under target current

To date 1200 Hamiltons have been produced. With each device being a complete, networked system, we were able to create an in-situ test facility within the manufacturing line that programs each device with a full firmware image and tests operation all the way to the radio. With this, we achieve ~98% yield, but more important essentially eliminate the cost associated with even small numbers of faulty devices appearing in the field. The contract manufacturer replaces faulty components to provide 100% yield.

DDoS Protection To empirically confirm the effectiveness of an entity-blacklist DDoS policy at the router, we measured the impact on legitimate traffic when the designated router is under an attack comprising 130 new connections per second, each with a unique entity attempting unauthorized actions. To sustain an attack of this magnitude, an attacker would need to spend approximately 360 million gas (the Ethereum blockchain's unit for transaction costs) per block in the chain that registers new entities. This is two orders of magnitude larger than the gas limit of the main Ethereum chain and therefore serves as a generous upper bound.

The legitimate traffic is a synthetic stream of smart meter data, approximating a large city at roughly 1600 messages per second. We observed 10 minutes of traffic under normal conditions followed by 10 minutes under attack conditions. The end-to-end message latency is broken down in the table below. The blockchain rate limiting of identity creation effectively renders this attack vector irrelevant, having no detrimental effect on legitimate traffic



Percentile	Normal [ms]	Attack [ms]
99.9%	77.6	74.9
99%	10.4	10.7
95%	4.0	3.9
Mean	3.50	3.55

Table 5: End-to-end latency of data streaming during normal periods and under attack.

Blockchain Characterization: A primary concern in using a blockchain is the cost of an agent participating in the chain. We performed a large-scale test on over 100 BOSSwave nodes and emulate different levels of load on the blockchain to observe the impact on these nodes in terms of CPU, memory, and network bandwidth consumption. We also monitored the current “age” of the chain on each BOSSwave node. This is a measure of consistency specific to blockchain-based systems and the primary metric of fitness for purpose. If an agent cannot keep up with chain updates, it must choose to drop messages or elevate its trust in the router, as it is potentially unaware of recent revocations.

We ran BOSSwave on three different platforms. Docker containers are used to encapsulate 100 nodes, consisting of both miners and agents, on Amazon EC2. We use seven m4.16xlarge instances featuring Intel Xeon CPUs and SSD-backed storage that guarantee a baseline performance of about 1000 IOPS. We randomly distribute the containers among these hosts with the constraint that no host runs more than 16 containers. This ensures that each host has at least 4 CPU cores and 8 GB of memory per container. A container is thus assigned to one of four networking classes based on a profile of wired Internet connections in North America. In addition to the AMD64 cloud nodes, three agents are run on Raspberry Pis with quad-core ARM v7l CPUs and 1 GB of RAM and three agents are run on i386 machines with Intel Atom CPUs and 2 GB of RAM.

No BOSSwave node was CPU-bound at any point. Miners use multiple CPU cores regardless of the load on the chain, as expected. CPU consumption for regular agents was higher when the blockchain was under attack, but never exceeded ~1/2 of a core. We observe no issues on machines with at least 2 GB of memory. The bandwidth used to participate in the chain is reasonable even during periods of excessive load.

Development of standardized BOSSwave interfaces for common devices: Device drivers will expose multiple interfaces on BOSSwave: a vendor interface exposing the full native functionality of the device, and a standardized interface implemented on top of the device’s native features. The standard interface simplifies development because BOSSwave applications do not need to alter their implementation when working with a device from a new vendor, which may have different native functionality. The current set of BOSSwave drivers is at https://github.com/SoftwareDefinedBuildings/bw2-contrib

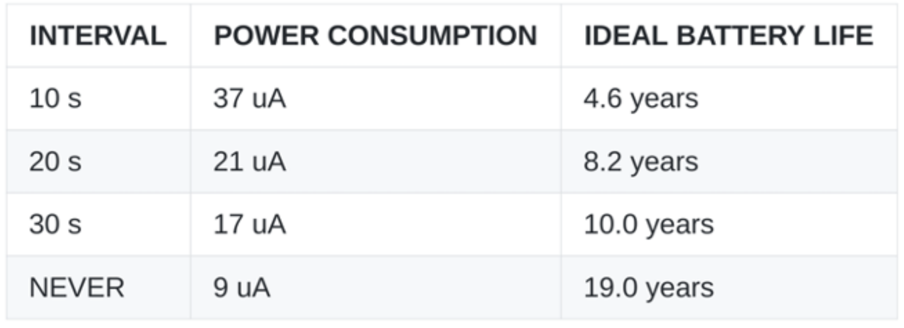
Because the content of BOSSwave messages do not follow a uniform schema, Archive Requests must be able to express how to programmatically extract the relevant timeseries information from each received message. To this end, we have developed "ObjectBuilder" https://github.com/gtfierro/ob a small DSL for extracting fields from structured data independent of its serialization format. This is used by the BOSSwave archiver.

Budget Period 1 Go/No-Go

Metric 1: *Design and implementation of Generation I sensor node with functional embedded operating system and networking stack demonstrating not more than 10% greater than 44.5 microAmp average power consumption at a per 5 minute average report rate in scalable deployment scenarios of up to 50 nodes per contention range. This corresponds to 5 years on a 1950 mAh battery and represents a challenging combination of extremely low idle current, rapid activation, robust communication and sensing at <0.1% duty cycle, and extremely parsimonious networking protocols.*

In the first quarter, we measured the idle and active power consumption of the Hamilton prototype version 1. The idle power consumption of just the MCU was 2.65 microAmp (uA) and with the sensor pack was 5.2 uA. Compared to the Go/No-go goal of 44.5 microAmp at a 5 minute average report rate, the v1 Hamilton measured 18 uA at a 30 second report rate. Thus V1 consumes less than half of the energy at 10 times the report rate of the Go/No-go goal. We expect V2 with an expanded sensor pack to consume more energy.

We measured the idle and active power consumption of the Hamilton V2 3/7C (see Table below and discussion above). At 20s data interval, the Hamilton has achieved the Go/No-Go goal.



INTERVAL	POWER CONSUMPTION	IDEAL BATTERY LIFE
10 s	37 uA	4.6 years
20 s	21 uA	8.2 years
30 s	17 uA	10.0 years
NEVER	9 uA	19.0 years

Table 6: Power consumption and battery life at various data intervals

Metric 2: *Functional implementation of BOSSwave Router and Agent demonstrating the capacity to route the equivalent of 100,000 sensors at a 5 minute average reporting rate and achieve an average transaction propagation (i.e., change in permissions structure) of under 5 seconds with 10 Routers, 100 Namespaces, 1000 Agents, 10,000 Entities, 100,000 resources, and 1,000,000 delegations of trust (DOTs).*

We conducted an initial functional implementation of the BOSSwave Router and Agent. We demonstrated the capacity to route the equivalent of 100,000 sensors at a 5 minute average reporting rate, which is 333 messages per second. Our tests showed a capacity that surpassed this rate, at approximately 1600 msgs/s on a single router. We observed an average transaction propagation time of less than 5 seconds using one router. We initially tested seven routers concurrently. Our testing used 10,000 Entities successfully and successfully showed 1,000,000 delegations of trust (DOTs). We added more routers as part of our continuously running infrastructure; we have shown more than ten routers managing traffic for tens of namespaces and millions of datapoints a day. This has given us operational confidence in the system, which has been operating at this scale for over a year. To extend this, we have performed two independent large scale studies: one studying the scalability of the system under the emulated load of San Francisco was previously discussed and published at ACM BuildSys (Andersen, Kolb, Chen, Culler, et al., 2017).

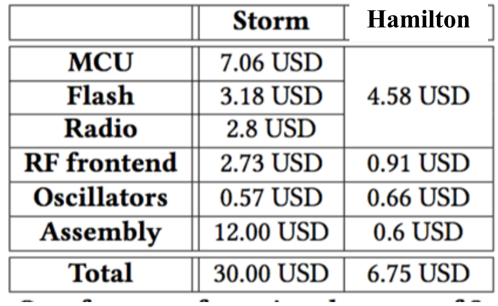
We performed a second study which extends the prior work adding 10 new routers, 100 new namespaces, 1000 running agents and 10,000 new entities publishing on 100,000 resources at a rate of one message every 4 minutes. Combined with the previous experiment there are more than two million entities and five million DOTs, far exceeding the Go/No-Go milestone. We add that this experiment was performed while operating control and monitoring in more than 20 buildings on the same shared global system without any downtime, illustrating that the system has the ability to operate at an even higher than the Go/No-Go milestone, which is encouraging.

Budget Period 2

The scope of the first budget period was to produce the technical foundations of the low-cost sensor tier and the secure attack-resistant middleware tiers (the building intelligence layer), with preliminary integration steps. Work in the second budget period focused on the design and development of a second-generation sensor platform to support additional sensor modalities and applications. BOSSwave was renamed WAVE, and was expanded to provide secure, interoperable encapsulation of various commercial products that provide network access to devices providing typical building functions, such as lighting, thermal management, and metering. Initial versions of occupant-centered control were developed.

We designed, implemented and manufactured a Hamilton based plug load monitor and Hamilton powered router. In addition, we constructed the Hamilton-7C, which extends the original Hamilton design to incorporate a higher accuracy temperature and acceleration sensor along with humidity, radiant temperature, motion (via PIR) and magnetic field. The Hamilton-10 manufactured in Oakland would cost $10.90; our original $9.80 design is still valid but uses a significantly less accurate temperature sensor and does not sense relative humidity.

The cost breakdown of the Hamilton core along with comparison to a high-quality previous generation design is shown below. The reductions obtained from the availability of very recent highly integrated system-on-chip designs permit aggressive design for manufacturing techniques that essentially eliminate assembly costs. This open-source core places minimal constraints on the underlying PCB, allowing it to be dropped into vendor designs.



	Storm	Hamilton
MCU	7.06 USD	4.58 USD
Flash	3.18 USD	
Radio	2.8 USD	
RF frontend	2.73 USD	0.91 USD
Oscillators	0.57 USD	0.66 USD
Assembly	12.00 USD	0.6 USD
Total	30.00 USD	6.75 USD

Table 7: Initial cost of Hamilton compared to precursor.

We developed WAVE drivers for three thermostats, two lighting systems, two plugload monitors, two electric vehicle chargers, an energy monitor, utility interval meter gateway, a photovoltaic system, and several weather stations. We demonstrated cloud-based energy analytics, implemented a schedule and a Model Predictive Controller in a small commercial building to optimize HVAC energy, occupancy and electricity price. A start-up, HamiltonIOT (https://hamiltoniot.com/), was created to sell the sensors, border router, and provide data services. We have kept track of industry interest and used this to develop a list of potential building product applications.

This system has been deployed in multiple test beds: Soda Hall, Jacobs Hall, Sutardja Dai Hall, test sites for the CEC-funded EPIC fans project, CIEE office in downtown Berkeley, as well as residential locations. Initial integration of these technological innovations is performed through the creation of execution containers containing the WAVE agent and various driver, proxy, or building system function logic.

Task 5 Generation 2 Low Cost Wireless Sensor System

This section describes the hardware design of the next version of Hamilton.

Sensor Hardware

We improved upon the deployment process of Hamilton to reduce total cost of ownership. No static pairing is required between sensor and border router: sensors can move and be replaced. The Hamiltons have QR codes carrying:

- sensor pack (which sensors are on the device)
- unique ID
- hardware version
- firmware version

In addition, the Hamiltons automatically determine their orientation; one merely has to capture location.

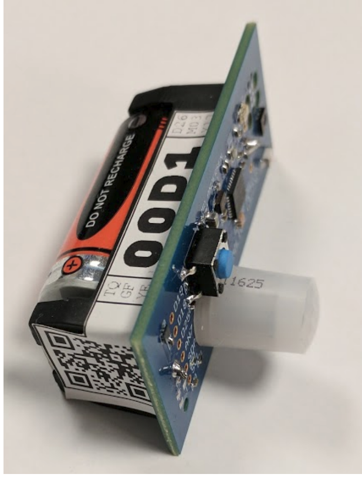


Figure 25: Hamilton with occupancy sensor

The current default set of sensors includes: temperature, radiant temperature (90degree cone), relative humidity, magnetometer, accelerometer, and light. There is also an optional occupancy (Passive InfraRed) sensor.

Thus, the readings include:

- Magnetic field - 3x 16 bit X Y Z (more precision than ever required)
- Acceleration - 3x 14 bit X Y Z (more precision than ever required)
- Air temperature - 14 bit (0.1 C repeatability, 0.2 C accuracy)
- Air Relative Humidity - 14 bit (0.1 % repeatability, 2% accuracy)
- Illumination - (1 lux precision, not sure about accuracy and repeatability yet)
- Radiant temperature - still need to calibrate
- Aggregated PIR occupancy fraction (e.g., 1s/20s occupied)

In addition, we explored the addition of a “ranging” sensor that uses ultrasonics to detect motion and a VOC sensor.

We designed, implemented and manufactured a Hamilton based plug load monitor, pictured below:

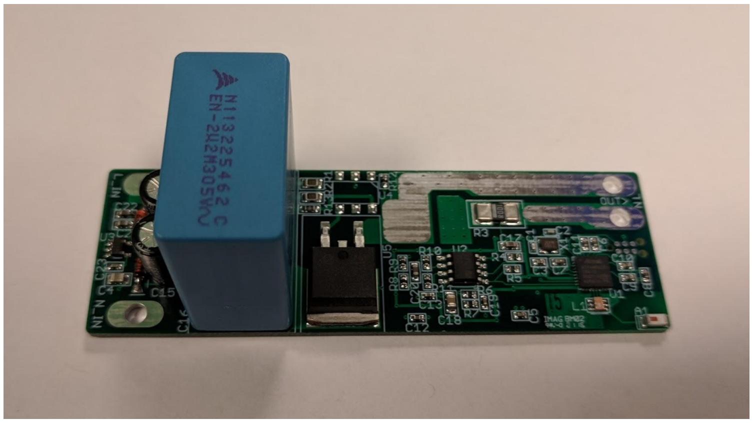


Figure 26: Hamilton-based plug load monitor.

This unit is novel in that it is very inexpensive to manufacture (approximately $8 per unit at large quantities) while also sensing more than existing entry-level plug load monitors: it not only senses the line voltage and current but also the power factor of the given load. It offers actuation of the load, which is also often omitted in plug load monitors.

The envisioned role of the plug load monitor was two-fold. Firstly, it would provide plug-level metering for use in building operation. Secondly the always-on nature of the plug load monitor would allow for routing of other Hamilton radio traffic in a modern OpenThread routing framework that utilizes powered routers with ultra-low power leaves.

After manufacturing and testing the units and evaluating their deployment, we determined that obtaining the necessary certifications (e.g., UL) to deploy at scale it would involve costs appropriate for later stage commercialization, so we determine, with program management agreement, not to deploy these units beyond laboratory settings. The milestone is successfully obtained, as we have demonstrated that a transformerless low-cost actuating plug load monitor can be constructed for less than $10.

During the course of the project, commercial plug load monitors have become available that fill the role of plug load metering, albeit at a higher price point ($30 per unit), and have been incorporated into the project. The successful demonstration of our transformerless design is now available to the market to advance technological options to further reduce cost.

To fill the remaining role of a router for Hamilton traffic, we designed and implemented the **Hamilton Powered Router**, pictured below:

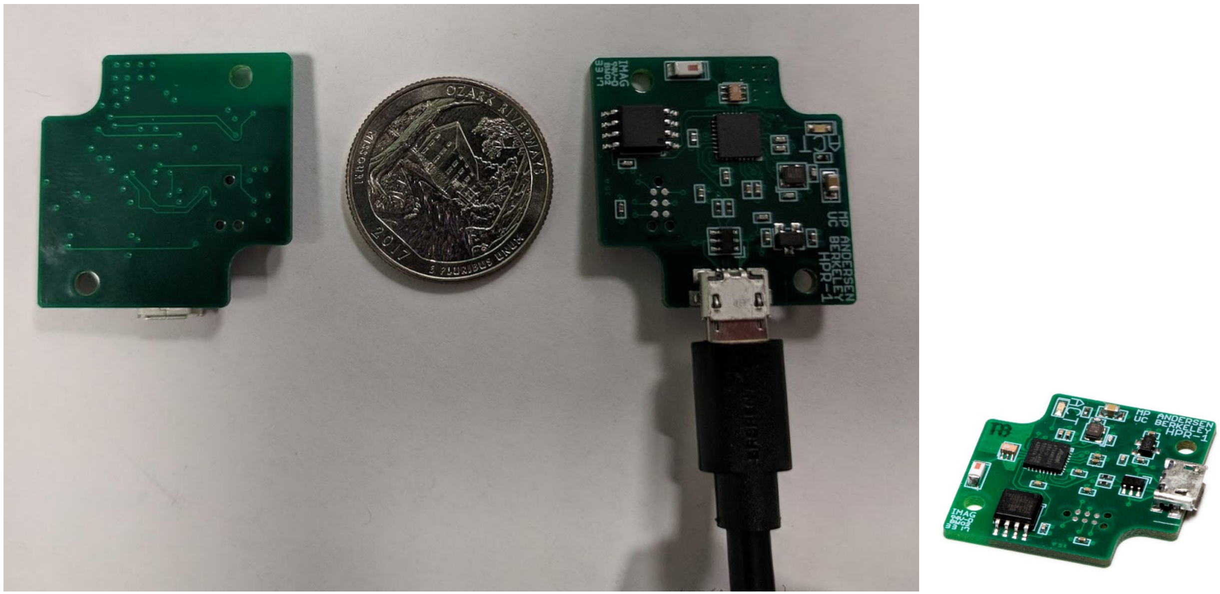


Figure 27: Hamilton-powered router

This is an ultra miniature (slightly larger than a US quarter) Thread-compatible router that draws its power from micro USB. Like the Hamilton, it has a similar core (SAMR21 SoC + Balun + Chip antenna), has a JTAG programming interface, and is small and inexpensive. Unlike the Hamilton, it has no sensors and has a mini-USB port for its power source. The form factor is compatible with an off-the-shelf enclosure allowing the whole unit (including an enclosure and a USB power supply) to be manufactured for less than $15. We also added some flash storage to permit experimentation with in-mesh deep buffering.

In addition to the plug load monitor and Hamilton Powered Router, we have constructed the Hamilton-7C, which extends the original Hamilton design to incorporate a higher accuracy temperature and acceleration sensor along with humidity, radiant temperature, motion (via PIR) and magnetic field. We had built the 2nd generation Hamilton (Hamilton-3C) with a wide variety of sensors to get an idea for what people would want. Based on usage data and conversations with early Hamilton users, the most sought-after sensor is air temperature, followed by air humidity. Given this, we sought to discover what a platform with just the best-in-class of those two sensors would cost. The Hamilton-10 manufactured in Oakland would cost $10.90; our original $9.80 design is still valid but uses a significantly less accurate temperature sensor and doesn't sense relative humidity.

This is pictured below:

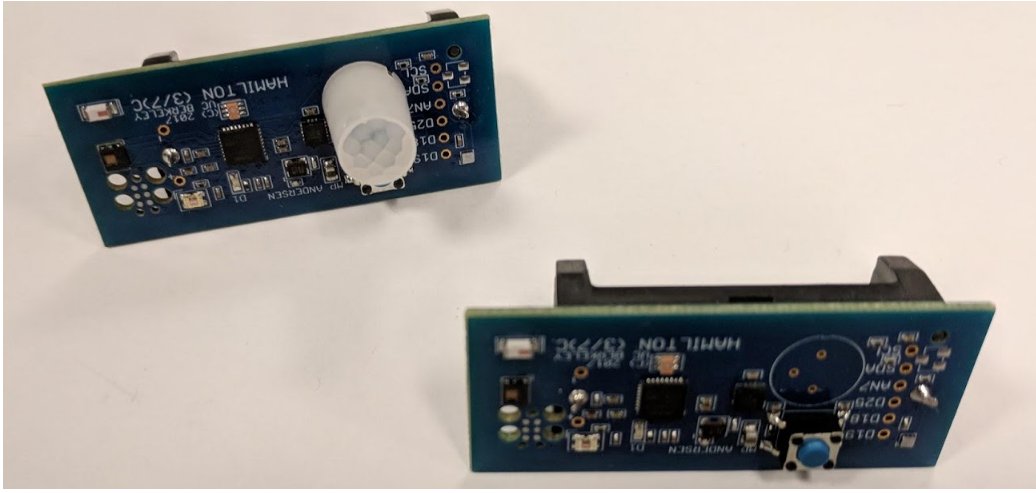


Figure 28: Hamilton 7C with improved sensors.

These were manufactured both with and without the motion sensor (the white cylinder) as the sensor contributes significantly to the cost. These units cost approximately $20 without the motion sensor and $40 with the motion sensor. We pursued this design after feedback from building operators that the cost of the sensor was not as significant as the cost of installation and they would rather pay more per sensor to obtain higher accuracy and multi-modal sensor information.

Secure border router

We are continuing to work on the robust secure transport of data from the Hamiltons to the Border Router and then to the WAVE secure bus.

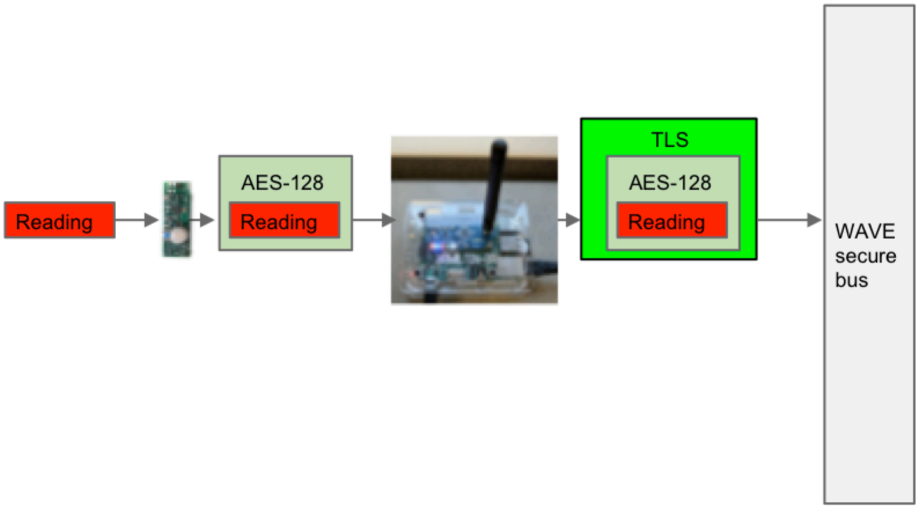


Figure 29: Transport Level Security and Hamilton sensor readings routing to secure bus.

In Budget Period 2 we had 19 separate deployments of the original Border Router (pictured below), some of which have been active for several months.

Figure 30: Border router

We continue to push our low-power embedded operating systems developments into the most active open source community for this tier of design, RIOT (riot-os.org), thereby making the work widely available for transfer to market.

All Hamilton-related drivers (board, sensors) have been uploaded to RIOT-OS's open source community. It typically takes at least several months for this code to be eventually merged onto RIOT's main release, due to the review process, standardizing code style and so on. Thus, we also provide an open repository of our current releases.

We completely rewrote RIOT’s ETHOS module to create RETHOS: Reliable Ethernet Over Serial. This includes:

- 1 Mbaud serial to Raspberry Pi
- 255 channels (to distinguish stdio, raw network, debug, L7 GW
- Strong checksums (Fletcher) with frame retry & queueing

The RIOT maintainers requested that we provide five Hamiltons, so that they can include Hamilton boards in the RIOT's test procedure. Thus, going forward the Hamilton release will be part of the mainline test and QA process. Whenever RIOT is updated, they must make sure that the updated code operates on Hamiltons. With this procedure, we no longer need to keep following the RIOT's upstream update and adjusting the updated code for Hamilton since they will do that instead of us.

We continue to develop our multi-hop communication or meshing protocols, versus the current “star” network or single hop Hamilton to Border Router. These will have lower total cost and be easier to deploy. Thread is a newly released low-power IPv6 mesh protocol from Thread Group that is optimized for network architecture comprising battery-powered leaf nodes and plugged in routers. UC Berkeley is becoming an academic member of Thread Group. OpenThread is an official open source implementation of Thread by Google Nest.

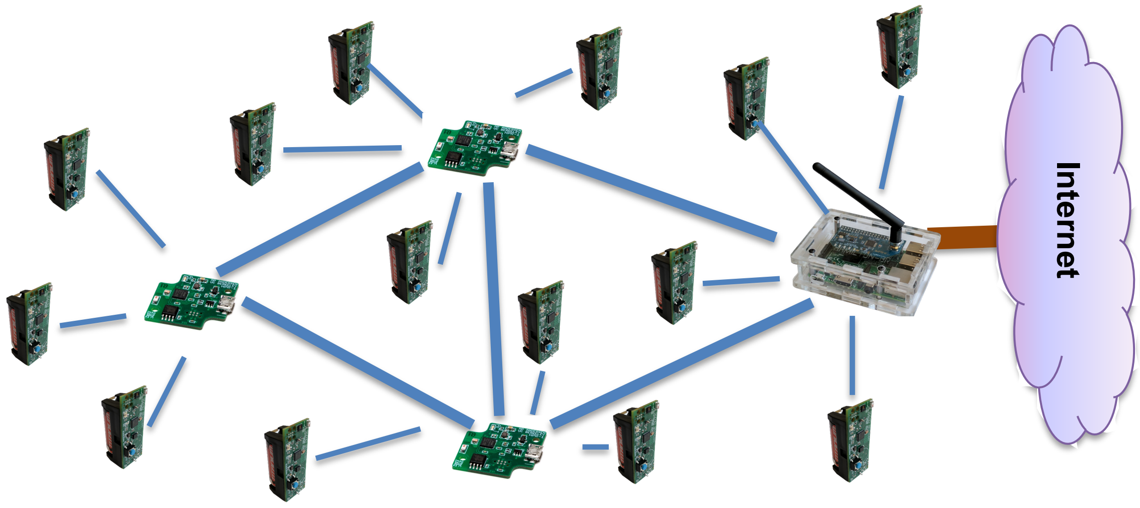


Figure 31: Multi-hop mesh networking with the border router and miniature border routers.

We continue to develop and test our OpenThread Border Router, and will modify it to use RETHOS. Thus, we will port OpenThread on RIOT-OS, and design a preemptive network architecture for RIOT-OS and OpenThread that will be more responsive, efficient, and robust. We pushed these networking developments to RIOT-OS upstream, but opened our own code separately (https://github.com/Hyungsin/Hamilton-RIOT-Openthread-fw)

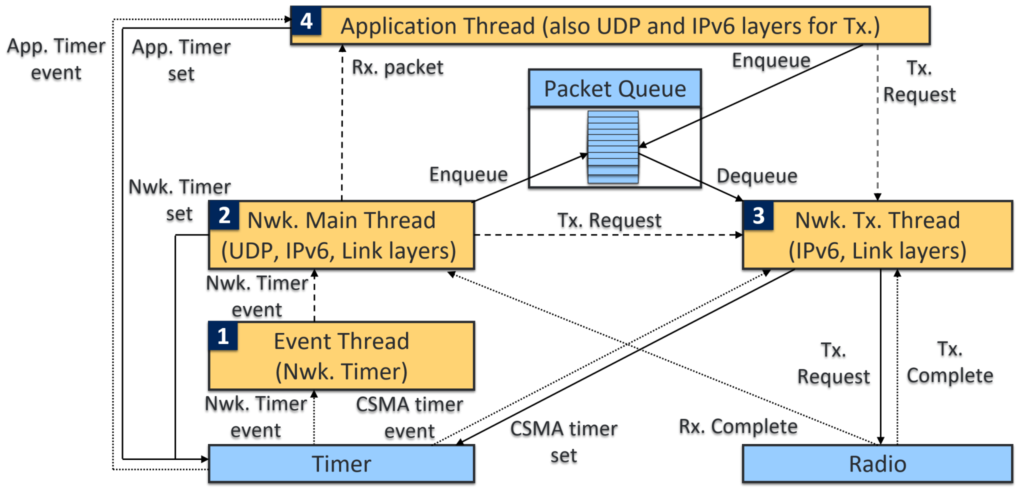


Figure 32: OpenThread in a multi-hop network.

Routing Study

We have asked the question, is TCP, the *de facto* reliable transport protocol in the Internet, viable in wireless sensor networks? And so far, the answer seems to be Yes. The protocol logic and buffers fit within memory constraints of a Hamilton sensor. It can overcome hidden terminals with delay between link retries. The congestion control is not a problem due to low bandwidth/smaller buffers. The power consumption is comparable to WSN-specific protocols. We've seen the highest TCP throughput (75 kb/s) in WSNs ever recorded. There is seamless interoperability with normal Internet services. So, yes, sensors, like Hamiltons, can be first-class citizens of the Internet!

Milestone 6.1 *Second-generation sensor node with application-specific sensor modalities streaming plug load monitoring samples at 1 hertz (Hz).*

We have been developing Hamilton into multiple application-specific sensor modalities for use in buildings, including a thermostat for HVAC systems, a desk fan and heated insole, and an anemometer. Of the three applications, the anemometer is the only one that requires 1 Hz monitoring samples. These efforts have cemented collaborations with stakeholders in industry and government.

We created a prototype of a low-cost thermostat using the Hamilton for its sensors, memory, and 802.15.4 radio output. The design is being utilized in thermostat project funded by the California Energy Commission (CEC), primed by EPRI, to create energy efficient heating solutions for the low income sector. An inexpensive wall unit is connected to the HVAC equipment (furnace, air conditioning or fan) through wires and has a very simple button interface with LED lights. This wall unit communicates to a mobile device via Bluetooth for fuller user interface and functionality.

Figure 33: Prototype wall unit with underlying Hamilton and mobile phone interface.

The figure below shows the layout of the hardware board.

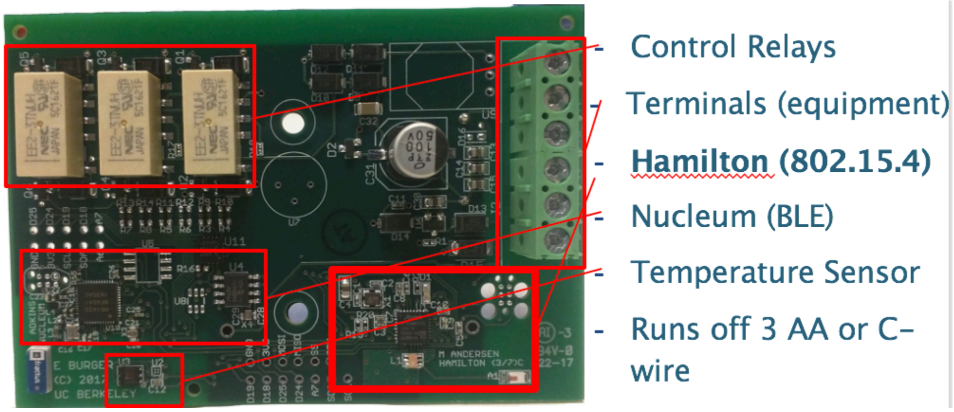


Figure 34: Hamilton embedded in thermostat design for Low Income thermostat project (CEC funded)

The second family of Hamilton-based sensor/actuator systems have been developed in collaboration with the Center for the Built Environment (CBE) at UC Berkeley, which is conducting a project, funded by ARPA-E, to develop several personal comfort system devices, including a wirelessly charged desk fan and heated insole.

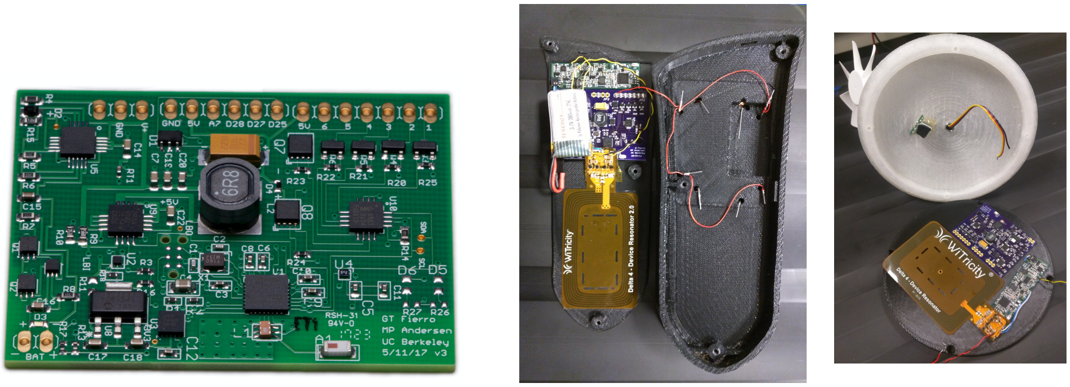


Figure 35: Hamilton embedded in wirelessly charged personal heating and cooling systems, such as a desk fan (right) and heated insole (center).

A third project, also in collaboration with CBE and funded by CEC, seeks to create a low-cost air flow sensor or anemometer using MEMS ultrasonic transceiver technology developed by collaborator Chirp Microsystems, acquired by TDK. For determining ambient air flow, the Hamilton is used to aggregate time-difference-of-arrival data from all pairs of four such transceivers in a tetrahedral arrangement and securely transport this sensor data to the cloud via a border router. A second Hamilton-based design is constructed to insert into ducts.

The vast majority of buildings in the U.S.—most offices, labs, hospitals, and data centers—use air systems for transferring energy, such as heating and cooling; air speed and direction affect human comfort and safety indoors. However, air speed and direction is almost never monitored in indoor spaces because of the expense, power draw, directional sensitivity, and fragility of existing sensors. Designers and operators use large safety factors for their minimum flow setpoints, causing widespread over-ventilation and over-cooling. The lack of accurate air speed and air flow monitoring feedback impacts indoor comfort and ventilation, indoor air quality, occupant health and safety, and about half of the energy used in HVAC. Accurate air speed and air flow measurements at a low cost will reduce the common practice of over-ventilation, which consumes needless energy as well as creating heating and cooling discomfort. Affordable ultrasonic anemometers have the potential to disrupt how air flow and temperature are measured and controlled in buildings by enabling more measurement locations, temperature-averaging across a duct cross-section, and increasing accuracy at the same time.

The Hamilton prototype room and duct anemometers are especially sensitive at low flow rates, and do not go out of calibration; fouling can only reduce signal strength, not change the time of flight of the sonic signal. Also, anemometers with wireless communication can be retrofitted inexpensively in the existing building stock where flow stations are currently nonexistent. This technology can drastically reduce HVAC energy consumption, decrease labor costs associated with testing and balancing, and provide better ventilation and thermal comfort.

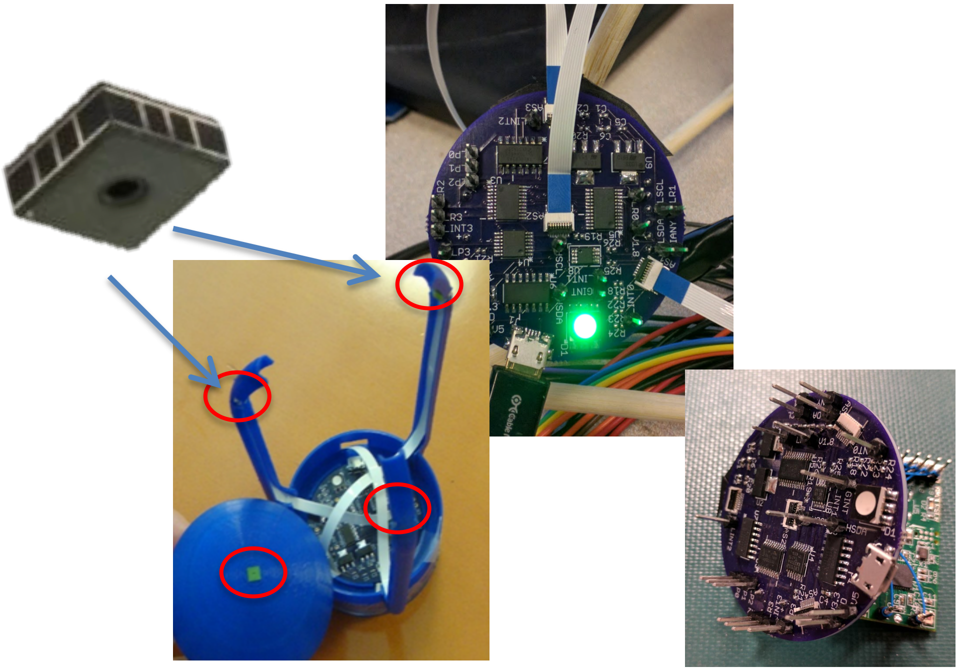


Figure 36: Ultrasonic anemometer: uses tiny ultrasonic transducers to measure time of flight translated to air flow and direction. A carrier board (round) receives the data through the cables and is plugged into the Hamilton board (rectangular board in lower right photo). The Hamilton transmits data to the border router.

The Hamilton is an integral component of the room based and duct anemometer prototypes (Figure 37). Each of the four transducers emits an ultrasonic signal that, in turn, each receives with a precise time stamp. The phase of each signal is tracked at 1 Hz to precisely determine the time of flight (ToF) of each of the six paths in both directions. These ToFs are then converted to a velocity vector with speed and direction.

The room anemometer and duct anemometer prototypes are shown below.

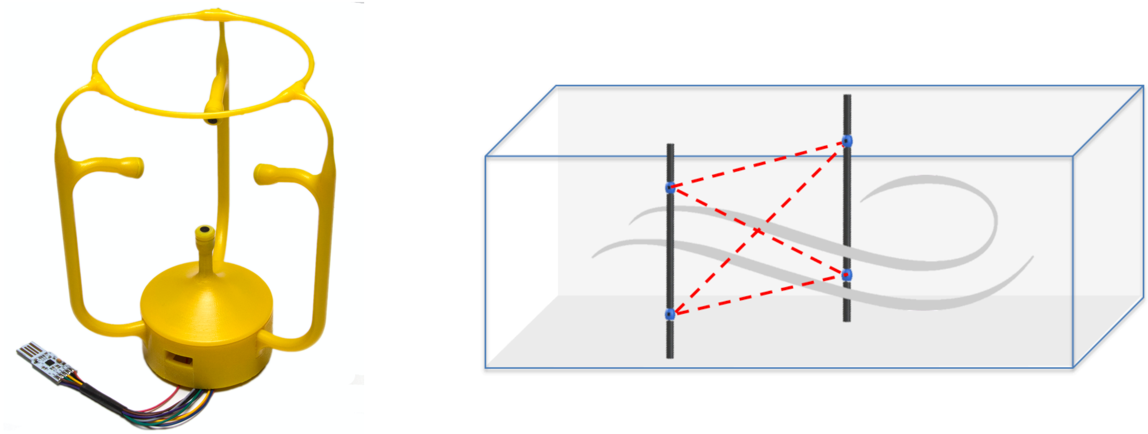


Figure 37: Three dimensional room anemometer prototype (left) and schematic of the duct anemometer "wands" (right).

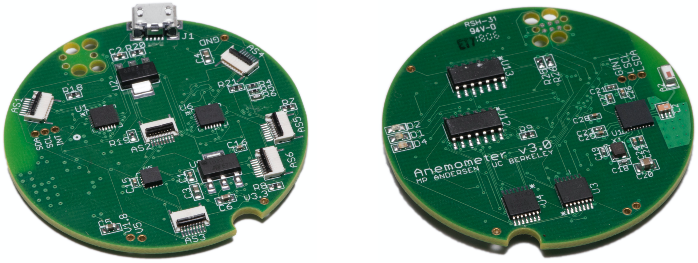


Figure 38: Final prototype 6-layer Hamilton board integrated with the anemometer carrier board.

The initial results are quite promising, as shown in the figure below:

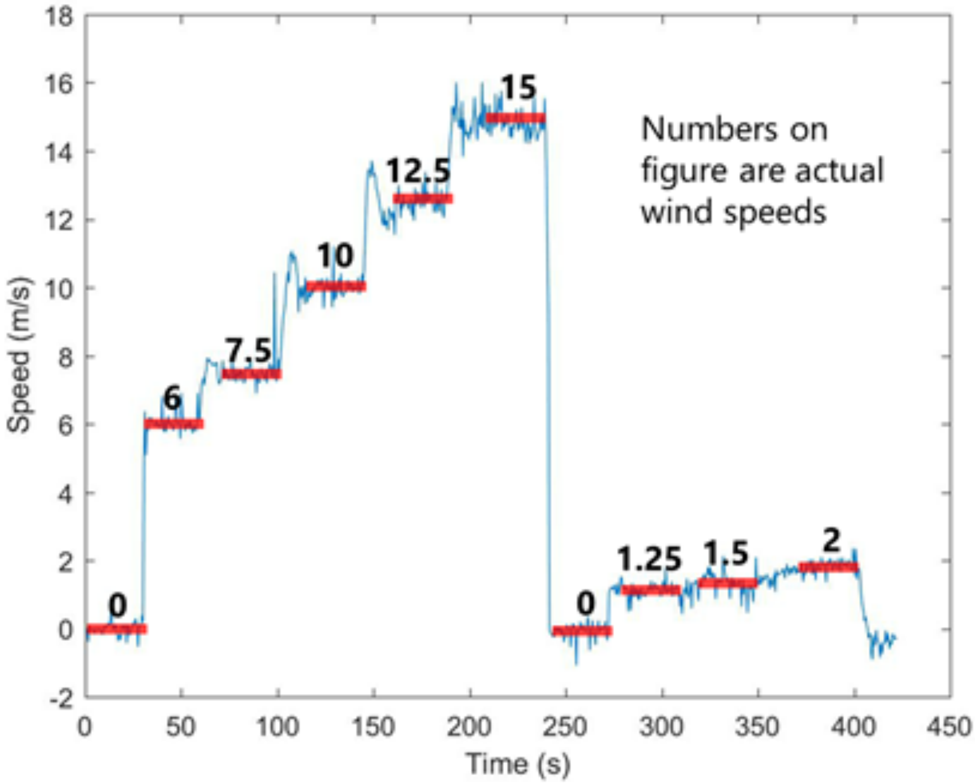


Figure 39: Average anemometer readings (red) as tested in a wind tunnel. The blue lines are actual readings showing the sensitivity of the anemometer.

The following figure shows the output of the room anemometer measuring a short duration of air flow in the test bed. The top six graphs show the velocity on each of the six paths; the four sensors are labeled from 0 to 3. The third row of graphs shows the velocity in each 3-dimensional axis. The final row shows the wind speed and angles. The blue dots are actual measurements and the red dots are filtered data.

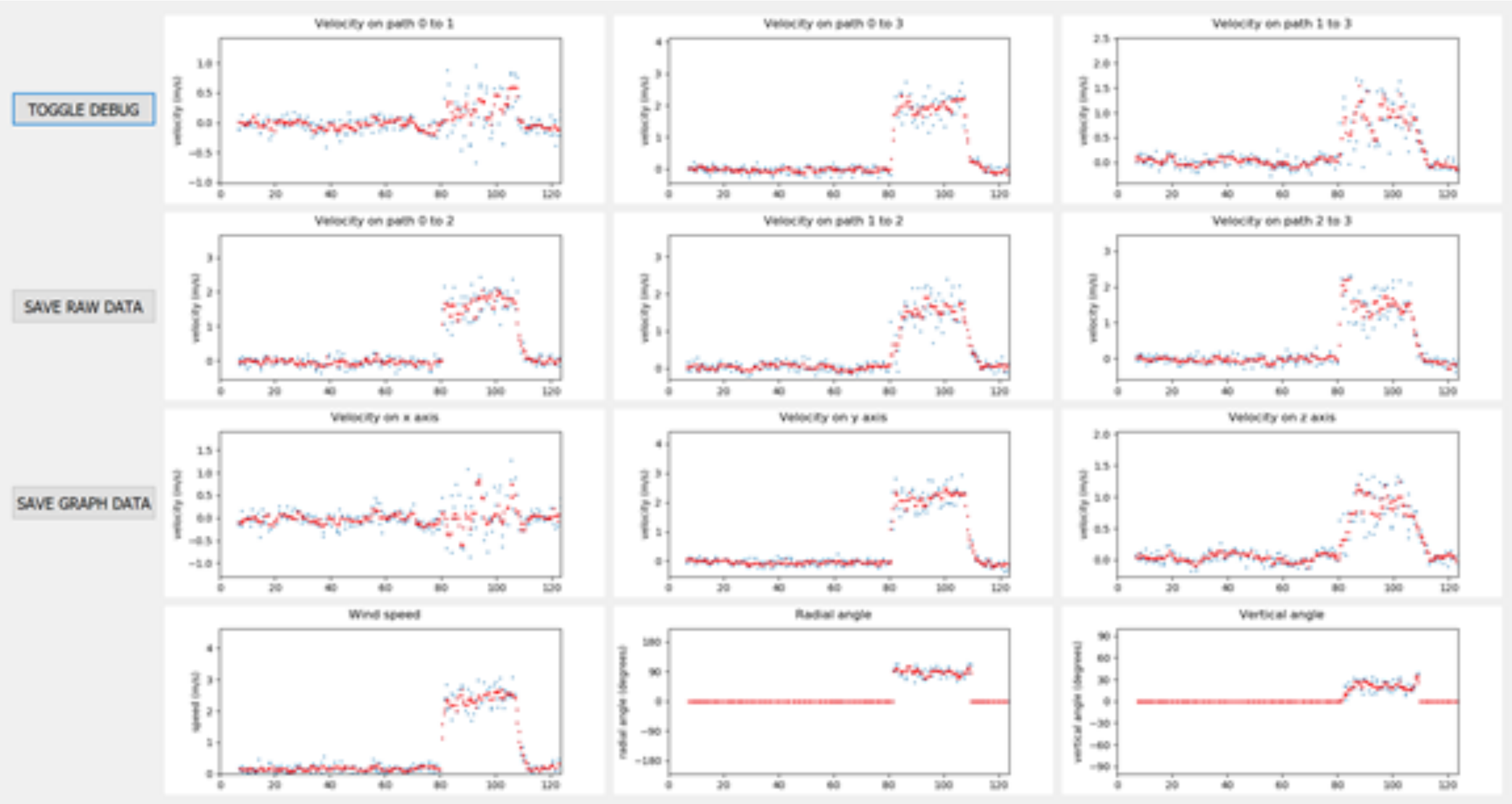


Figure 40: 1 Hz data from the room anemometer.

Task 6 Secure information bus

This section describes the design and proof of concept of the WAVE router and agent, Spawnpoint container, and comfort application interface.

Secure commercial devices

We developed a secure driver for the Venstar, Pelican, and Prolyphix iMT550C thermostats, Enlighted and LIFX lighting controllers, and TPLink and Echola plugload monitors. In addition, we have drivers for TED energy monitors, Weather Underground weather data, Rainforest Eagle interval meter data, Enphase Photovoltaic data, and Electric Vehicle chargers JuicePlug and Aerovironment.

We described over a dozen drivers created with BOSSwave. These drivers are being moved to the second generation security system, called WAVE 2.0. This new generation of security is compared to the previous version in the following table.

Gen1: BOSSwave	Gen2: WAVE + WAVEMQ
Blockchain based	Not blockchain based (VLDM)
High resource on embedded systems: remote agent	Much lower resources on all platforms Deployable on the border router directly
Specific to securing sensor telemetry	Applies to wider authorization
Very sensitive to intermittent internet connectivity	Resilient to intermittent internet connectivity

Table 8: Comparing BOSSwave with WAVE

The next development in secure routing is called WAVEMQ, which is reminiscent of Message Queuing Telemetry Transport (MQTT). WAVEMQ provides: persistent queueing to prevent data loss due to unreliable WAN connections, continued local delivery to enable schedulers and controllers to operate without internet, and WAVE security enforced at Site Router and Designated Router level.

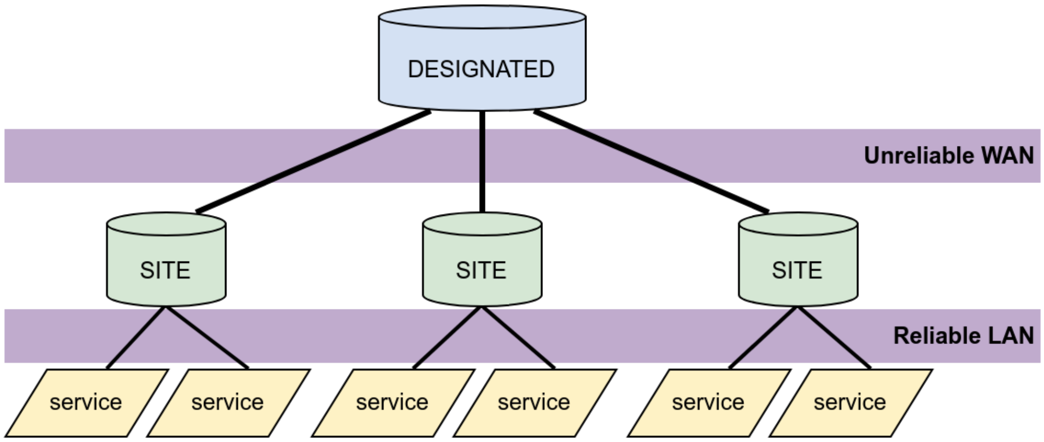


Figure 41: WAVEMQ

Secure cloud services

Milestone 7.1: *Demonstrate cloud-based building energy analytics utilizing data streamed from building sensors and building systems to a high-performance time-series database by computing correlation of variation of whole-building electric load shape to differences in temperature.*

This portion of the project leverages work in developing standardized terminology in building controls that we have conducted with stakeholders in industry and academia forming the Brick consortium—both ASHRAE and Johnson Controls are engaged in this effort. Brick is an open-source, BSD-licensed development effort to create a uniform schema for representing metadata in buildings; the collaborative effort included members from six universities and one from industry research (http://brickschema.org/). We are using Brick to describe deployments of Hamilton sensors, and how those sensors relate to the equipment, infrastructure, and environment of those deployments. The telemetry gathered from Hamilton sensors combined with the data context from the Brick model has enabled us to build interesting applications that can adapt themselves to the resources available in a given deployment.

We have developed energy analytics that use data streaming from Hamilton sensors, networked thermostats and realtime power data from the interval meter. A simple query written in Brick shows the correlation between temperature and electrical demand for summer and winter months.

We have deployed Hamilton sensor systems in a living laboratory test facility: the CIEE offices in downtown Berkeley (Figure 41). This is the top floor of a three floor building constructed in 1935, with windows on the north, east and south sides. The 7500 square foot space has both private and open office space with a conference room and kitchen. The floor plan below shows the location of the Hamilton sensors in the private offices on the south and east edges of the building. A border router in the open plan space transports the data to BTrDB via the cloud.

Figure 42: Floor plan of the CIEE offices in downtown Berkeley with location of Hamilton sensors in red.

There are four Heating Ventilation and Air Conditioning (HVAC) zones served by Roof Top Units with both ducts in dropped ceilings (the private offices have suspended ceilings (11 feet)) and exposed ducts (the open floor office area has 17 foot exposed wood ceilings). The East private offices use a WiFi Pelican thermostat connected to a Pelican gateway and the three other zones use Venstar WiFi thermostats.

In addition, a Rainforest Eagle gateway transports realtime interval meter data from the smart meter for this floor of the building. A miniature computer (FitPC) located in the conference room acts as a gateway to communicate with all four thermostats and interval meter gateway, and stream the data to the time series database BTrDB.

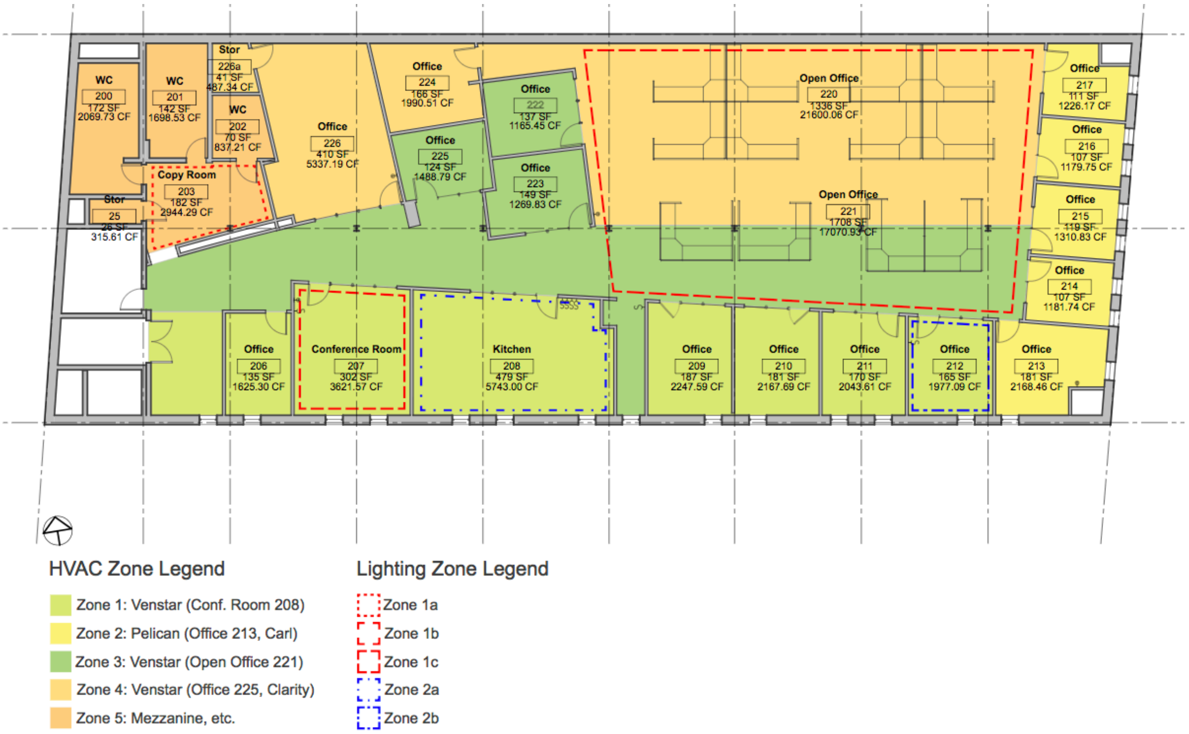


Figure 43: Floor plan of CIEE offices showing the HVAC and lighting zones.

The plot below shows the Hamilton temperature stream (in C) with the best correlation (0.475) with the demand data (kW) for the summer months (August and September). Note the heatwave in Berkeley in late August/early September when the outdoor temperatures reached 90s for several days.

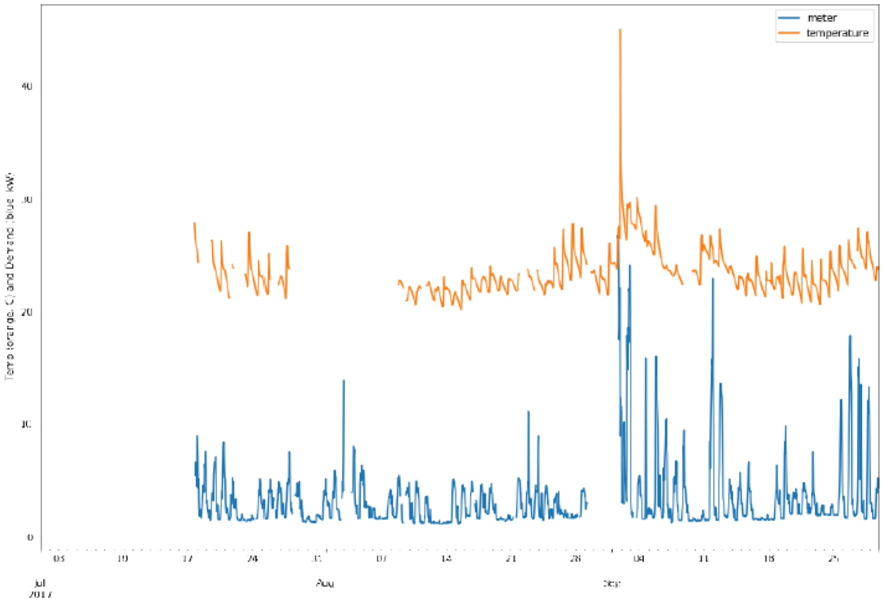


Figure 44: Data from a temperature sensor from Hamilton and the electrical demand in a small commercial building in the summer months.

The temperature data from the thermostats showed a lower correlation (0.33).

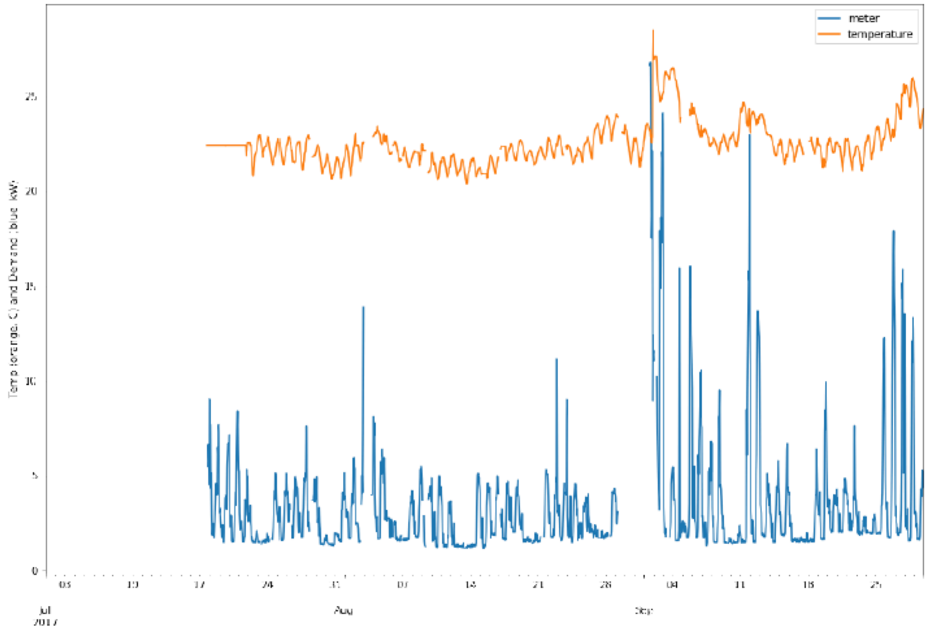


Figure 45: Data from thermostat temperature sensor and electrical demand data at CIEE in the summer months.

In the winter months (Oct-Jan), the correlation was about the same (0.46 and 0.42 respectively).

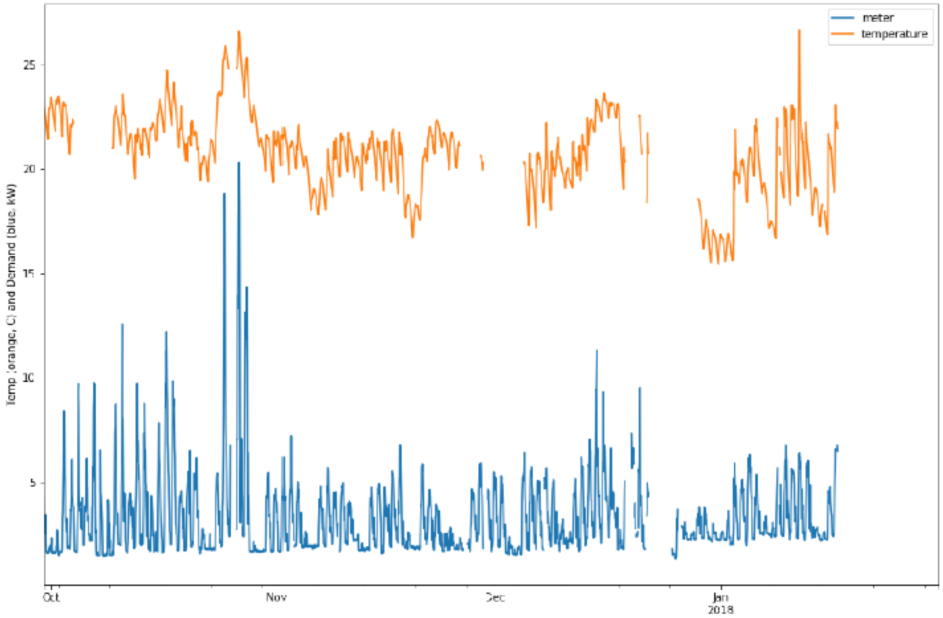


Figure 46: Data from Hamilton temperature sensor and electric demand at CIEE in the winter months.

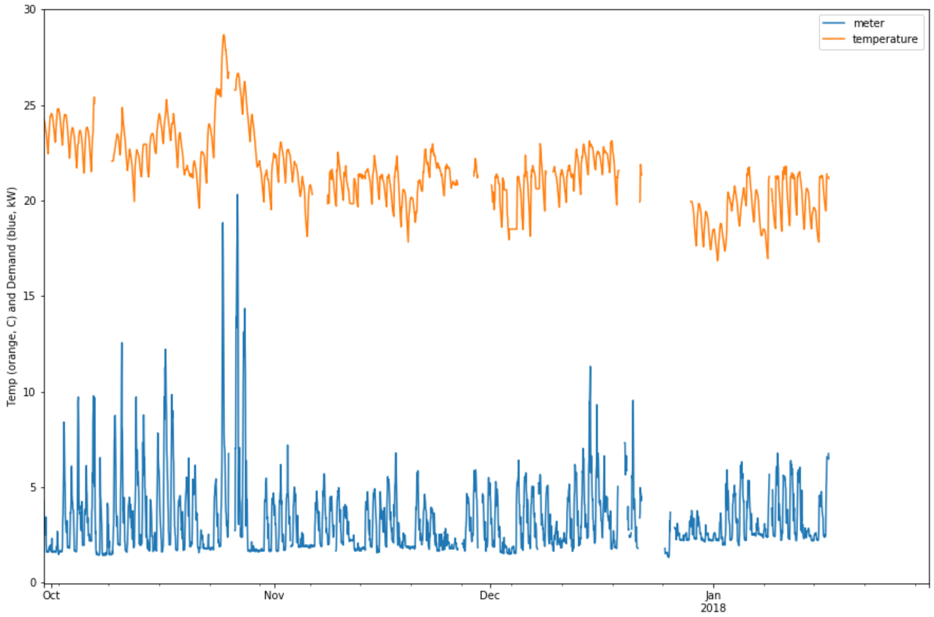


Figure 47: Data from thermostat temperature sensor and electrical demand in the winter.

Task 7: Building Application Components and Services

Scheduler

We have developed a general-purpose scheduler component that executes a description of a schedule against a set of configurable or discoverable resources.

Resources such as networked thermostats are discovered by querying the Brick model of a deployment, but can also be specified explicitly in the schedule description. Such resources are referenced using their WAVE URI name. Actuation and monitoring of these resources by the schedule is performed through a WAVE agent by the schedule's entity and therefore uses the WAVE authentication/authorization model. The scheduler process is written in Python and runs within a Spawnpoint environment.

We have also augmented the scheduler component to learn its own schedules. The scheduler can discover existing sensors (such as Hamiltons) reporting temperature and/or occupancy data by querying the Brick model of a deployment, and access the historical values of these sensors by querying an archival service. All discovery of sensors and retrieval of historical data is performed over WAVE. The scheduler learns and executes a schedule based on the thermal characteristics and occupancy patterns of the spaces it controls.

Secure Controller

We have developed and tested an MPC controller in a small commercial building, the CIEE offices in downtown Berkeley, where we have a deployment of Hamilton sensors and networked thermostats. The following graphs illustrate the savings due to the MPC compared to a static schedule for different “lambda” or optimization coefficient. The occupancy and temperature data from the Hamilton sensors is used to drive the optimizations. The results simulate the long-term average performance of CIEE’s South thermal zone on a hot summer day where cooling is needed.

We are working to improve the identification of the HVAC consumption for each zone to improve the performance. We have discovered that the Venstar thermostat temperature sensors do not have the precision of the temperature sensors of the Pelican and Hamiltons, which affects the accuracy of the MPC.

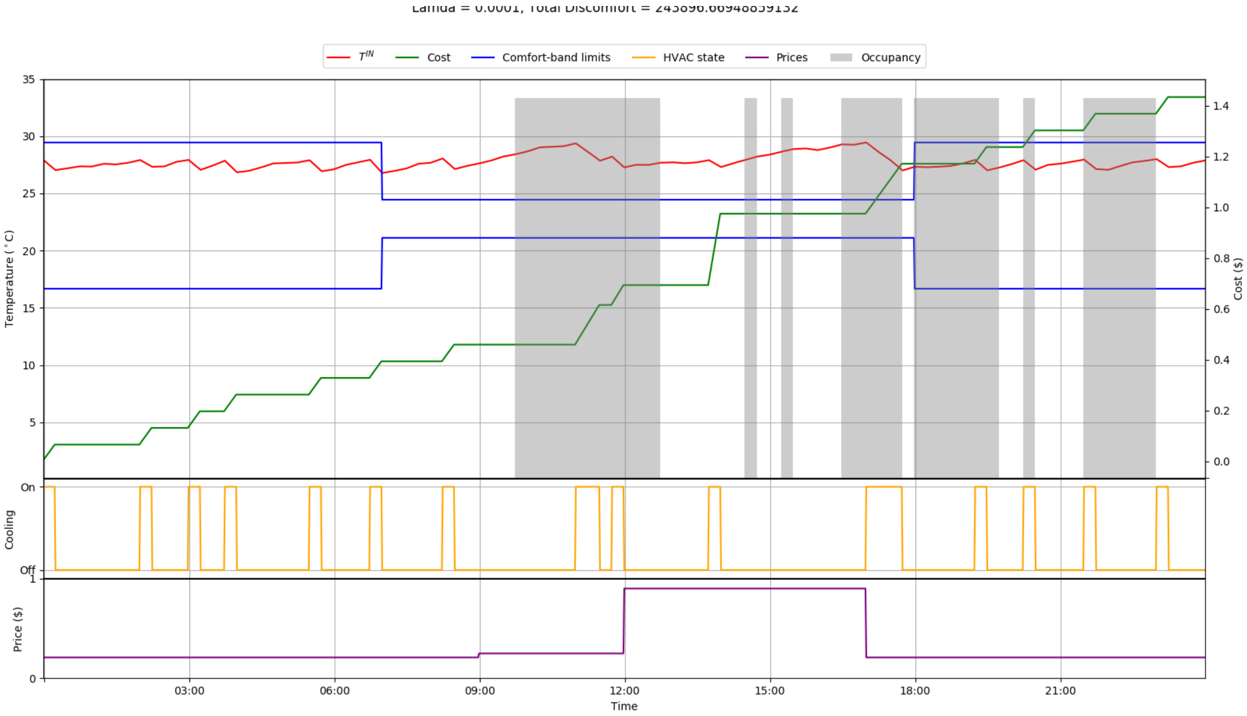


Figure 48: Performance of the Model Prediction Control of one zone in CIEE office with lambda = 0.0001 (Max savings, poor comfort). The graph depicts one day. The grey bars represent occupied times. The pink bottom line is price. The yellow line is state of HVAC (up is cooling on). The green line is accumulated cost of running the cooling. The blue lines indicate the temperature range during business and closed hours. And the red line indicates indoor temperature. Note very few cycles of the HVAC system during the highest price time.

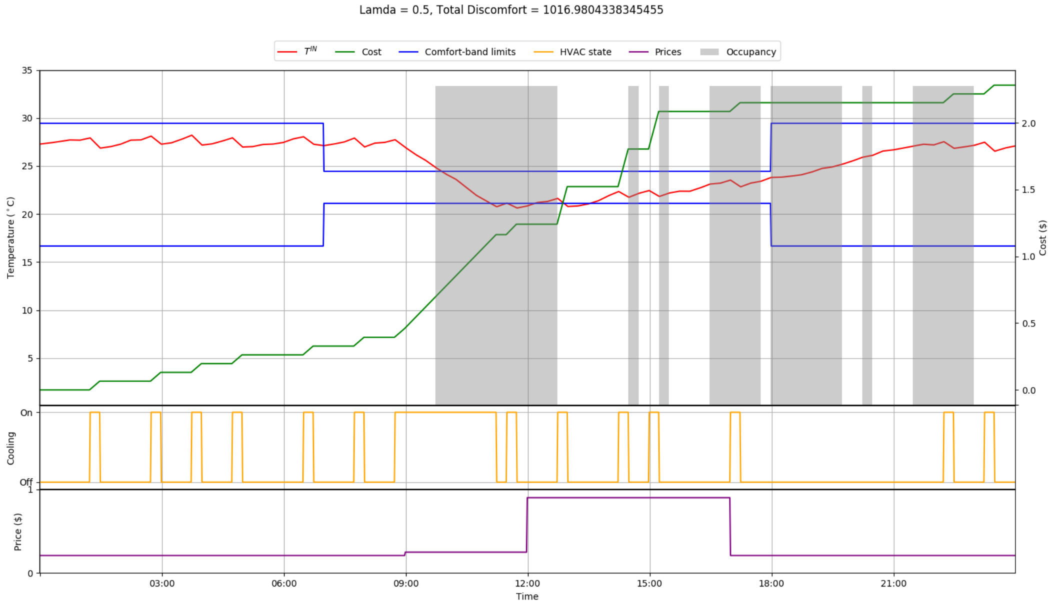


Figure 49: The same graph as in Figure 24 but with a lambda of 0.5: balance of cost and comfort. Note during occupied periods the indoor temperature (red line) is within the comfort band.

Secure interface to data management system

We have developed an integrated platform of software: applications/controls/schedule, data storage and metadata management, and hardware abstraction layer through the various drivers developed to control devices and including the Hamilton sensors. This platform is implemented in the CIEE offices in downtown Berkeley and in several other small commercial buildings as part of the CEC-funded XBOS-DR project.

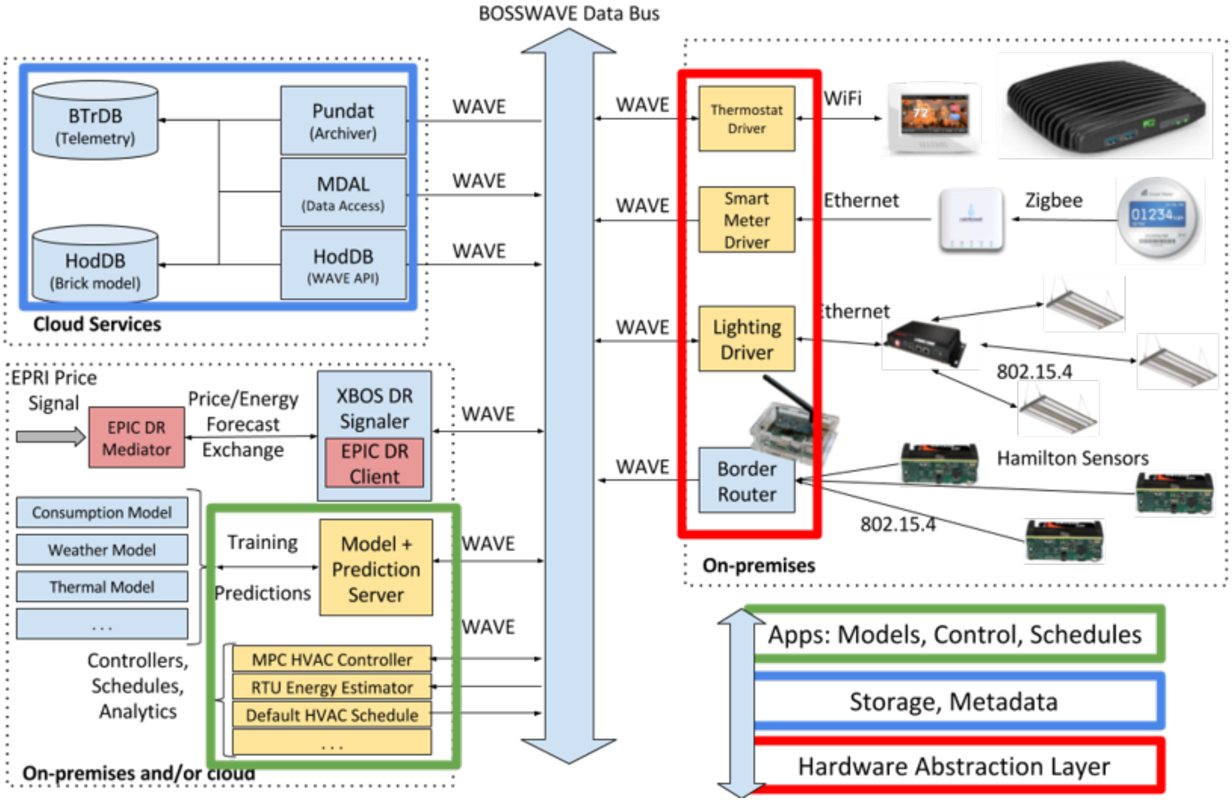


Figure 50: Hamilton sensors incorporated into building operating system platform.

Occupancy

We have installed a set of Hamilton sensors with occupancy sensors in our small commercial test bed, and have started working on control applications. The figures above show the simulated results of using occupancy data.

Milestone 8.1: Demonstrate subsecond latency of application control component actuating a building device, such as a thermostat deadband.

In previous sections we have described the MPC control at the CIEE office. We have begun an analysis of both the MPC and demand response controls in the CIEE building. The initial analysis looks at the average energy use, the power use versus temperature, and the peak energy usage days.

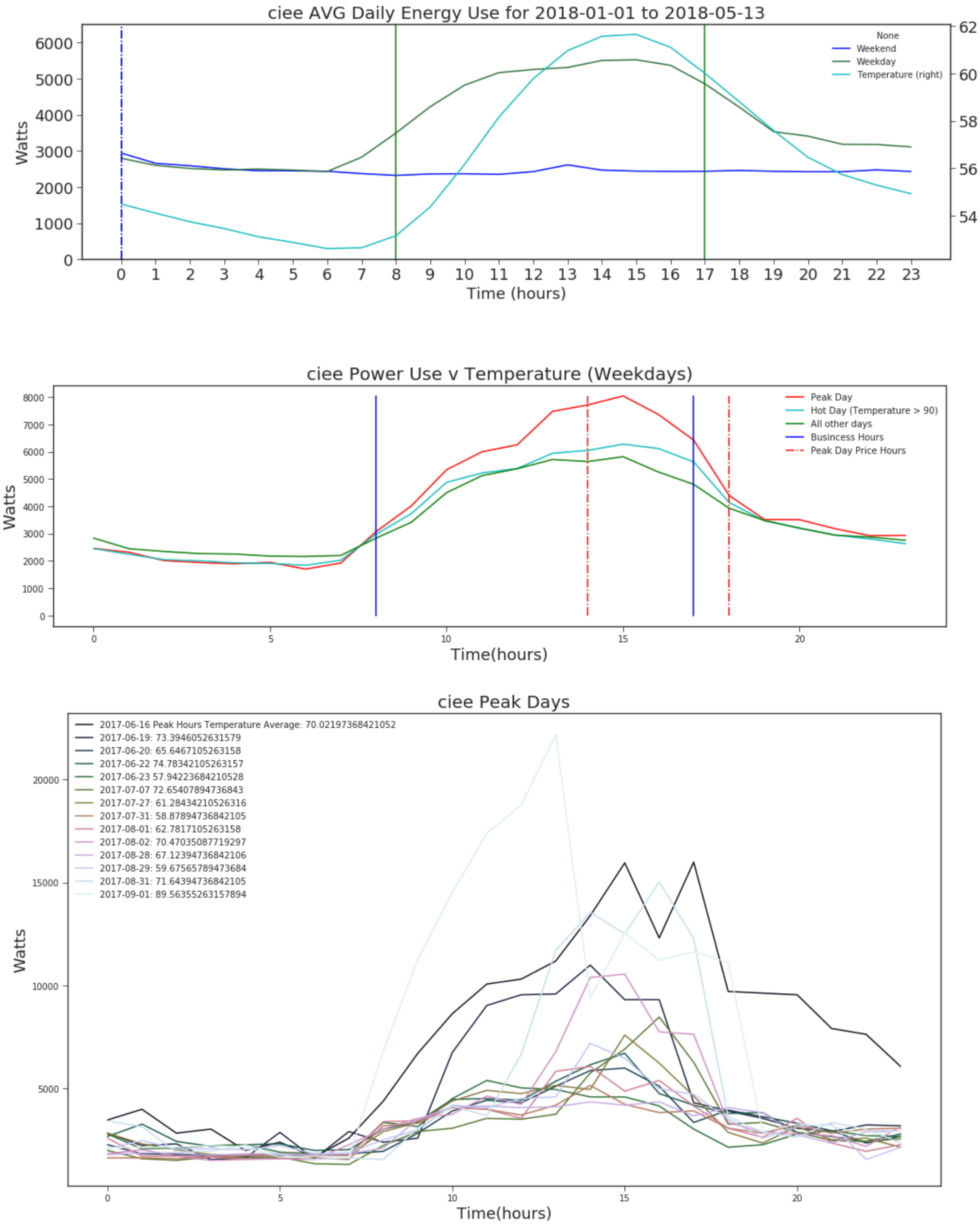


Figure 51: Energy analysis of the CIEE office.

Initial results of the MPC indicate savings compared to the baseline.

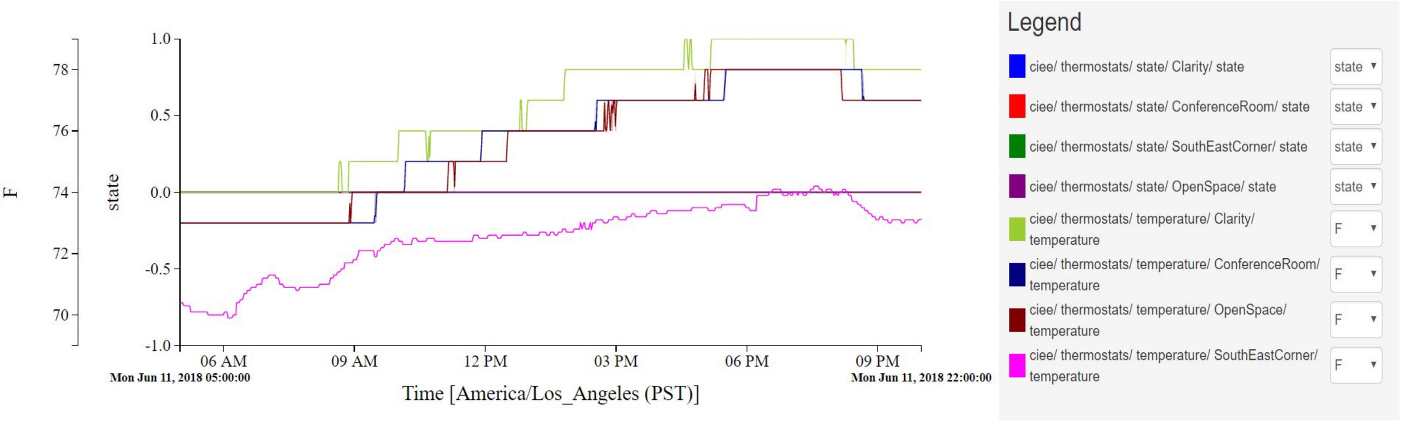


Figure 52: Temperature and HVAC state in CIEE due to MPC.

Task 8: Technology to Market

Market Discovery

Milestone 9.1 Assessment of potential applications and energy savings and environmental quality impact of low-cost sensors and secure middleware for building energy efficiency.

We have assessed several potential applications by consulting with our research colleagues on campus and listening to industry members interested in deploying Hamiltons.

Sensors	*Application*
Occupancy via PIR or via range-finding (ultrasonic) or special PIR used for counting people, Carbon Dioxide	Demand Controlled Ventilation
Occupancy via PIR or via range-finding (ultrasonic) special PIR used for counting people, temperature/RH	Demand Controlled heating/cooling
Temperature/RH	Optimizing the performance of HVAC systems by determining the temperatures achieved per zone Commissioning HVAC systems.
RH	Equivalent CO2
VOC sensor	Air quality indoors or outdoors, including VOC's like formaldehyde and benzene (a single TVOC reading is obtained); exploring equivalent CO2

Light	Tuning lights during commissioning, general energy efficiency, feedback during demand response events, maintenance.
Power	General energy efficiency, feedback during demand response events, maintenance.
	Anemometer
Anemometer (includes ChirpMicro ultrasonic sensor, and Hamilton temperature, accelerometer and magnetometer sensors)	Zone airflow in offices can be controlled to supply only the required amount of ventilation air rather than the oversizing procedures currently used to account for sensor inaccuracy. Used during commissioning or operation. Outside air intakes can be controlled optimally to balance energy and health. Dynamic feedback from airflow sensors in data centers reduces energy input required to maintain servers at required temperature. Low-cost anemometers positioned in the air handling systems of hospitals, laboratories, and clean rooms can improve safety and reduce the over- or under-pressure margin required. Direct flow measurement in ducts, fume hoods, and the occupied space will enable more responsive system control, and more immediate safety alarms should airflows be reversed.
	Platform for other applications (thermostat, personal control devices)

Table 9: List of sensors and applications that can use these sensors especially to reduce energy consumption.

Milestone 9.2: Applicability study of potential building product incorporation of project technology completed.

Below is a table showing the application goals for which Hamilton-based products could be developed and potential building product incorporation, as well as issues and barriers.

Application Goals	Potential Products	Product fit with Goals	Issues & Barriers
Low cost monitoring and actuation of plug loads	plug load monitor	Measures given load for: line voltage; power factor. Allows for load actuation.	Plug load monitoring devices are commercially available
Low cost and small sized router	miniature router	Quarter-sized router: thread compatible; micro USB powered; off-the-shelf enclosure compatible	
Energy efficient heating solutions for low income sector	low-cost thermostat	Connected to HVAC through wires; simple button interface; communicates with mobile device via Bluetooth for fuller user interface and functionality.	Maintaining low-cost BOM

Low-cost air flow sensor for determining ambient air flow	anemometer	Hamilton used to aggregate time-difference-of-arrival data pairs transceivers; secure transport of sensor data cloud.	second Hamilton-based design is constructed to insert into ducts.
Personal comfort system device control	sensor/actuator systems	Wirelessly charged desk fan and heated insoles to maintain user comfort at the person-scale.	
Multi-modal, high accuracy, low cost Environmental sensing	Stand-alone deployment	Hamilton users primarily need sensors for: air temperature (DBT) , air humidity (RH), followed by light (lux), and occupancy (PIR motion sensors).	sensor cost < installation cost; prefer higher cost sensor with greater accuracy
Demand Controlled Ventilation	Stand-alone deployment	Occupancy via PIR or via range-finding (ultrasonic) or special PIR used for counting people, Carbon Dioxide	
Demand Controlled heating/cooling	Stand-alone deployment	Occupancy via PIR or via range-finding (ultrasonic) special PIR used for counting people, temperature/RH	
Commissioning HVAC systems.	Stand-alone deployment	Optimizing the performance of HVAC systems by determining the temperatures achieved per zone using temperature/RH	
Determine levels of CO2 by proxy	Stand-alone deployment	Using the RH sensor to determined equivalent CO2	
Air quality indoors or outdoors, including VOC's	Stand-alone deployment	including VOC's like formaldehyde and benzene (a single TVOC reading is obtained); exploring equivalent CO2	
Lighting systems commissioning	Stand-alone deployment	Tuning lights during commissioning, general energy efficiency, feedback during demand response events, maintenance.	
Power monitoring and Energy efficiency	Stand-alone deployment	General energy efficiency, feedback during demand response events, maintenance.	
Control zone airflow in offices to supply only required amount of ventilation air	Anemometer	High accuracy anemometer prevents oversizing procedures currently used to account for sensor inaccuracy. Used during commissioning or operation.	Outside air intakes can be controlled optimally to balance energy and health.
Dynamic feedback from airflow sensors	Anemometer	reduces energy input in data centers required to maintain servers at required temperature	
improve safety and reduce the over- or under-pressure margin required in clean rooms and hospitals.	Anemometer	Low-cost anemometers positioned in the air handling systems of hospitals, laboratories. Direct flow measurement in ducts, fume hoods, and the occupied	enable more responsive system control, and more immediate safety alarms should airflows be reversed.

Table 10: Potential applications using Hamilton

There are three kinds of application categories for which the Hamilton is well positioned to support. they are: embedded in other hardware or devices, for enabling prototyping of future hardware or devices, and as a hardware-in-the-loop control (HIL) sensor.

Company	Company Type	Application	Application Category
Sage Glass	Dynamic fenestration manufacturer	building recommissioning	Sensing (HIL)
Perkins and Will	Architecture Firm	environmental monitoring	Sensing (HIL)
Integral Group	Engineering Firm	environmental monitoring	Sensing (HIL)
LPA inc	Architecture Firm	environmental monitoring	Sensing (HIL)
Armstrong World Industries	Building components and materials manufacturer	building recommissioning	Sensing (HIL)
Ingersoll Rand	Building components manufacturer	product development	Prototyping
Genentec	Pharmaceutical manufacturing	building recommissioning	Sensing (HIL)
Price Laboratory	Device manufacturing	product development	Embedded
Alsen	Device manufacturing	occupant-centered HVAC products	Prototyping
PingThings	Utility scale monitoring equipment and data collection	(BTrDB backend, w/ syncrophasor) pilots with PGE	Prototyping, Embedded
Chirp Micro Systems	Device manufacturing	ultrasonic anemometer	Embedded
Building Robotics	Software based building controls	connecting Hamilton technology with COMFY offerings	Sensing (HIL)
BuildingOS	Building energy dashboard developer	connecting Hamilton technology with BuildigOS offerings	Sensing (HIL)
Hamony.ai	Building commissioning	re-segment the building commissioning market	Sensing (HIL)
Intel	IoT and metadata acquisition	low cost insertion of our technology into buildings	Sensing (HIL), Embedded

Table 11: More likely applications using Hamilton

We identified three potential building products in this report; of these the anemometer has the most likelihood of commercialization. CBE has been in communication with Davis Instruments in Hayward, California and Price Industries in Toronto who are both very interested in the applications (outdoor air speed and duct air speed respectively).

Next Steps in addressing barriers

The stand-along deployments have had a number of lessons-learned that will be valuable in exploring future building product incorporation. We learned that the Hamilton installation requires a highly reliable internet connection and that each border router needs to be in the line of site of a sensor or with little to no obstruction. This line-of-sight should not exceed 30 feet. This makes the installation of the stand-alone Hamiltons more ideal for large, open spaces such as open office plans, auditoriums, or lecture or performance halls. These types of spaces allow for the benefits of the mesh network to be used to its full advantage. In the XBOS-DR program where we explored the installation of Hamiltons in different buildings, it was determined that the physical layout of the buildings themselves made this mesh network difficult to achieve as the rooms were compartmentalized with solid partitions and doors preventing a small number of routers from serving a larger volume of Hamiltons. In addition, having to supply multiple border routers in buildings with compartmentalized spaces required both additional expense and available network ports for the border routers. The new networking scheme may help these issues.

The Center for the Built Environment (CBE) semi-annual Industry Advisory Board Conference has been used as a platform for outreach to over 100 industrial partners from leading engineering, architecture, and buildings-related manufacturers. This outreach resulted in numerous requests to receive Hamilton kits. These kits will be used by a variety of companies representing traditional architecture and engineering, software development, building controls manufacturing, and building component manufacturing. Some of these companies will be using the Hamilton-based technology in their own facilities for the purpose of retro-commissioning of their operations, some will be developing new technology with Hamilton-based devices embedded, while others are in the prototyping phase for new products. These products lend themselves to the continuation of the current applications of the Hamiltons, where they are use in FDD, commissioning, and sensing, among other applications.

Value Chain of products

We received many inquiries of people wanted to purchase the Hamilton kits (sensors and border router) both in research and industry. These provide a starting point for broader market discovery. In addition to these engagements, we have started a series of deeper engagements with industry partners representing a spectrum of go-to-market strategies. We are also developing kits that would allow research and pilots to explore potential market segments. Collaborator Prof. Jeong Gil Ko is investigating its use in motion sensing from ICU beds at the Ajou University Hospital Trauma Center.

Budget Period 2 Significant findings, conclusions, or developments.

Hardware developments: There is an increasing array of low-cost plug load sensors on the market, so while we developed a novel, low-cost one with Hamilton, we elected not pursue further development or commercialization. Instead, we leverage commercial designs for building application demonstrations and focused hardware design effort on developing a miniature low-cost powered router and several specialized Hamilton-based sensor system designs.

Software developments: We worked with RIOT-OS and pushed low-power embedded operating systems developments upstream, as well as new multihop routing protocols (H. S. Kim et al., 2018) and implementations of OpenThread. The core RIOT development and test group has received Hamilton sensors and is integrating them into their test facilities, so the Hamilton platform is a required test before new code is released. We used OpenThread to develop multi-hop communication.

Applications: We developed and tested MPC and demand response HVAC control algorithms in three buildings; these were tested in 16 commercial buildings.

Path to Market: In December 2017, HamiltonIOT was created to provide data services and sensors to the public. Partner Chirp Microsystem has now been acquired by TDK and partner

Comfy was just acquired by Siemens, so collaborations with Hamilton extend from start-ups into well established forms in the building technology industry.

Hamilton sensors have been deployed in multiple test beds: a four multifamily housing sites for the EPIC fans project to optimize smart ceiling fans and thermostats, and several of the 16 XBOS-DR project buildings.

Brick schema and the consortium is documented at http://brickschema.org/, including white papers and tutorials found there. In February 2018, the ASHRAE BACnet committee announced that their partnering with Project Haystack and the Brick initiative to integrate tagging and data modeling into ASHRAE Standard 223P (https://www.automation.com/automation-news/industry/ashrae-bacnet-committee-project-haystack-and-the-brick-initiative-partner-to-integrate-tagging-and-data-modeling-into-ashrae-standard-223p).

Budget Period 2 Go/No-Go

Metric 1: *Functional application components with secure middleware agent. Demonstrate 5 distinct application software components with secure middleware agent materialized by 20 instantiations of those components comprising 10 controllers, 5 schedulers, a continuous metadata query processor, a time-series data management processor, and 3 energy analytics processes operating on an aggregate flow of 5,000 resources.*

We have demonstrated 19 distinct application software components (e.g., BTrDB, MDAL, HodDB, Pundat, Spawnpoint). We have deployed 35 instantiations of these components including:

- 16 controllers
- 2 pure schedulers and 10+ controller/schedulers
- 11 energy analytics processors
- A time series query processor
- A metadata query processor

These instantiations of components include the buildings within the XBOS-DR project. We are controlling between 5-24 Roof Top Units at each of these buildings; a few buildings include lights, plugloads, and EV charging. All buildings have an associated Brick model stored in a HodDB database (Gabe Fierro, 2017). The corresponding timeseries data is stored in a BTrDB database cluster. Energy analytics applications access historical and real-time data by presenting Brick queries to the MDAL service. We have several energy analytics processing operating on these buildings.

These instantiations operate over a total (combined with the resources associated with metric 3) aggregate flow of 41,262 resource streams. The time series archiver has stored 13,236,921,268 data points.

Metric 2: *Functional border router integrating sensor network with middleware routing traffic from 50 sensor systems each reporting every 5 minutes with less than 1% loss rate through the router.*

We have an evaluation testbed on the fourth floor of Sutardja Dai Hall on the UC Berkeley campus. This testbed has 14 nodes (border routers) and 25 sensors. We have evaluated remote reprogramming and remote monitoring protocol operation and performance metrics. When each of 14 nodes sends a packet every 30 seconds to the one border router, we get 2.76% of loss rate with 250 kbps data rate and 0.42% of loss rate with 2 Mbps data rate. We conducted the experiments when the wireless channel is very noisy. We integrated the OpenThread protocol stack into the border router and checked that this delivers traffic between sensors and servers as a bridge. We are making an OpenThread-enabled border router usable out of box (without configuration), as we did with the previous Hamilton border router.

We have deployed and run more than 50 sensors reporting every 20 seconds in a single hop configuration with less than 1% loss, **exceeding the goal.** Our aspirational goal is to support >50 sensors in a **multi-hop** configuration using TCP to obtain loss < 1%, enabling a physically larger network, and we were able to achieve this in a lab setting.

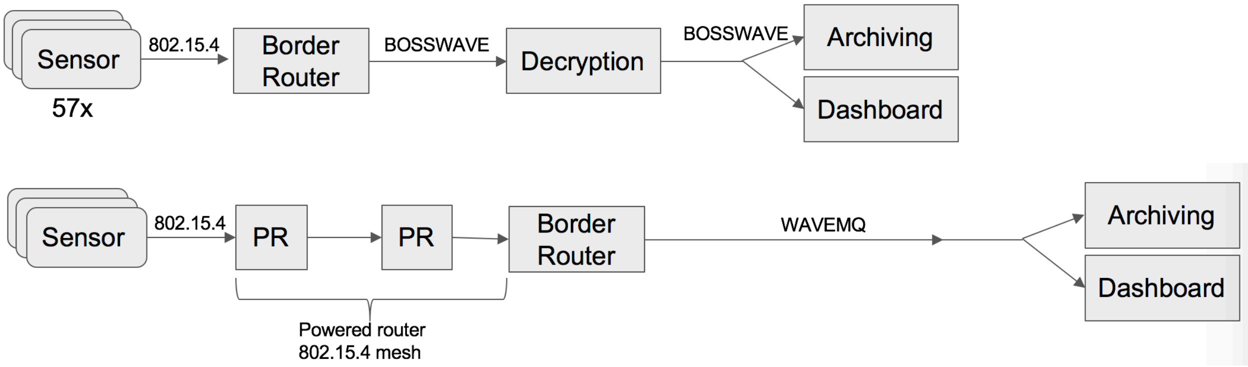


Figure 53: Top represents the goal we achieved; bottom figure represented our next goal using WAVEMQ.

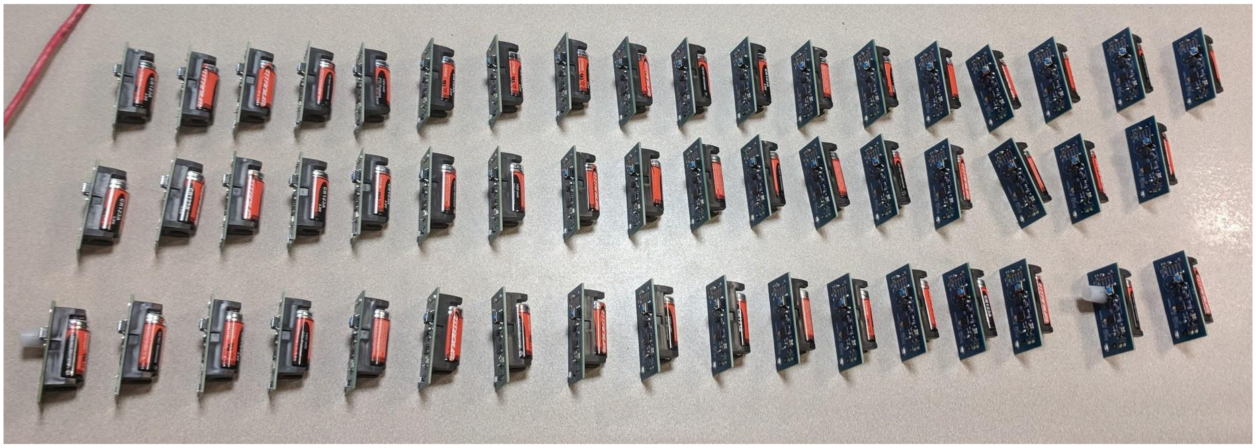


Figure 54: greater than 50 Hamilton sensors in a multihop mesh network with 20 second sampling and using WAVEMQ on the border router.

<u>Metric 3</u>: *Functional proxy and driver components for local API and vendor cloud API devices, including BMS and commercial products. Demonstrate 5 distinct proxy and 5 driver software* *components agents materialized by 20 instantiations, including 1 BACnet-based BMS system, producing 5,000 resources with access granularity at multiple levels ranging from individual sense points to large collections.*

We have constructed and deployed several proxy and driver components:

- We have 6 proxy components (Rainforest Eagle smart meter, Enphase, Pelican thermostat, National Weather Service, BACNET, Juiceplug level 1 EV charger)
- We have 9 driver components (TED, TED3, EMU2, Enlighted, Proliphyx Thermostat Imt550c, Venstar thermostat, Lifx light bulbs, Tp link plug monitor, Aerovironment level 2 EV charger)
- Proxy components instantiated: 21
- Driver components instantiated: 19
- We have instantiated 3 BACNET-interfacing components

Resources for these components are not distinguished from those in metric 1:
There is a total aggregate flow of 41,262 resource streams

As mentioned above, we have developed and tested WAVE drivers for three thermostats, two lighting lighting controllers, and two plugload monitors. In addition, we have WAVE drivers for an energy monitor, Weather Underground weather data, interval meter gateway, Photovoltaic gateway, and two Electric Vehicle chargers. At least 20 of the buildings in the XBOS project have connected thermostats and energy meters. One of our BACnet buildings (Orinda Library) has more than 4500 points and the other, SDH, has more than 2800 points. We developed the BACnet-based BMS driver for these two buildings.

Budget Period 3

Task 9: Generation 3 Low-Cost Wireless Sensor Systems

Design study of Sensor System utilizing industry advances

A review of the market drove several component changes for the Generation 3 of Hamilton. For one, there are global manufacturing shortages of the NXP Semiconductor FXOS8700CQR1 Series 6-Axis sensor with accelerometer and magnetometer.

Feedback from users of Hamilton was that the original light sensor was insufficiently accurate for indoor comfort applications.

Several deployments of Hamilton sensors in campus buildings and in the field allowed an assessment of the technology. In two field tests, the communication to the border router sometimes failed. In one case this was due to the distance between the sensor and the router.

There are multiple WiFi / Bluetooth IoT modules now hitting reasonable price points, with a number of (low grade) product offerings based on them. There is an active open source community for ESC-32, but the power question is still unclear. Cheap sensors from China (e.g., Espressif ESP32-WROOM-32 MCU Modules) still pull current in the mA range where Hamilton has achieved uA.

Proof of principle design and implementation

Generation 3 has switched to the LSM 303C accelerometer and magnetometer sensor, which has comparable price and precision. The Generation 3 sensor has switched to a digital illuminance sensor – the ISL 29035, which promises increased accuracy and 120Hz flicker rejection.

Hamilton is part of RIOT-OS's main branch (took 10 months), which means RIOT maintainers manage Hamilton code as RIOT is updated.

Our paper on OpenThread (H. S. Kim et al., 2019) was accepted to IEEE Communications Magazine, in a special topic about Future Internet Architecture. This is the first introduction of Thread to academia; our methodology includes a qualitative comparison between RPL vs. Thread and a quantitative analysis by way of a testbed evaluation. A summary of the comparison (RPL vs. Thread) is shown in Table 11 below. RPL is represented by academia, has just the L3 routing spec, can support thousands of nodes, and has upward focused routing, whereas Thread is represented by industry, has a complete routing spec (L1-L3), supports hundreds of nodes, and has all-to-all direct routing. The qualitative comparison includes RPL's flexibility (ambiguity) vs. Thread's specific guidance (restriction), and issues regarding security, network partition/merge, address, link cost metric, and radio duty-cycling. We find that testing and analyzing each function of OpenThread has academic value, suggesting where we should go (or should not go).

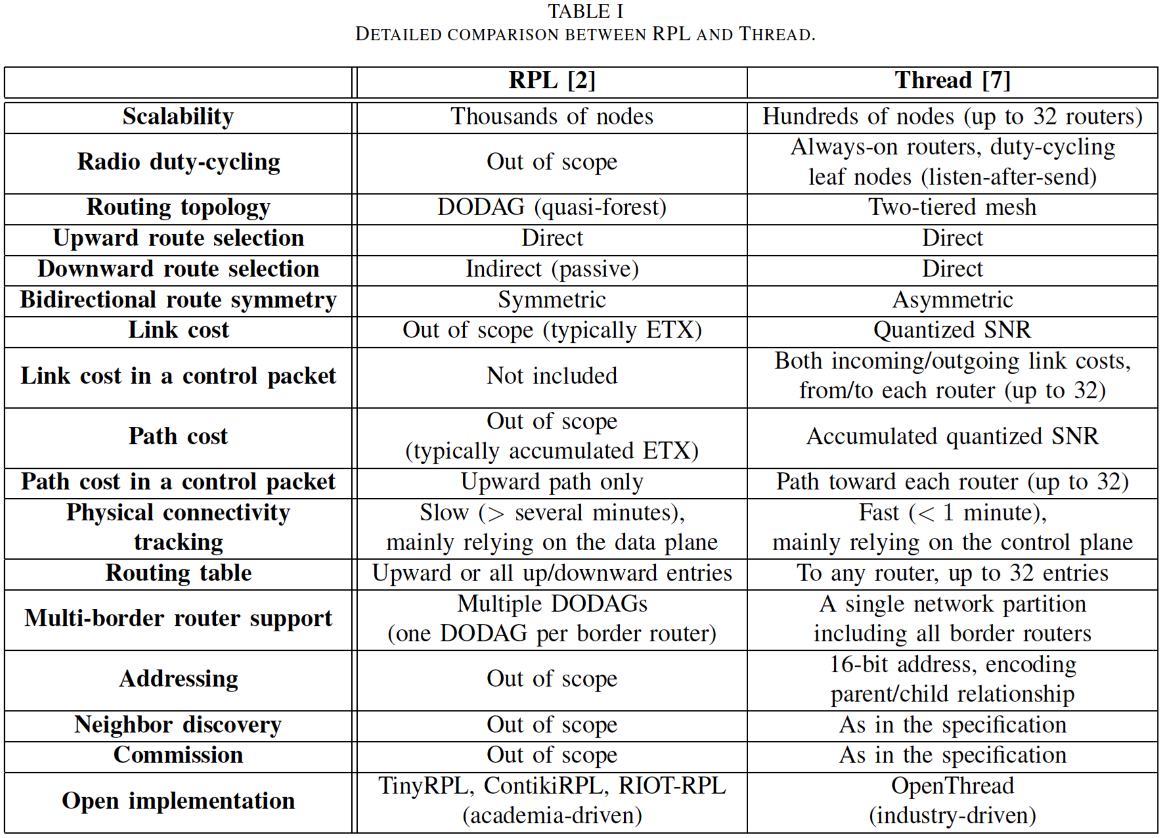


TABLE I
Detailed comparison between RPL and Thread.

	RPL [2]	Thread [7]
Scalability	Thousands of nodes	Hundreds of nodes (up to 32 routers)
Radio duty-cycling	Out of scope	Always-on routers, duty-cycling leaf nodes (listen-after-send)
Routing topology	DODAG (quasi-forest)	Two-tiered mesh
Upward route selection	Direct	Direct
Downward route selection	Indirect (passive)	Direct
Bidirectional route symmetry	Symmetric	Asymmetric
Link cost	Out of scope (typically ETX)	Quantized SNR
Link cost in a control packet	Not included	Both incoming/outgoing link costs, from/to each router (up to 32)
Path cost	Out of scope (typically accumulated ETX)	Accumulated quantized SNR
Path cost in a control packet	Upward path only	Path toward each router (up to 32)
Physical connectivity tracking	Slow (> several minutes), mainly relying on the data plane	Fast (< 1 minute), mainly relying on the control plane
Routing table	Upward or all up/downward entries	To any router, up to 32 entries
Multi-border router support	Multiple DODAGs (one DODAG per border router)	A single network partition including all border routers
Addressing	Out of scope	16-bit address, encoding parent/child relationship
Neighbor discovery	Out of scope	As in the specification
Commission	Out of scope	As in the specification
Open implementation	TinyRPL, ContikiRPL, RIOT-RPL (academia-driven)	OpenThread (industry-driven)

Table 12: RPL vs Thread comparison (H. S. Kim et al., 2017)

The figure below shows the layout of the nodes in the test.

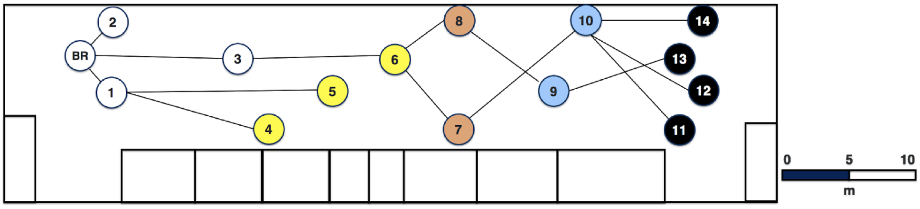


Figure 55: Plan of topology of sensors, distant from the border router (BR).

The figure below shows the results of the quantitative analysis. As traffic load increases, network loses more data packets (left); however, the significant packet loss does not diminish routing since Thread's path cost is not packet delivery ratio but RSSI (Received Signal Strength Indicator).

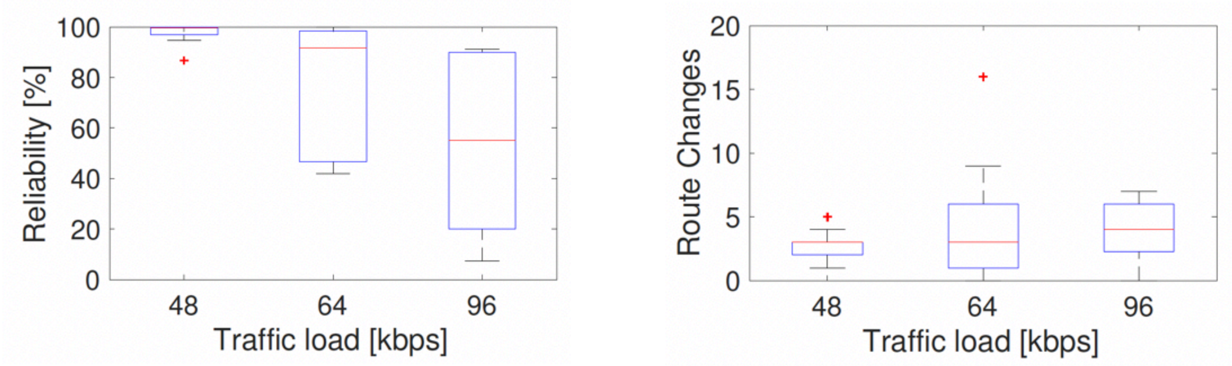


Figure 56: Results of the quantitative analysis of OpenThread, showing data packet loss with more traffic (left), but with few changes to routing (right).

We find that OpenThread's L2 implementation (IEEE 802.15.4-2006) has room for improvement, as the system experienced nontrivial packet loss in the daytime (e.g., Time synchronized Frequency hopping (IEEE 802.15.4-2015), and BLE). Having TCP (a reliable protocol at L4) improves OpenThread's end-to-end data delivery performance.

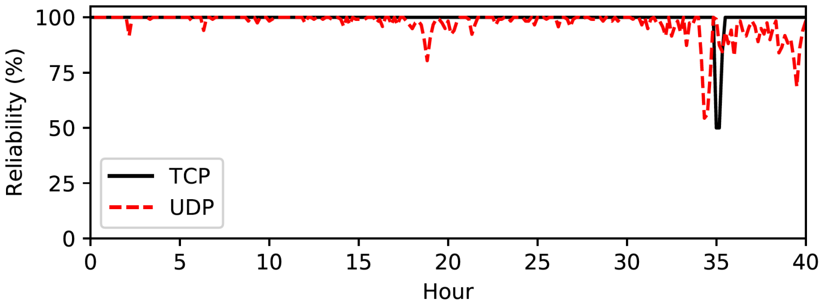


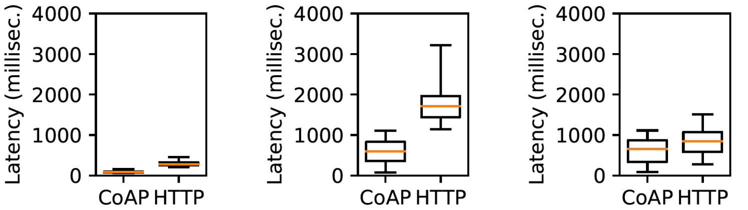
Figure 57: Results of the quantitative analysis of OpenThread, showing data packet loss at different times of the day.

Scaling Study

Milestone 11.1: Building Scale proof of sensor network tier integrated with BMS and other devices. (M30)

We have described the development of TCPlp, OpenThread, and RIOT, as we move away from the first generation border router single hop approach to a multi-hop network. We conducted basic tests using the border router on the second-generation Hamilton sensors to ensure proof of concept.

Regarding TCPlp advances, we developed new techniques to run TCP over *duty-cycled* links and gained insight into behavior with simultaneous flows (compared to CoAP). Our testing revealed that the self-clocking behavior of TCP interacts poorly with a duty-cycled link (see Figure 59b). The solution was an adaptive duty cycle: shorten the duty cycle when round-trips are expected (Figure 59c).



(a) No duty cycling (b) 1 s sleep interval (c) 1 s sleep interval with adaptive duty cycle

Figure 58: Latency of web request: CoAP vs. HTTP/TCP

Regarding behavior with simultaneous flows, we examined the situation where many flows compete for bandwidth: is sharing fair? *TCPlp* performs similarly to existing state-of-the-art in Low-Power and Lossy Networks (LLNs) (Figure 60).

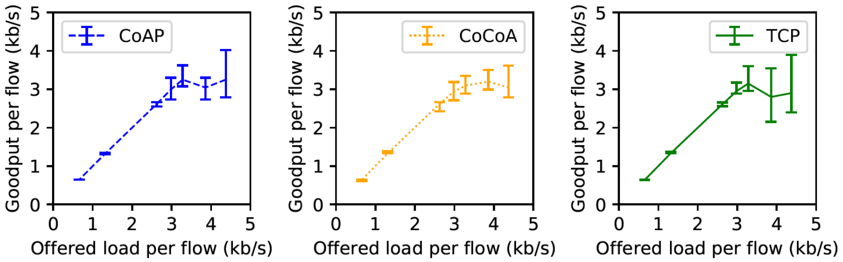


Figure 59: Performance of CoAP, CoCoA, and TCP with four competing flows.

We prepared a large-scale, multi-hop deployment on the fourth floor of Soda Hall, which underwent a renovation in the last several months. The ceiling has Power over Ethernet (POE) cables to power the border routers (Figure 61). The boards are available.

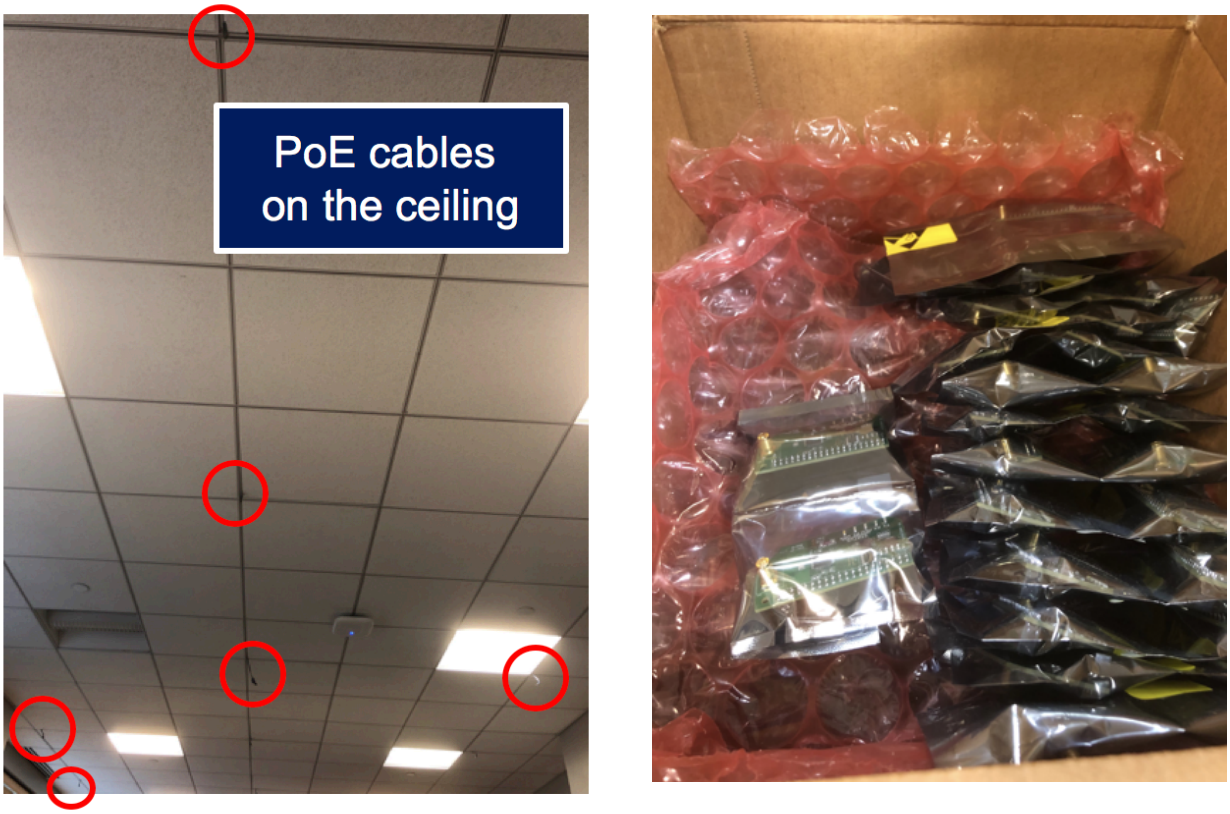


Figure 60: Getting ready to test the multi-hop network. Left: Power over Ethernet cables in the ceiling of the 4th floor of Soda Hall awaiting the routers. Right: boards have arrived.

Previously we have shown the integration of Hamilton sensors used in conjunction with thermostat control. Lab testing is critical to test the performance of the communication in the presence of typical interference; simulation cannot regenerate channel dynamics in the real world (e.g., communication disruption due to human activity, other interference). Having a large-scale testbed facilitates networking research by providing convincing results. Surprisingly, there are only a few of this type of open testbeds now, even more so when it comes to modern 32-bit platforms. Examples are the testbed at the Graz Technical University, University of Southern California, ETH Zurich, Inria, and National University of Singapore (Figure 62).

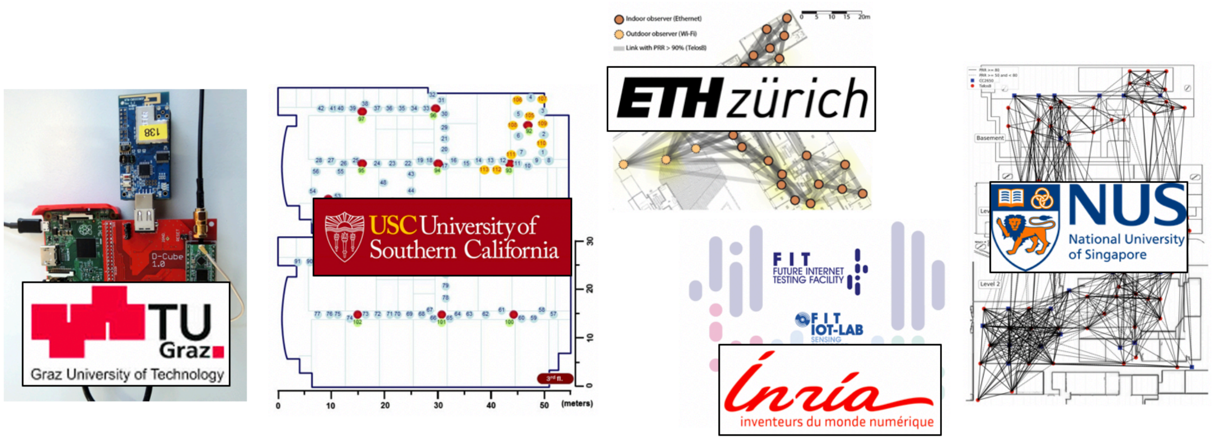


Figure 61: Examples of large open sensor testbeds at other institutions.

The architecture of the Hamilton Border Router is shown in Figure 63, with a close-up photo to the upper left. The Hamilton Border Router unit is connected to the pins of the Raspberry Pi miniature computer running Linux. The Raspberry Pi is connected via Ethernet to the network for data transfer to the cloud.

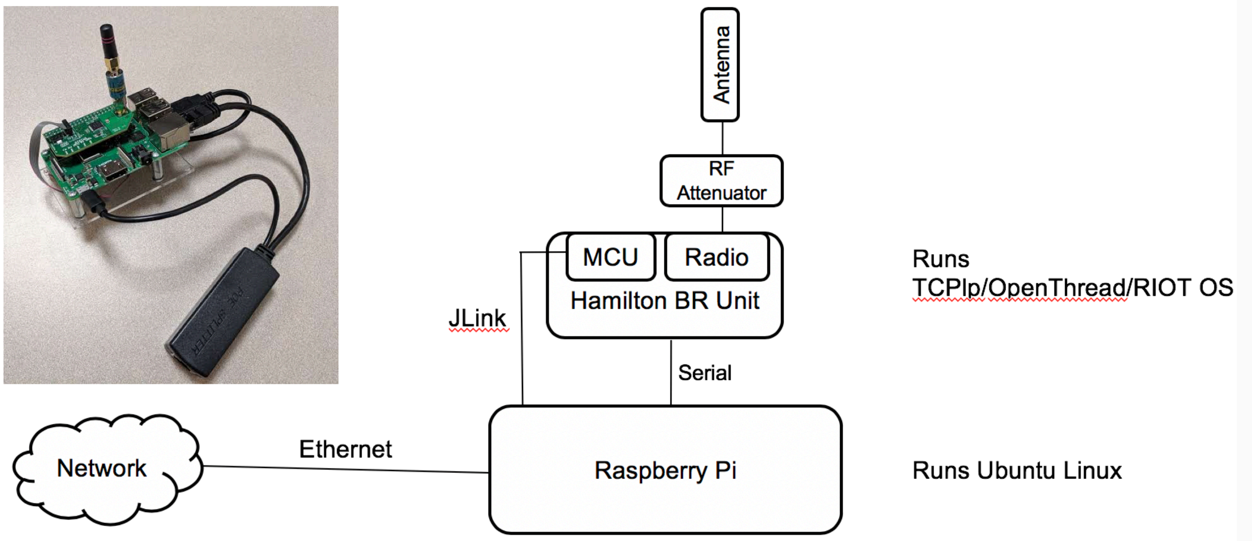


Figure 62: Hamilton Border Router is based on Raspberry Pi miniature computer, connected to the Hamilton unit with antenna.

In an office in Soda Hall, we installed 60 border routers (Figure 64).

Figure 63: Hamilton Border Routers installed in ceiling in office in Soda Hall, UC Berkeley.

Task 10: Secure Middleware

BOSSwave has moved away from its original blockchain structure and has been renamed WAVE. We continue to improve the decentralized authorization framework. The advances include:

- New cryptographic curve offering faster operations (and higher security level)
- Implementation in optimized ASM for embedded devices
- Extensive benchmarks where the embedded device does all the crypto

We ran these new scenarios on Hamilton compared to industry standard Advanced Encryption Standard (AES). The per-Hour Latencies are similar: 6.5 s (encrypt only) compared to 9.9 s (encrypt and sign). The “fancy crypto” for end-to-end encryption still provides several years of battery life (Figure 65).

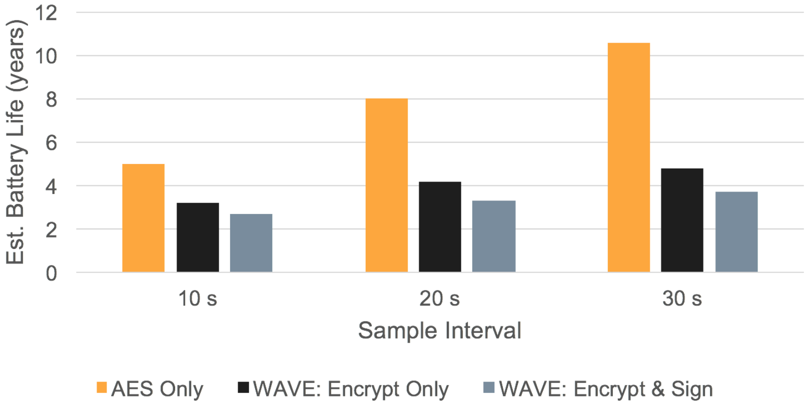


Figure 64: Latencies using AES compared to WAVE

A major effort was developing the third generation of BOSSwave, renamed WAVE 3 as the decentralized Authorization Framework. We redesigned a reimplementation for crypto-based protection of Delegation of Trust (DoT) graph and scalable untrusted storage tier. We developed JEDI (Kumar et al., 2019), the end-to-end (E2E) encryption protocol compatible with WAVE that is feasible even on the Hamilton platform. JEDI provides encryption support in WAVE 3 (and therefore WAVEMQ). JEDI utilizes the authorization discovery process to enable E2E encryption / decryption, and allows publish/subscribe multiparty confidentiality without preshared keys. Benchmarks using JEDI (Figure 66) show a low overhead in usual case, a single-digit millisecond overhead for first message after key rotation (on laptop), and feasibility on low-power embedded devices.

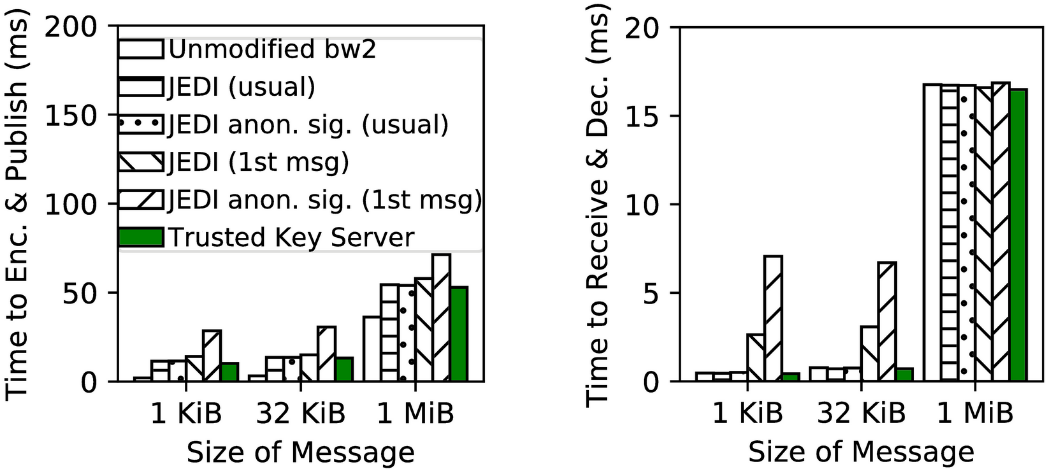


Figure 65: JEDI Benchmarks

We developed WAVEMQ, a publish-subscribe system using both WAVE and JEDI. WAVE remains the backbone of secure information flow among sensors and building systems (Figure 5). We presented two papers, one on WAVE (Andersen et al., 2019) and one on JEDI (Kumar et al., 2019) at USENIX Security 2019.

The latest development (WAVEMQ) allows local caching and management, but requires rewriting all the drivers for devices and services, including drivers that interface with the building controls, such as thermostats and BAS. We migrated the WAVE 2 drivers and services to WAVEMQ. WAVEMQ will help with continuous monitoring and telemetry ingestion.

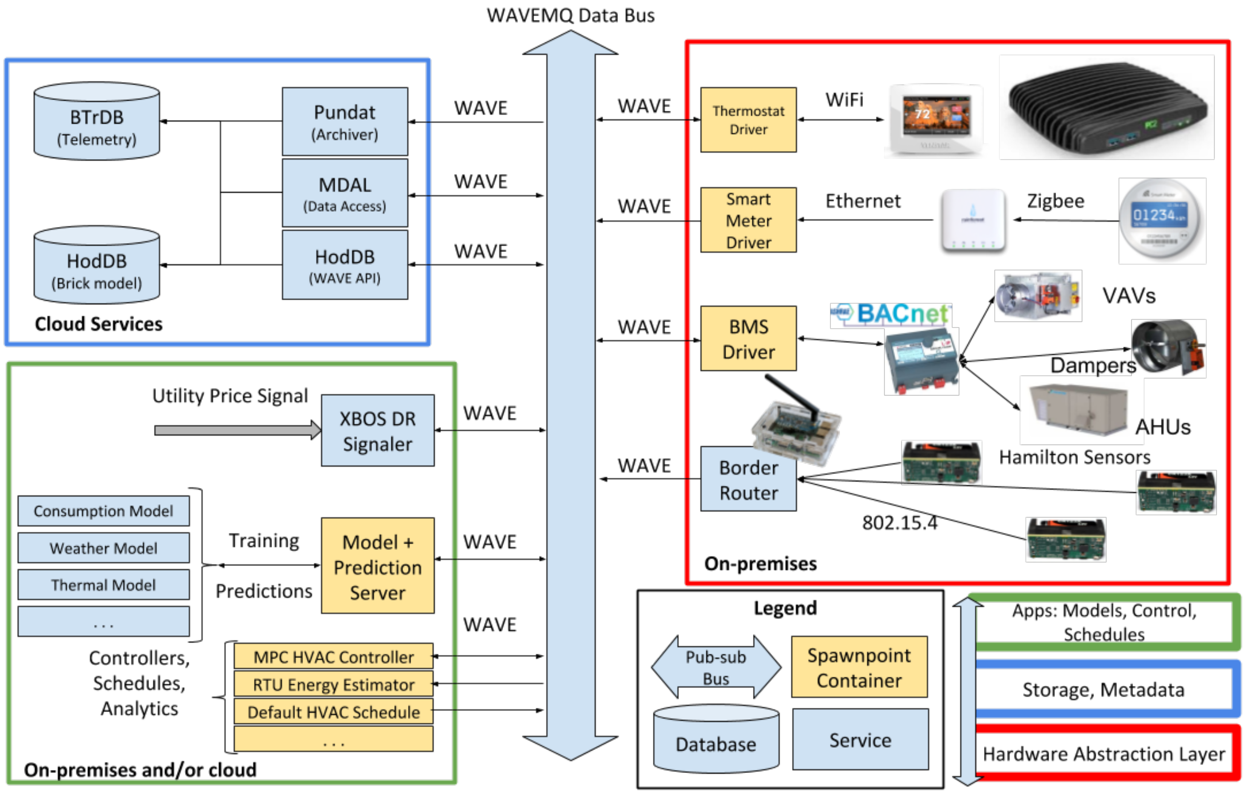


Figure 66: The new WAVEMQ secure data bus.

Local caching includes adding multiple caching layers to WAVEMQ (our tiered syndication engine):

- Cached proof building
- Cached proof transmission
- Cached proof verification

This allows the bandwidth and latency characteristics of a fully WAVE-augmented pub/sub system to be very close to those of a standard pub/sub system. The figure below shows a schematic of the local versus remote data consumers using pub-sub.

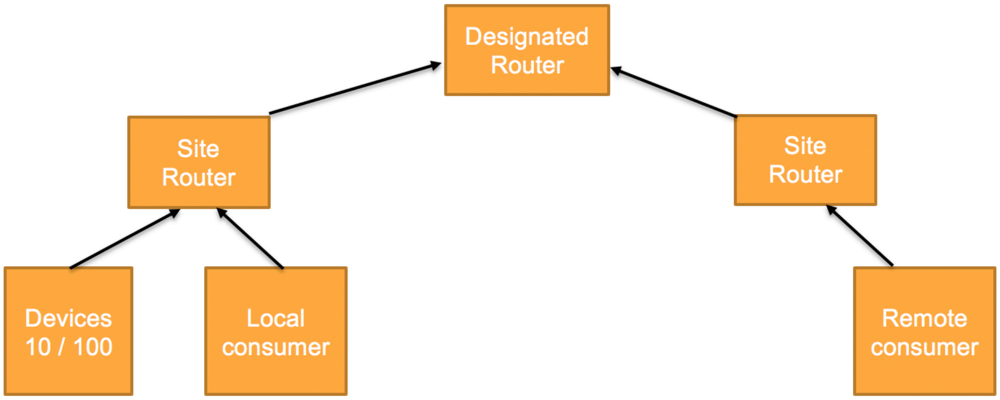


Figure 67: The architecture for testing WAVE-MQ.

The graph below shows the effectiveness of the caching.

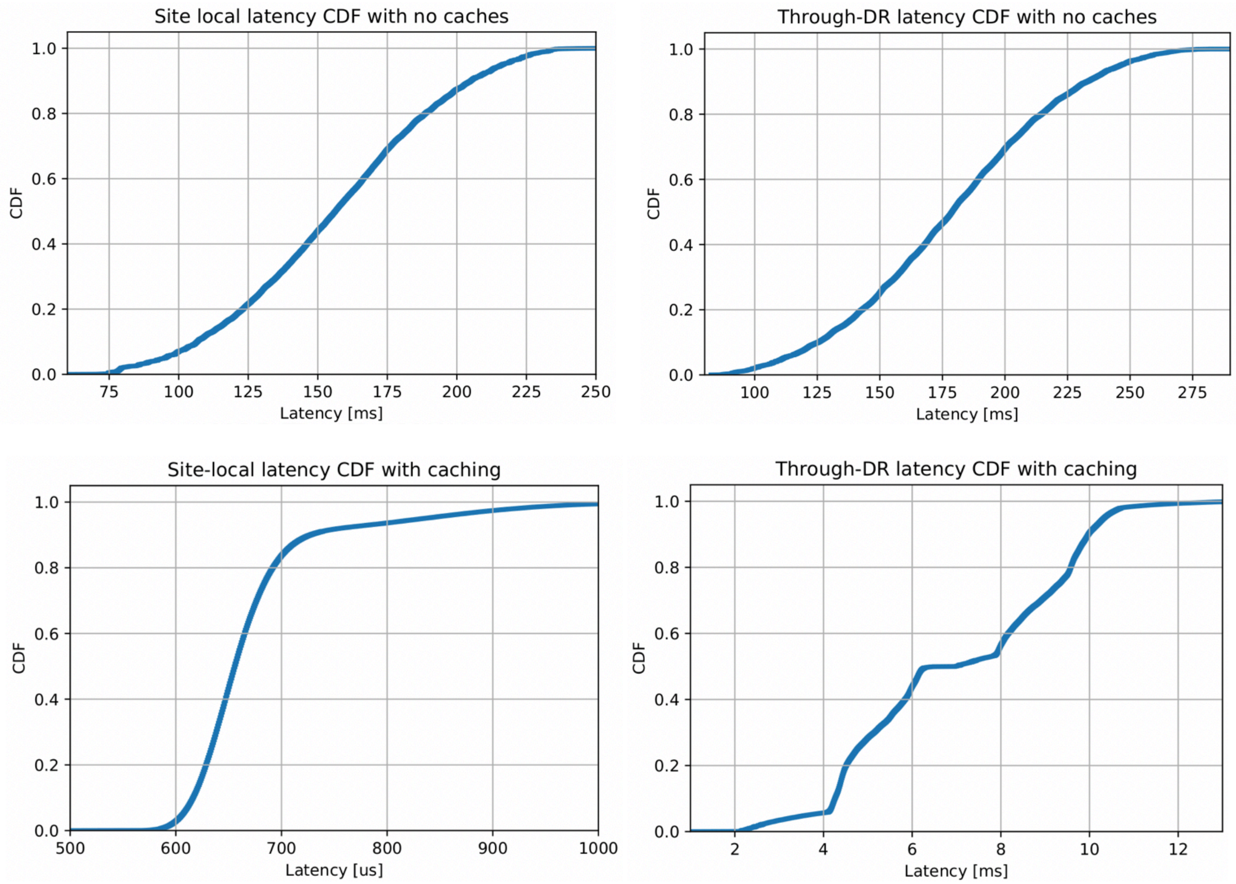


Figure 68: The graphs show the Cumulative Distribution Function (CDF) of the latency. The top two graphs show the latency with no caching; the bottom two graphs show the latency (with 10 times the load) with caching.

We published a paper on the performant TCP work (Kumar et al., 2020). In general, the low power networking community has not considered supporting a full-scale TCP stack. But with modern low power networks, the expected problem with TRC does not apply and TCP works as well as protocols specialized for the embedded setting.

As we reflect on the evolving industrial landscape, we find there are many wireless protocols; we are reaching a point where either IEEE 802.15.4 (LoWPAN) will either go mainstream for IoT Internet connectivity, or it fades away (after 16 years of slow development) and 802.11 (WiFi) – with all its challenges and shortcomings – becomes the only avenue for IoT. OpenThread,[4] the BSD-licensed open-source low-power mesh networking technology that is IPv6-based has gained traction since its release by Google Nest. Figure 70 shows the system architecture of OpenThread and Figure 71 shows many companies that support OpenThread.

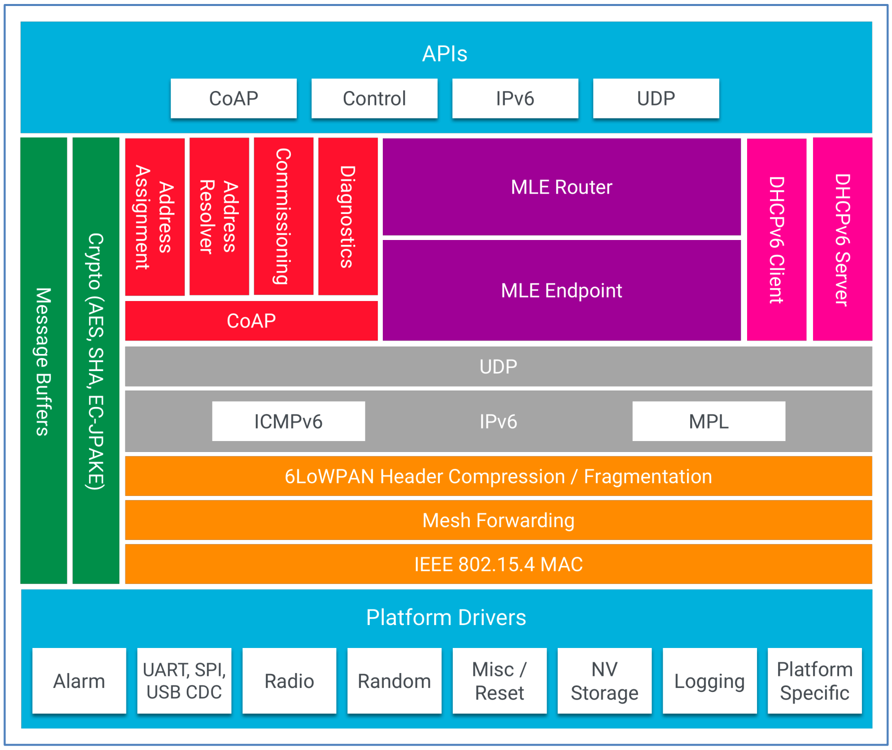


Figure 69: The architecture of OpenThread

---

[4] https://openthread.io/

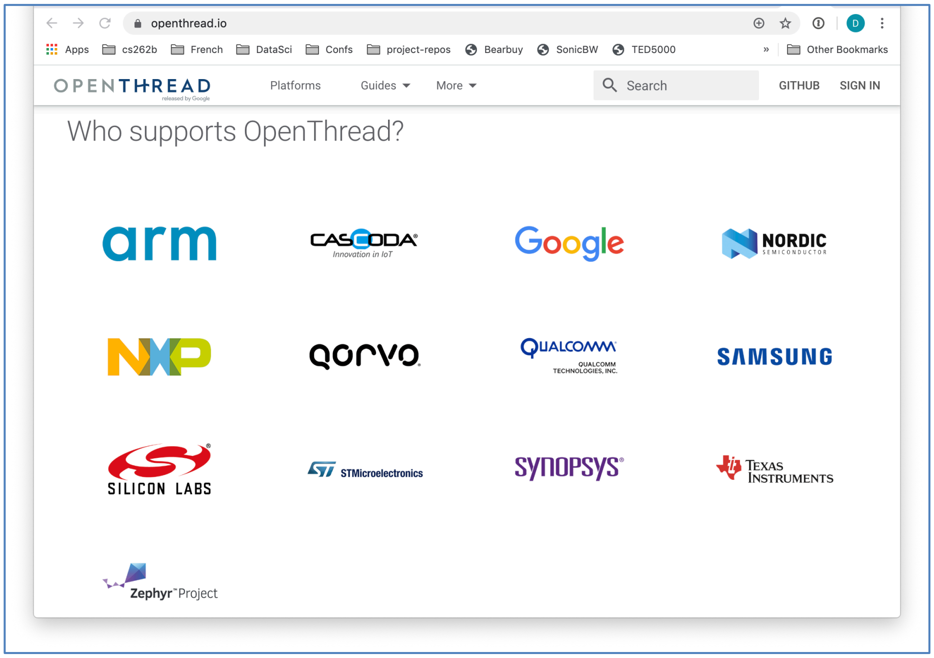


Figure 70: Companies supporting OpenThread

The team has explored potential for TCP (Kumar, Andersen, Andersen, et al., 2018; Kumar, Andersen, Kim, et al., 2018). The Constrained Application Protocol (CoAP) was developed as a light-weight protocol for the many small devices that are resource-constrained (low power, lossy), especially as a replacement for HTTP, which has high computation complexity, low data rate and high energy consumption.[5] CoAP uses UDP, which is unreliable, and not TCP, which is considered complex; some felt that CoAP might replace TCP. However, the latest OpenThread release has a simplified TCP implementation: "OpenThread RTOS (OT RTOS) provides both system-level and application support for connecting a device to a Thread network and the internet. OT RTOS integrates OpenThread and LwIP (a small independent implementation of the TCP/IP protocol suite) into a single platform solution."[6] OT RTOS includes application-layer demo commands for the HTTP, MQTT, and TCP protocols. This makes 802.15.4 a viable competitor to 802.11.

There is potential for full-scale TCP to gain industry adoption. (Kumar et al., 2020) shows empirically that TCP can be made to perform well in sensor networks and argues that the *interoperability* afforded by a TCP-based architecture would bring substantial benefits (puts the I in IoT). The paper and code are now publicly available – we have a close engagement with the Google NEST team and others.

---

[5] https://www.cse.wustl.edu/~jain/cse574-14/ftp/coap/

[6] https://openthread.io/platforms/rtos/ot-rtos

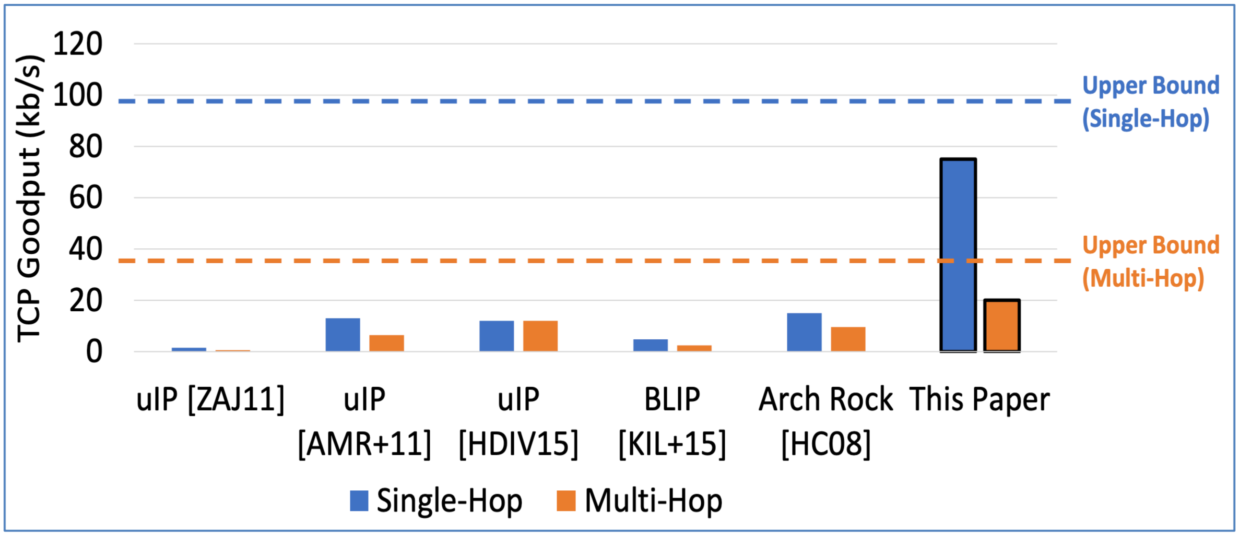


Figure 71: Performance of TCP in sensor networks (Kumar et al., 2020)

A new development in WAVE is in secure multiparty computation. WAVE supports authorization across multiple administrative domains, but authorization domains need to cooperate in more complex ways than just accessing resources: they need to provide services.

Figure 72: Secure data from service

Secure Multi-Party Computation (SMPC) allows two (or more) parties to jointly compute any function over their private inputs, without revealing these private inputs.

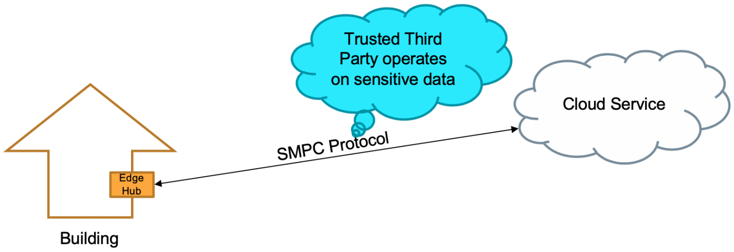


Figure 73: Secure Multi-Party Computation

The current challenge is the overhead:

- CPU time: ~10,000x slower than computing on plaintext
- Takes a lot of memory: 128x blowup over plaintext

- 32 bytes of network I/O per *bit* of input

This led to MAGE , an Efficient Execution Engine for SMPC and a holistic approach to SMPC execution. We have achieved zero-cost virtual memory abstraction for garbled circuits.

Authenticated External Entity for DR notification

The microservices (described below) include a microservice that provides actual event signals delivered from two different utilities: PG&E’s Peak Day Pricing events and SCE’s Critical Peak Pricing events.

Recently another project (CEC-funded Solar+, primed by Humboldt State University with LBNL as a subcontractor) used the XBOS platform with WAVEMQ in their demonstration project at a convenience store on a reservation in northern California. Using the WAVEMQ tiered message bus was vital to the project due to the ability to operate in the event of network outages.

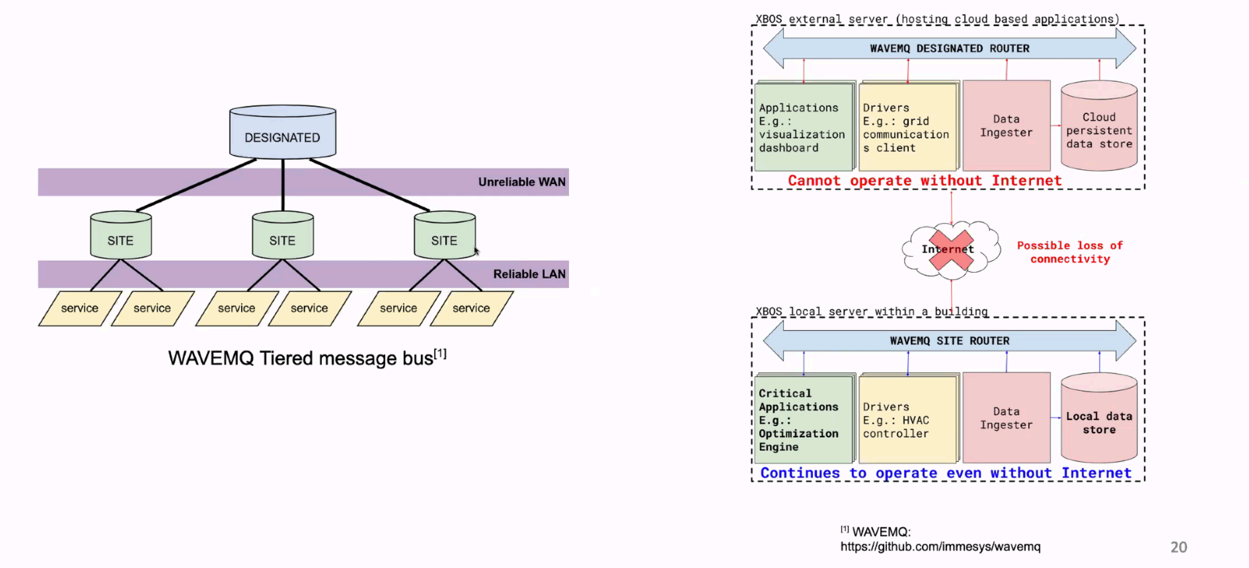


Figure 74: Solar+ Project configuration using WAVEMQ tiered message bus.

Milestone 12.1: Demonstrate less than 20% increase in latency during onset of simulated attack at 10% of authorized traffic. (M27)

One of the goals of the security system is to resist attacks. Embedded IoT-class devices cannot be over-provisioned with excess CPU/Memory; this makes them especially vulnerable to Denial-of-Service (DOS)[7] attacks. Typical communication patterns (e.g., Constrained

---

[7] In computing, a denial-of-service attack (DoS attack) is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. From Wikipedia https://en.wikipedia.org/wiki/Denial-of-service_attack

Application Protocol or CoAP[8]) complicate firewall configuration (Figure 76).

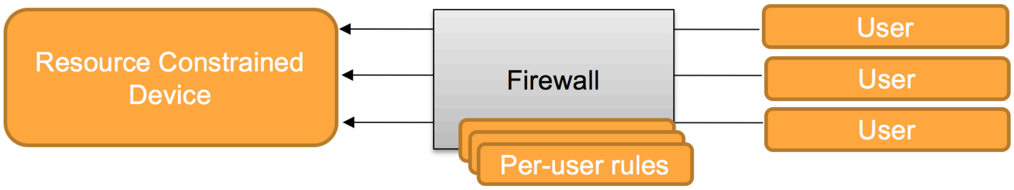


Figure 75: CoAP firewall configuration

The management overhead leads to overly-permissive firewall rules (Figure 77).

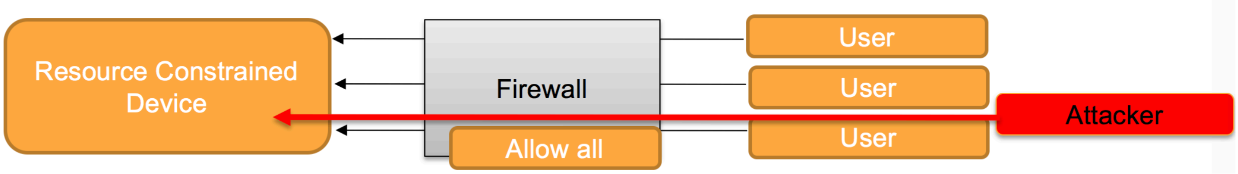


Figure 76: Overly-permissive firewall rules

The solution is indicated in Figure 78. Secure syndication fabric simplifies upstream firewall configuration, increasing DoS resistance – as long as the syndication fabric itself is resistant.

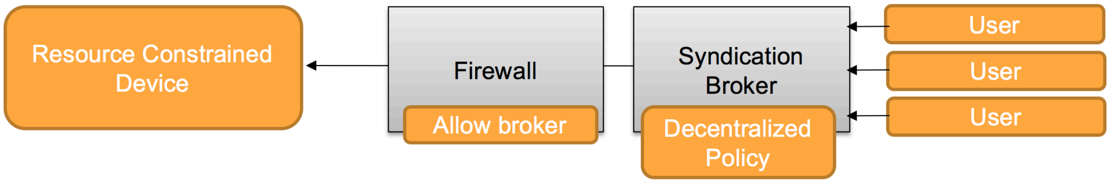


Figure 77: Secure syndication fabric

This milestone is verifying that the syndication broker is unaffected by attack (Figure 79).

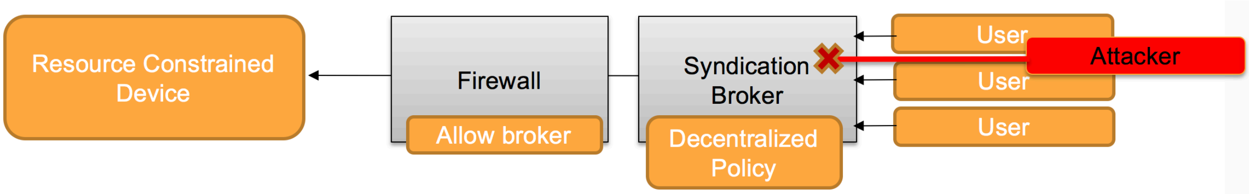


Figure 78: Syndication broker unaffected by attack.

We developed an experimental setup with a transmitting and receiving device, site routers and a designated router, to demonstrate the Milestone (Figure 80 left). The experiment would measure

---

[8] https://en.wikipedia.org/wiki/Constrained_Application_Protocol

end-to-end latency distribution from one site to another, with a rate of 1200 messages per minute (msg/m). We contrast this with a measurement distribution under attack (Figure 80 right) with a rate of 600 msg/m (50%).

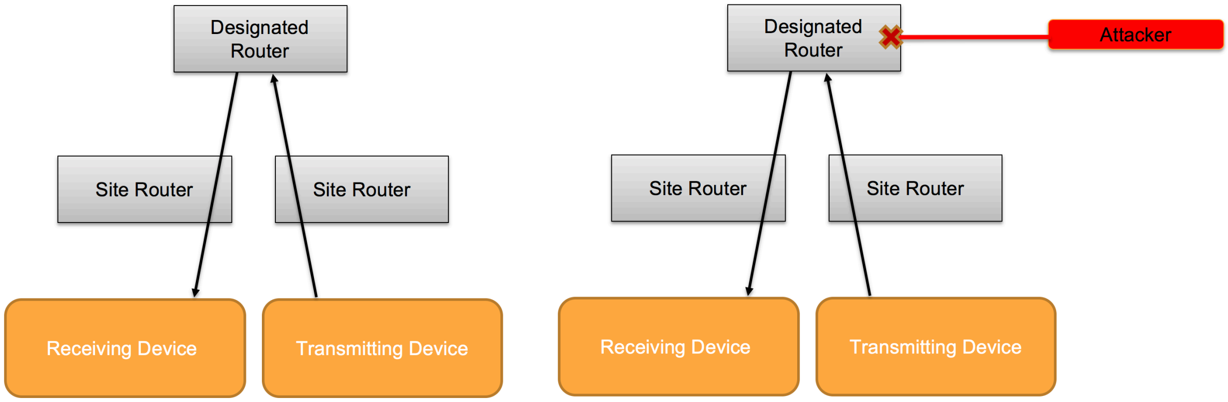


Figure 79: Schematic of testing end-to-end latency (left) without attack and (right) with attack.

We compare the latency distribution with no attacker (Figure 81 left) with that under attack (Figure 81 right). The mean Round Trip Time (RTT) with no attacker is 17.03 ms and under attack is 17.4 ms.

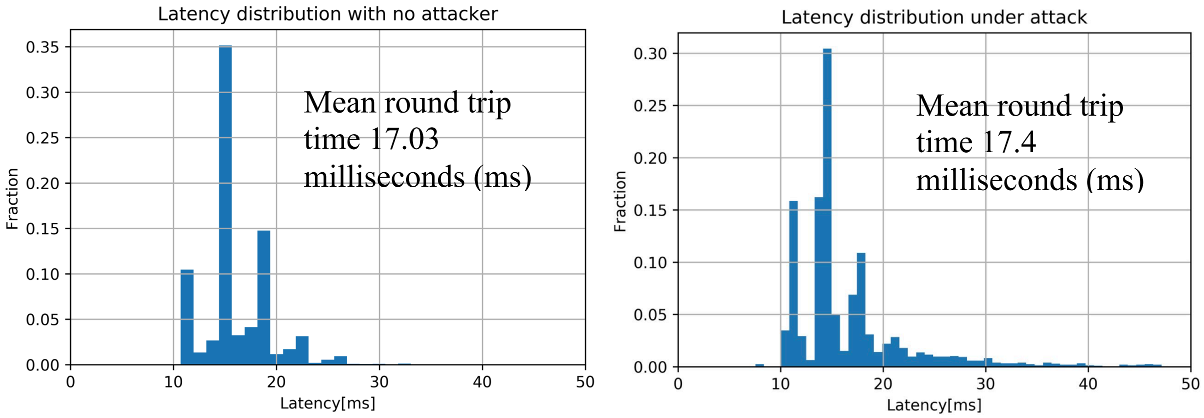


Figure 80: Result of testing (left) without attack and (right) under attack.

Thus the experiment shows that WAVE exceeds the milestone. Traffic latency is essentially unaffected by the presence of 50% attack bandwidth (only 10% required). While the latency under attack exhibits a longer tail distribution, this would not be significant for any application.

Task 11: Building Applications

Multi-application study

We developed a platform for multi-applications in buildings (Figure 82) with Hamilton sensors, routers, device drivers (thermostat, interval meter) and price (or Demand Response (DR)) signal, a time-series database, and an open library of analytics. The analytics frontend provides access to archived data. The platform is programming language-agnostic. The data access is authorized with WAVE.

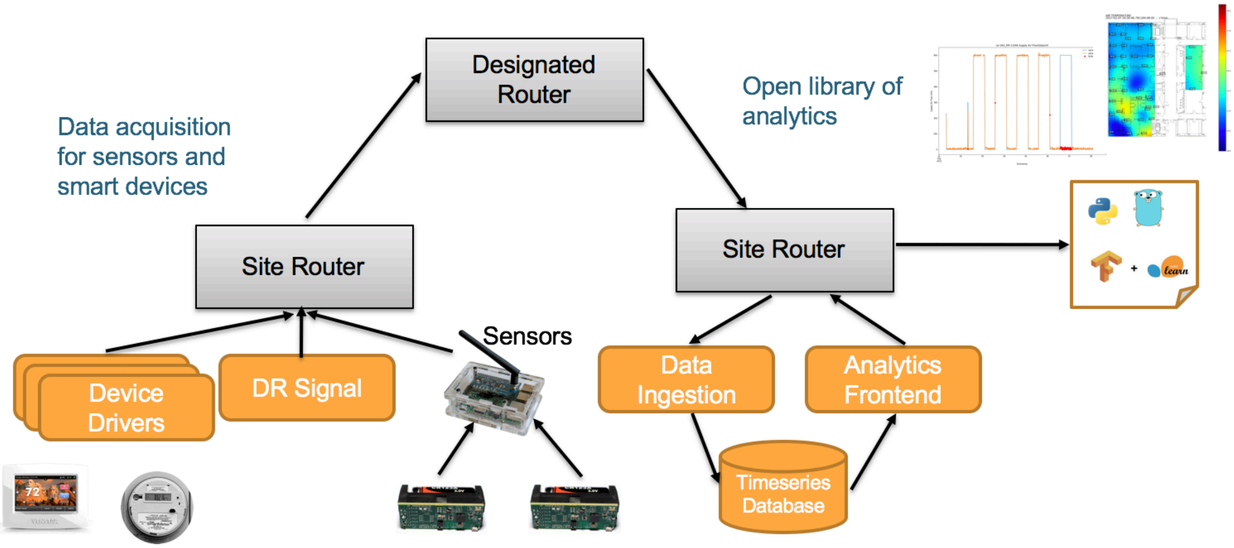


Figure 81: Building Energy Analytics

We developed and tested applications for Model Predictive Control and Reinforcement Learning. We have developed models for occupancy, thermal systems, and comfort for use in Model Predictive Control and visualizing simulations based on user-input of a cost-comfort index. We developed microservices such as those shown in Figure 83. We developed integrated controls for testing DR events summer 2018.

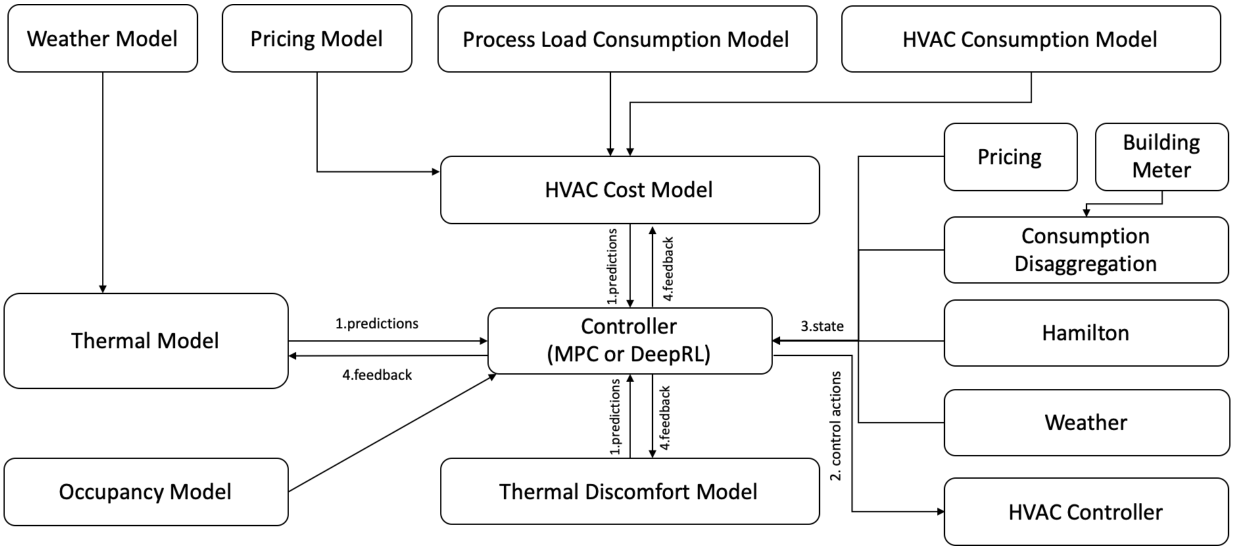


Figure 82: Multi-application microservices architecture.

As reported previously, the XBOS platform allows the aggregation and management of heterogenous data from a variety of sources (e.g., Hamilton sensors, thermostats, plug loads, interval meters, lights). Often there are multiple conflicting objectives: minimizing cost, discomfort, or energy consumption. We have developed multiple control strategies from simple heuristics to advanced optimization techniques (e.g., setpoint expansion, alternate HVAC zones, Model Predictive Control (MPC), Deep Reinforcement Learning (DeepRL)). Research questions we explored: how can we pick the right control strategy for a given building? How do we port a good strategy to work across multiple buildings? How can we compare multiple control strategies across one or more buildings? How do we do all of these easily while enforcing the existing guarantees (security, delegation, etc.)?

Microservices can help us achieve these goals, as they provide a simple unified abstraction for domain experts. Using microservices (prepackaged scripts) present a low barrier for non-Computer Science experts to develop applications on top of XBOS, provides a uniform abstraction to access historical or future timeseries data, and uses state of the art systems (WAVE, Mortar, BTrDB). The individual services with micro functionality (e.g., price, occupancy) allow the modularity to reason about correctness and performance of each microservice separately, allows building new services by combining multiple existing microservices, and provides a means to improve/replace existing services without affecting others. The microservices can be hosted locally or in the cloud, and leverage state of the art in service management (e.g., Docker, Kubernetes, gRPC).

The types of microservices are as follows:

Historical data library/services:

- Retrieve historical/current data securely from Mortar/BtrDB (e.g., outdoor temperature, HVAC Data, Hamilton occupancy, etc.)
- Retrieve historical/current data from external sources

Forecast data services:

- Models to forecast the future state of the building for planning purposes
- Models are often based on historical data (using the historical data library) such as occupancy prediction or process load energy consumption prediction, but can also use external services (e.g., National Weather Service Forecast)

Uniform data services:

- Provides a uniform abstraction on top of historic and forecast data
- Combines historical & forecast services with some interpolation for gaps (e.g., a utility/tariff price or total building energy consumption)

HVAC optimizer & simulation services:

- They use the various uniform data services for planning
- Output an action (heat, cool, do nothing) or a series of actions (in the case of a simulation) and their corresponding expected (temperature(s), and building/HVAC energy consumption(s))

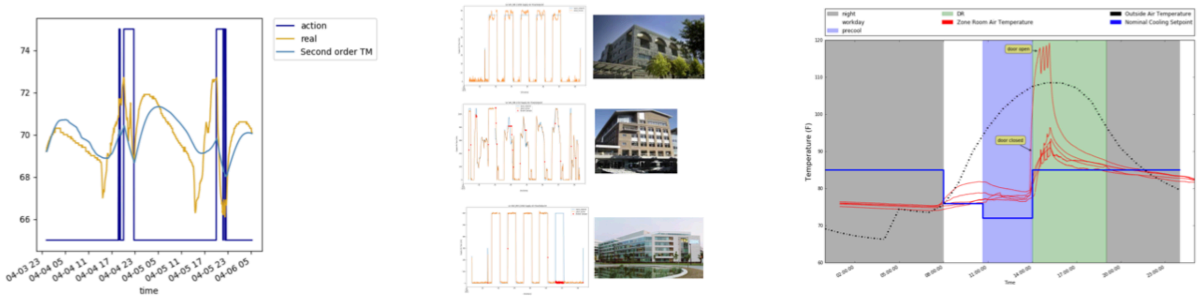
- From simple heuristics such as setpoint expansion to advance techniques such as MPC and DeepRL

Additional services:

- Action Enactor –enforces the action chosen by the optimizer in any given building
- Controller – combines different services (e.g., an optimizer, current data, and action enactor) to control one or more buildings
- Service Discovery – discover available services
- Client– high level library to facilitate use of services for domain expert

Demonstration of Building-Scale energy analytics

We have installed this platform in a number of buildings, with two having the Hamilton sensors. We developed several applications: one application uses the Hamilton sensors for thermal modeling for Model Predictive Control (MPC) in real-time. Another application provides automated fault detection and diagnosis using archived data. A third application is a demand response event controller with analytics.



Thermal Modeling for MPC

Automated Fault Detection + Diagnosis

Demand Response Event Controller + Analytics

Figure 83: Three applications using Hamilton sensors

We developed a means of obtaining price signals from two utilities and securely providing this data to applications that can then act on these signals, for example, by changing the setpoints of thermostats to reduce air conditioning.

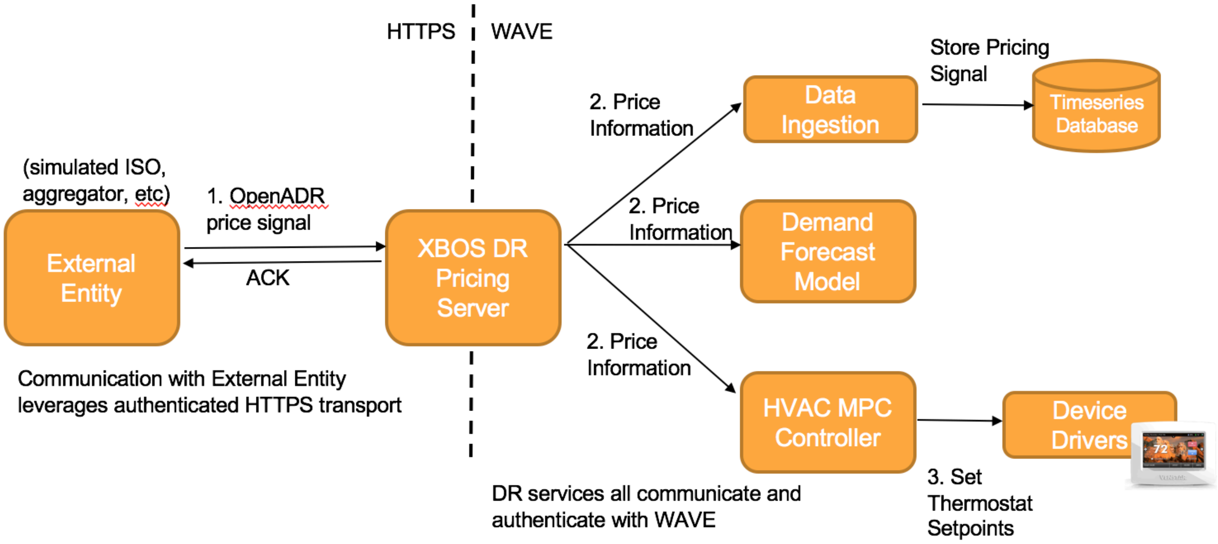
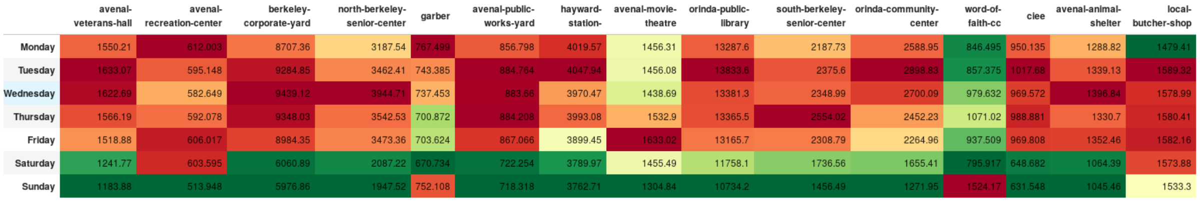


Figure 84: Schematic of price signal informing control

We developed energy consumption baselines for HVAC versus process loads, whole building consumption, and based on tariff price models. Figure 85 below shows a visualization of energy consumption for several buildings during different days of the week. We have developed fault detection methods to discover simultaneous heating/cooling (Figure 86) and "rogue" zones (HVAC zones that regularly fail to reach temperature setpoints).



	avenal-veterans-hall	avenal-recreation-center	berkeley-corporate-yard	north-berkeley-senior-center	garber	avenal-public-works-yard	hayward-station-	avenal-movie-theatre	orinda-public-library	south-berkeley-senior-center	orinda-community-center	word-of-faith-cc	ciee	avenal-animal-shelter	local-butcher-shop
Monday	1550.21	612.003	8707.36	3187.54	767.499	856.798	4019.57	1456.31	13287.6	2187.73	2588.95	846.495	950.135	1288.82	1479.41
Tuesday	1633.07	595.148	9284.85	3462.41	743.385	884.764	4047.94	1456.08	13833.6	2375.6	2898.83	857.375	1017.68	1339.13	1589.32
Wednesday	1622.69	582.649	9439.12	3944.71	737.453	883.66	3970.47	1438.69	13381.3	2348.99	2700.09	979.632	969.572	1396.84	1578.99
Thursday	1566.19	592.078	9348.03	3542.53	700.872	884.208	3993.08	1532.9	13365.5	2554.02	2452.23	1071.02	988.881	1330.7	1580.41
Friday	1518.88	606.017	8984.35	3473.36	703.624	867.066	3899.45	1633.02	13165.7	2308.79	2264.96	937.509	969.808	1352.46	1582.16
Saturday	1241.77	603.595	6060.89	2087.22	670.734	722.254	3789.97	1455.49	11758.1	1736.56	1655.41	795.917	648.682	1064.39	1573.88
Sunday	1183.88	513.948	5976.86	1947.52	752.108	718.318	3762.71	1304.84	10734.2	1456.49	1271.95	1524.17	631.548	1045.46	1533.3

Figure 85: Colored coded energy consumption in multiple buildings.

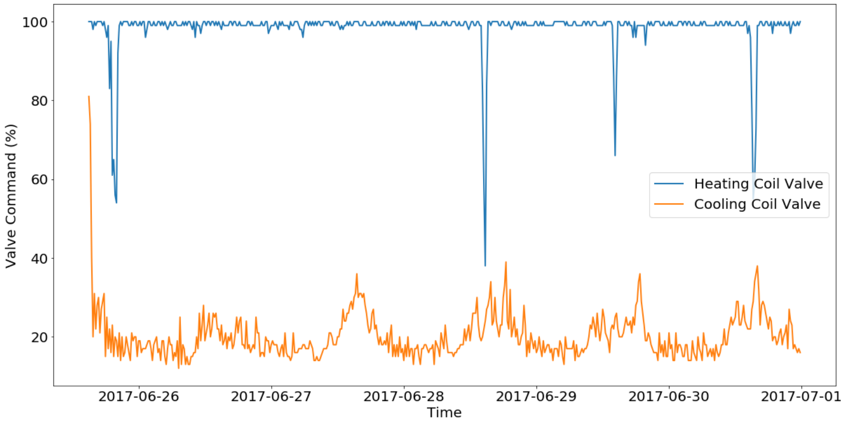


Figure 86: Analytic that discovers simultaneous heating and cooling.

Milestone 13.1: Demonstrate 80% correlation of wide deadband operation with non-occupied periods through the automated processes enabled by the sensor network. (M42)

As mentioned previously, we had a distribution of 14 Hamilton sensors, 4 networked thermostats, and a networked interval meter gateway for realtime building energy data in a 7500 sf floor of a building.

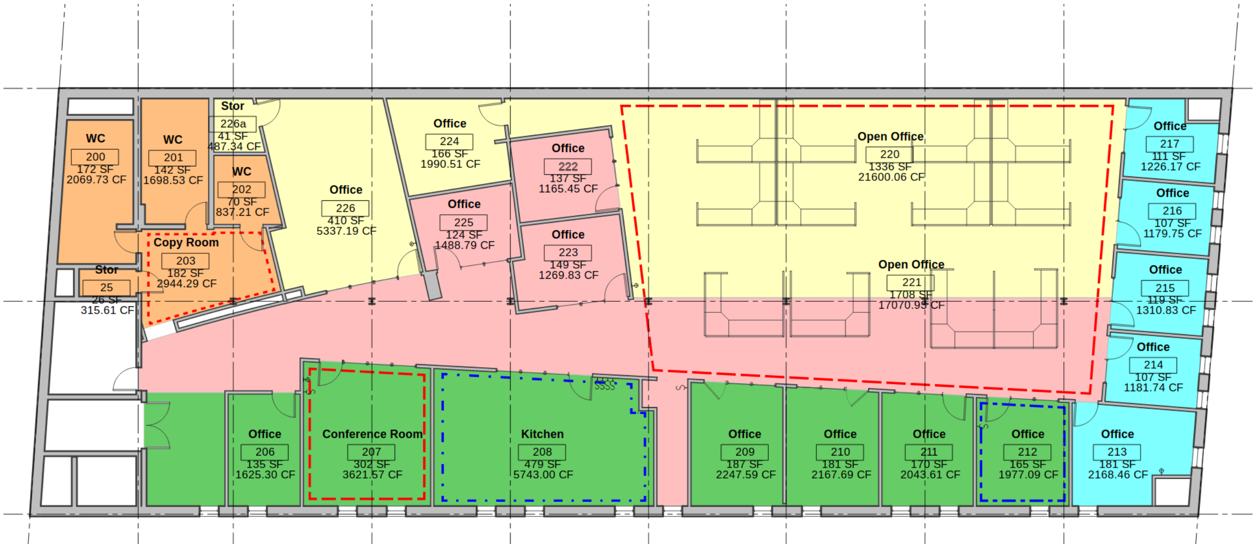


Figure 87: Floor plan showing HVAC zones. There was a Hamilton sensor in each of the private offices on the east and south of the building, and distributed in the open plan area.

The graph below shows a week of temperature (black line) from the Hamilton sensors compared to the thermostat setpoints (blue is cooling setpoint, red is heating setpoint) throughout the week for three different zones: top is the east zone (turquoise in the floor plan above), middle is the north zone (yellow), and south zone (green).

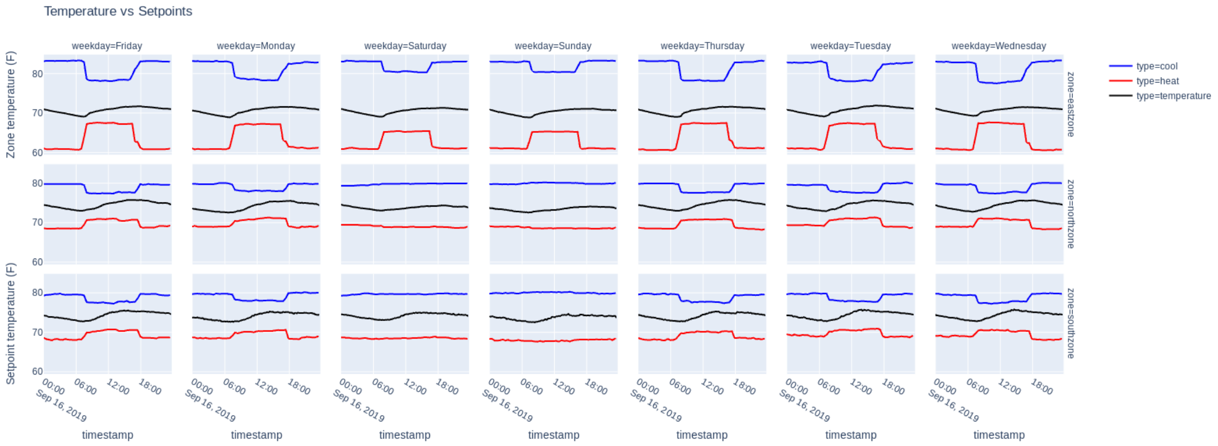


Figure 88: Ambient indoor temperature of the Hamilton sensors compared to the temperature setpoints for three zones.

The graph below shows the aggregated outdoor temperature compared to the temperature setpoint data during the same week.

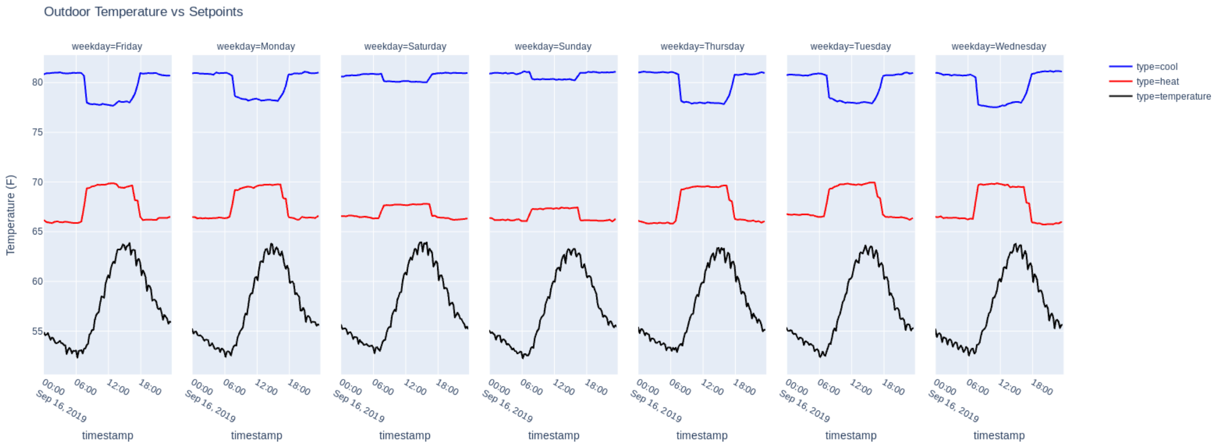


Figure 89: Outdoor temperature from weather station compared to temperature setpoints.

The graph below shows occupancy compared to temperature setpoints.

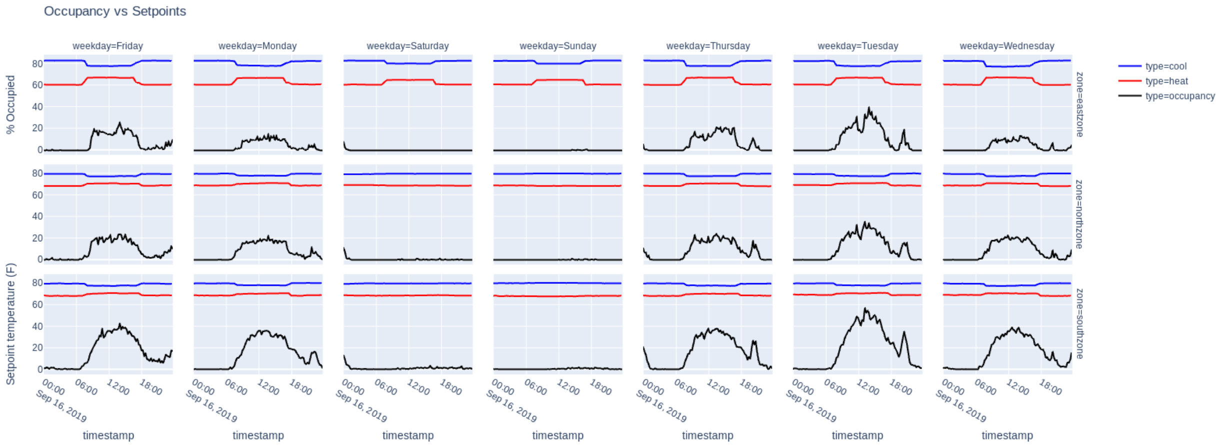
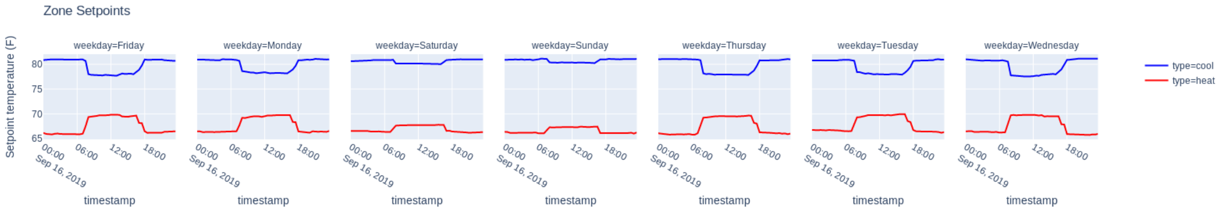


Figure 90: Percent occupied (black) compared to temperature setpoints.

The graphs below show building energy (bottom graph) compared to the setpoints (top graph).



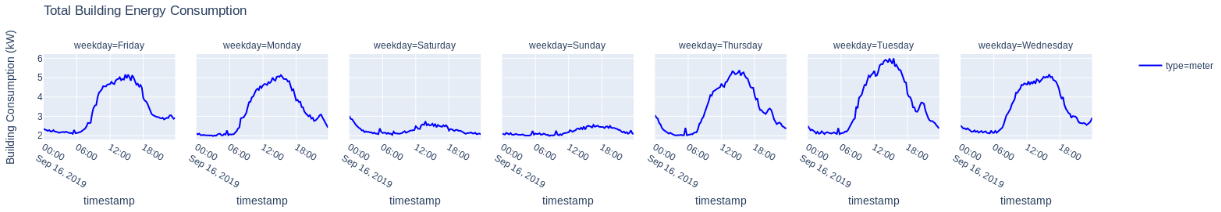


Figure 91: Top: temperature setpoints compared to Bottom: whole building energy.

A major benefit of Hamilton sensors was to provide inexpensive Occupant-Centered Control (OCC) in buildings to improve energy performance. We have implemented Brick data schema to allow Brick-based query processing (Gabe Fierro & Culler, 2018) to access longitudinal data, build models, establish semantic properties from observation, and isolate DR and OCC control improvements from more useful baselines (including weather and operational factors).

Where the original implementation of the Hamilton control platform included BTrDB, we are investigating advances in other open-source database offerings, such as Timescale (built on Postgres). We developed an analytics platform (G. Fierro et al., 2018; Gabe Fierro et al., 2020), which queries timeseries using Brick classes, and captures other building metadata (Gabe Fierro, Guduguntla, et al., 2019; Gabe Fierro, Koh, et al., 2019), and we are pushing models and analysis into the database (model params over time). Our current database has 107 buildings (all UC Berkeley), ~9000 data streams, ~44 billion data points spanning 2014 – 2020-03-17.

One recent example of metadata-based query processing was to look at energy consumption before and after shelter-in-place.

Milestone 13.2: Demonstrate 95% confirmed response of prescribed HVAC, lighting, and plugload actuation to DR event notification.

In this milestone, we integrated technological advances of Hamilton sensor development and the secure data bus WAVE into a complete building-wide secure system. We demonstrated the reliability of the end-to-end system in the collection and analysis of timeseries data from sensors and control in buildings. The following paragraphs describe the:

- Successful enactment of multiple control strategies for HVAC systems in a suite of buildings
- Successful control of plug loads and HVAC system as well as an electric vehicle charging station (an extension of a smart building deployment), and
- Stress test of the message bus (i.e., WAVEMQ) by leveraging it to deliver high frequency messages from a deployment of multiple microsynchrophasors measuring voltage angles and magnitude of the electric grid.

We developed several control strategies for small commercial HVAC systems, using networked thermostats, from heuristics (set point expansion) to advanced techniques (MPC and DeepRL). These strategies relied on the collection of sensor data to model the behavior of the building and its occupants (e.g., thermal response, HVAC consumption, occupancy prediction). We enacted these control strategies in a suite of buildings across California. The system achieved a reliability of 100% (all control events were successfully enacted by the system).

One building was a single-story classroom and office building at CSU Dominguez Hills in southern California. Duration: 14.5 hours over 4 days. The building had 16 HVAC zones each controlled by a Pelican networked thermostat that received data from the Enlighted lighting system occupancy sensors. We were able to control heating and cooling setpoints, mode (heat or cool), and override conditions for each zone. Given the 64 control points and 16 zones, we sent 7364 control requests to respond to real and simulated demand response events over several days: all 7364 events were enacted (100% reliability).

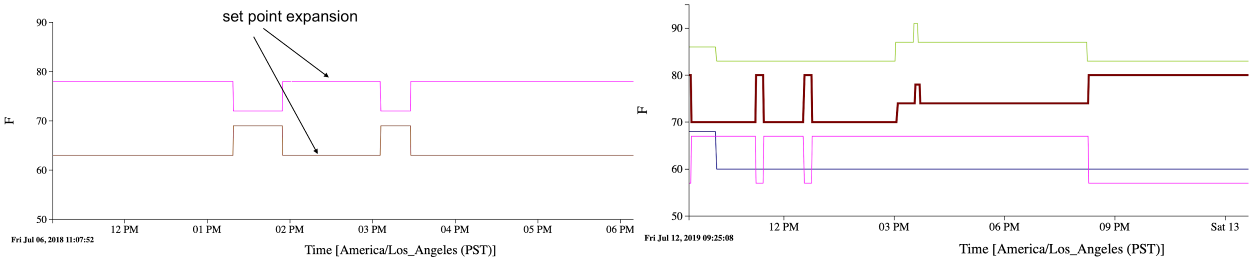


Figure 92: HVAC control measures in three zones in a CSUDH classroom during warm weather. Left: one zone used the expansion of setpoints to reduce cooling loads. Right: two zones show different behavior based on automatic occupancy detection as well as overridden conditions based on price: the green line shows expanded setpoint (unoccupied) and the brown line show periods of occupancy. Both show expanded setpoints due to an event at 3p.

In total, we analyzed the reliability of HVAC control in four buildings, with a total controlled period of 42 hours over 10 days, controlling 33 zones and 132 control points, with 9000 total number of control/enacted events.

We controlled multiple loads in a single building (Richmond Field Station, Richmond, CA). Here we analyzed the reliability of the actuation of heating, plug loads, and EV charging. The total duration of control was two hours. The type of control points included three HVAC Zones using networked thermostats for electric baseboard heating, five Plug Loads (refrigerator, portable heater, portable fan, microwave, electric kettle), 1 Level 2 Electric Vehicle charger. The total number of sent/received messages was 1,704 over 64 control/enacted events. The reliability of the sent messages to actual actuation was 100%.

The final reliability test of the WAVE/WAVEMQ message bus was in stress-testing using high frequency data for the ENERGISE project on distributed grid control. This DOE project developed a research platform for novel grid control frameworks. The deployment spanned LBNL's FLEXLAB, cloud services, campus datacenter, other distributed computers, and integrate micro phase measurement unit (uPMUs) or microsynchrophasor, load control, smart inverters, and distributed controllers. For this project, we developed a Python library over WAVEMQ to simplify development: researchers required minimal input from UCB team to prototype control and allowed easy reconfiguration of data flows. A single cloud-based broker served all traffic for project and operated continually for months.

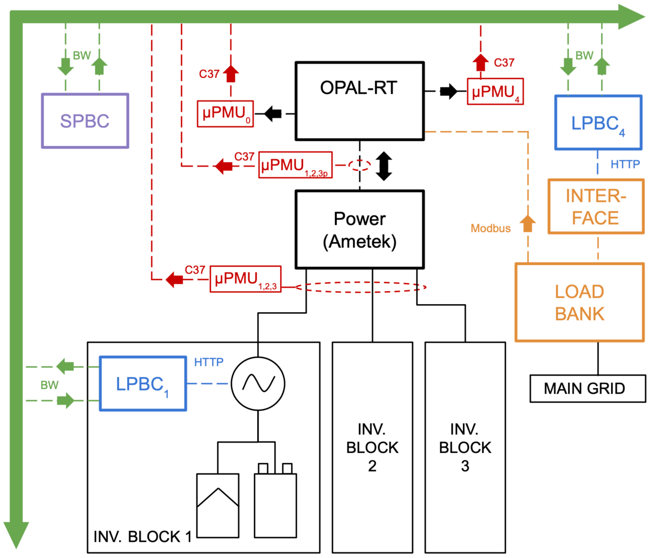


Figure 93: ENERGISE message bus connecting various devices

The table below shows the types of processes, and messages per minute received and sent.

Process Type	Application Class	# Deployed	# Sub Channels	Msg/min (Recv)	# Pub Channels	Msg/min (Send)
PMU Driver	Data Source	4	0	0	6	360
Distributed Ctrl	Controller	3	13	721	1	60
Supervisory Ctrl	Controller	1	9	540	3	3
Data Historian	Data Sink	1	30	1623	0	0

Table 13: ENERGISE message bus details

Over 27 days, WAVEMQ delivered 354,254,138 messages, of which 353,967 were dropped for "any reason" (including dropping messages out of unneeded/abandoned queues). This means that 99.9% (almost exactly) of messages were delivered, though this is actually an underestimate. All measurements were conducted on the primary WAVEMQ broker, which routes the majority of traffic in the WAVEMQ deployments. See Figure 94 below for the timeseries of message publish rate over the 27 day period (about 25-30 messages per second). Aggregation to the hour means that the drops to "zero" don't quite reach the bottom: this is due to downtime being <1 hr over this period. The downward slope of message rate from 6/3 to 6/10 is due to a data source experiencing disruption.

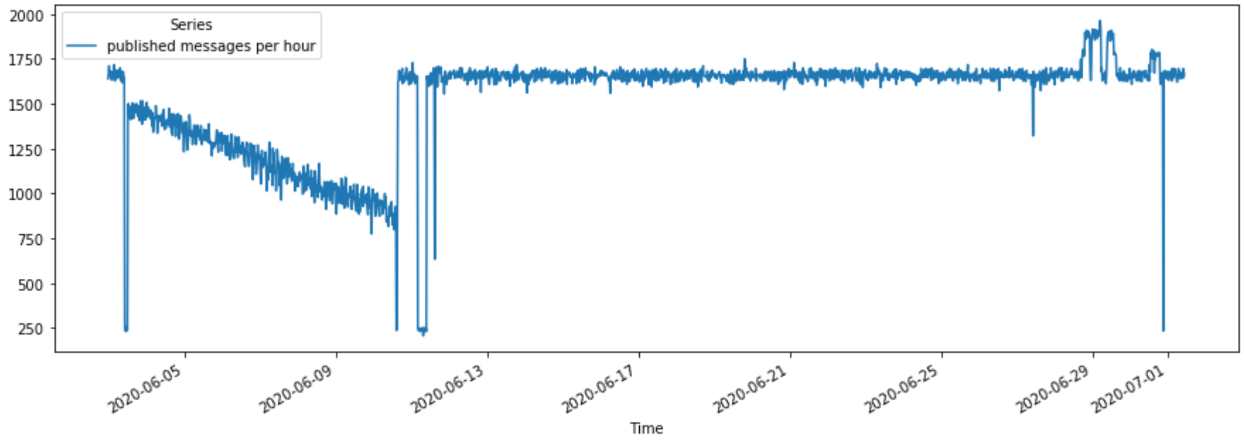


Figure 94: Published messages per hour over 27 days on WAVEMQ message bus

The graph below shows a histogram of the published messages per hour over the 27 day period.

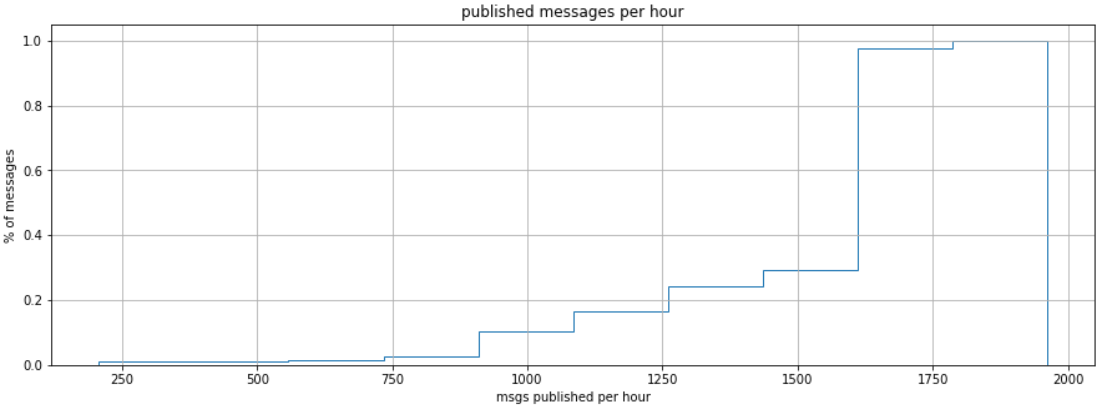


Figure 95: Histogram of messaged published per hour over 27 days.

The graph below shows the message delivery ratio (the fraction of messages which were successfully delivered through the broker). On average, the platform demonstrated a delivery ratio of .999 (out of 1.000) over the 27 day period.

Figure 96: Message delivery ratio

Another test looked at the message latency (1-way) timeseries over a 20 hour period (ping-pong setup between LBNL server and miniature computer server at a researcher's apartment). Figure 97 below show very short latencies and the response after a short and long interruption. At 6am (delayed messages arrive after short outage) and 12pm-4pm (longer internet outage, server reboot lead to lost messages (short expiry)).

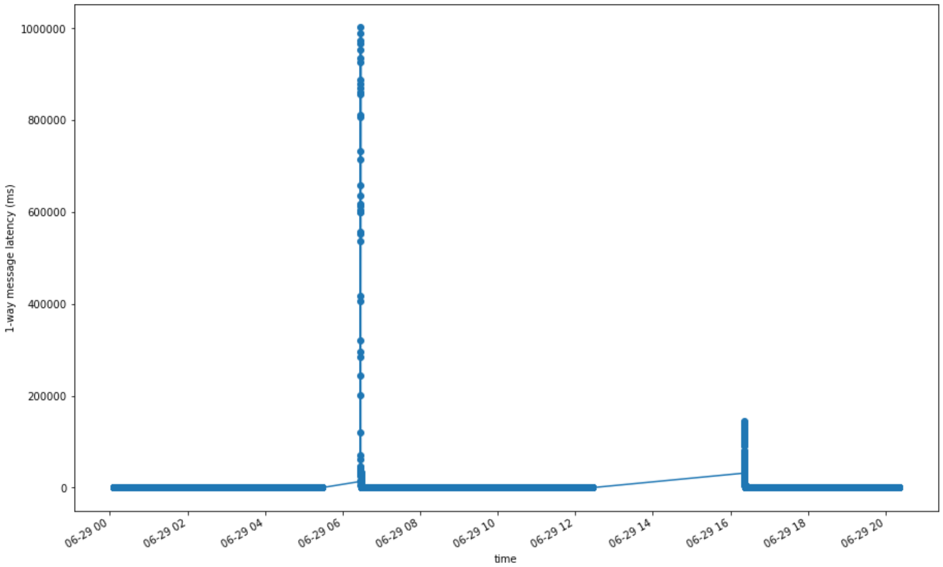


Figure 97: Message latency

In summary, the reliability of actuation messages exceeded the milestone set of 95%.

Task 12: Technology to Market

Transition Activities

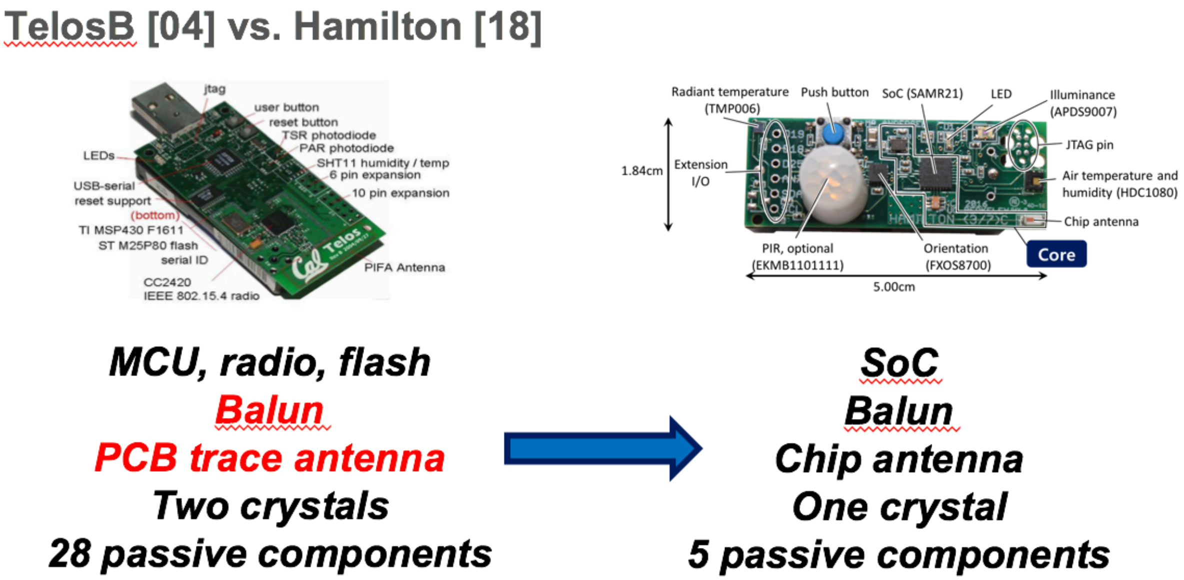
Milestone 14.1*: Preliminary post-project financing and transition plan presented, including industry study of potential market inclusion, expected business model, target partners, identified financing needs for next steps and potential funding sources if applicable, market sector, and commercialization strategy, key technological risks and mitigations, key manufacturability and scale-up risks and mitigations, and next steps and actions with associated timeline.*

We outlined several areas in the market, both in building energy applications, and as a platform for other sensors. Partners include architects and engineers; building types are residential, commercial, hospitals, laboratories.

We have identified key contributions of Hamilton to the research community: we have triggered shifts on the following fundamental paradigms in this regime, which was formed two decades ago when this field emerged:

1. Low-power board assembly is possible but hard/expensive [1999]
2. The event-driven model should be used for concurrency [2000]
3. Radio is the major energy consumer in a networked sensor [2004]

Figure 98 below shows the comparison between the TelosB mote from 2004 to Hamilton in 2018: low power board assembly is now possible with highly integrated components.



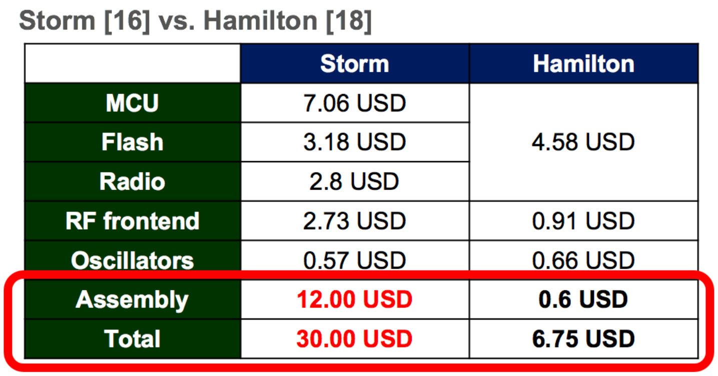
TelosB [04] vs. Hamilton [18]

MCU, radio, flash
Balun
PCB trace antenna
Two crystals
28 passive components

SoC
Balun
Chip antenna
One crystal
5 passive components

Figure 98: Comparison between TelosB sensor from 2004 with Hamilton in 2018.

Careful design has driven down the cost of assembly of Hamilton compared to the Storm mote, developed by Andersen in 2016 (Figure 99)(Andersen, Fierro, et al., 2017).



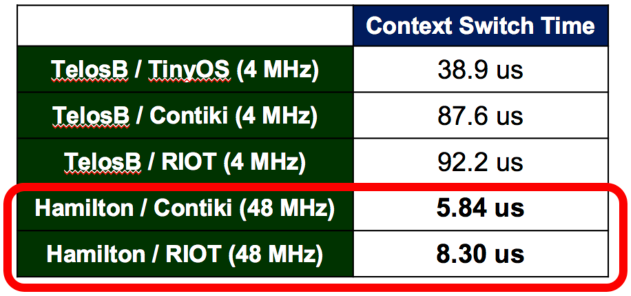
Storm [16] vs. Hamilton [18]

	Storm	Hamilton
MCU	7.06 USD	4.58 USD
Flash	3.18 USD	
Radio	2.8 USD	
RF frontend	2.73 USD	0.91 USD
Oscillators	0.57 USD	0.66 USD
Assembly	**12.00 USD**	**0.6 USD**
Total	**30.00 USD**	**6.75 USD**

Figure 99: Comparison between Hamilton-precursor (Storm) in 2016 with Hamilton in 2018

Referring to point one above, the low-power board assembly is now the cheapest part.

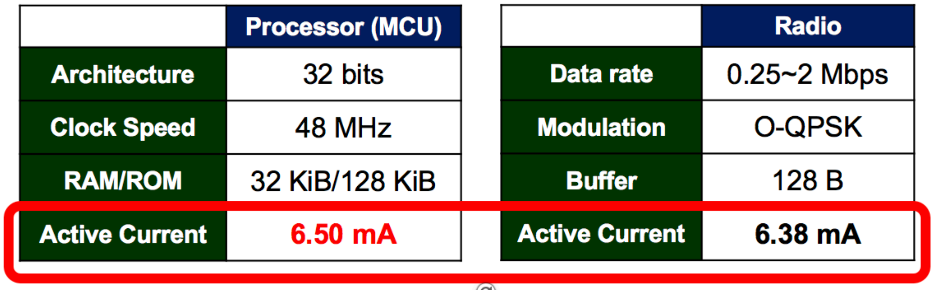
Regarding the second point, Hamilton has drastically lowered the context switch overhead, showing that (point 2) replacing an event model with the thread model is the most viable for concurrency.



	Context Switch Time
TelosB / TinyOS (4 MHz)	38.9 us
TelosB / Contiki (4 MHz)	87.6 us
TelosB / RIOT (4 MHz)	92.2 us
Hamilton / Contiki (48 MHz)	**5.84 us**
Hamilton / RIOT (48 MHz)	**8.30 us**

Figure 100: Hamilton drastrically reduced context switch time

Finally, regarding the third point, the radio power consumption is slightly lower than the MCU in the Hamilton.

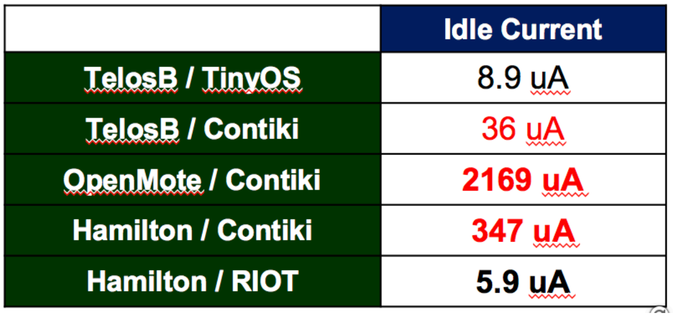


	Processor (MCU)
Architecture	32 bits
Clock Speed	48 MHz
RAM/ROM	32 KiB/128 KiB
Active Current	**6.50 mA**

	Radio
Data rate	0.25~2 Mbps
Modulation	O-QPSK
Buffer	128 B
Active Current	**6.38 mA**

Figure 101: Radio is the component with the highest use of power

With careful MCU operation, Hamilton consumes lower current when idle than TelosB; Figure 102 below shows the improvement using RIOT OS compared to Contiki.



	Idle Current
TelosB / TinyOS	8.9 uA
TelosB / Contiki	36 uA
OpenMote / Contiki	2169 uA
Hamilton / Contiki	347 uA
Hamilton / RIOT	5.9 uA

Figure 102: Hamilton consumes lower current when idle.

Recent activity includes other researchers' experiences using Hamilton sensors.

A colleague deployed Hamilton Generation 1 sensors (using BOSSwave or Wave 1.0) in a research project. The 32 sensors were installed for 2.3 years (July 2017-November 2019). The project required the use of indoor temperature sensors inside four multi-family sites in central California (community rooms, offices and dwelling units) and measuring temperature inside HVAC supply vents, in order to determine whether compressors were in heating or cooling mode, as thermostat data was not available at this site.

The research team reported a number of issues:

- Condensation shorts: The team found that the Hamilton sensors in the changing temperature air streams caused condensation to form on the devices, which was sufficient in some cases to short out the device. To eliminate this problem two methods were used: installing Hamiltons in plastic bags with a desiccant included, and installing separate temperature sensors wired directly into the Hobo U-30 data loggers. To be fair, these Hamiltons did not have an enclosure and were not designed for the fan coil locations.
- Data loss: Another issue was data loss: the server by which data was accessed and the local Internet went down periodically. Data loss was a central foci of the WAVE 2 development; WaveMQ was specifically designed for intermittent connectivity (for Internet interruption).
- Intermittent data reception: A few Hamilton sensors had spotty or intermittent data reception, where the distance and building construction was similar to other cases. We do not know the reason.
- Short battery life: Batteries needed replacing on some units after 18 months (great and not an issue, but not as long as expected)

Overall, the researchers felt that the Hamilton sensors performed "much better than other options out there for this application."

The Hamilton project was presented and demonstrated at a CONIX (Computing on Network Infrastructure for Persuasive Perception, Cognition, and Action) consortium on situation awareness; the CONIX Research Center is one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. Both Wave 2.0 (separate authentication/authorization from the data bus) and WaveMQ (allow more secure and more performant networking) were described and demonstrated. There was a Network Visualization demo that allowed one to visualize WiFi and sensor network traffic (Figure 103). The visibility of network flows was protected by WAVE, and earned the Best Demo award ("Realtime VR Network Traffic Visualization" by Meghan Clark, Gabe Fierro, Neal Jackson, David Culler and Prabal Dutta). The concept was to bring WAVE permission model into Virtual Reality (VR)—to "see" packets between devices, and to choose who can see (permissions using delegation of trust,) so end users can see where the data is going.

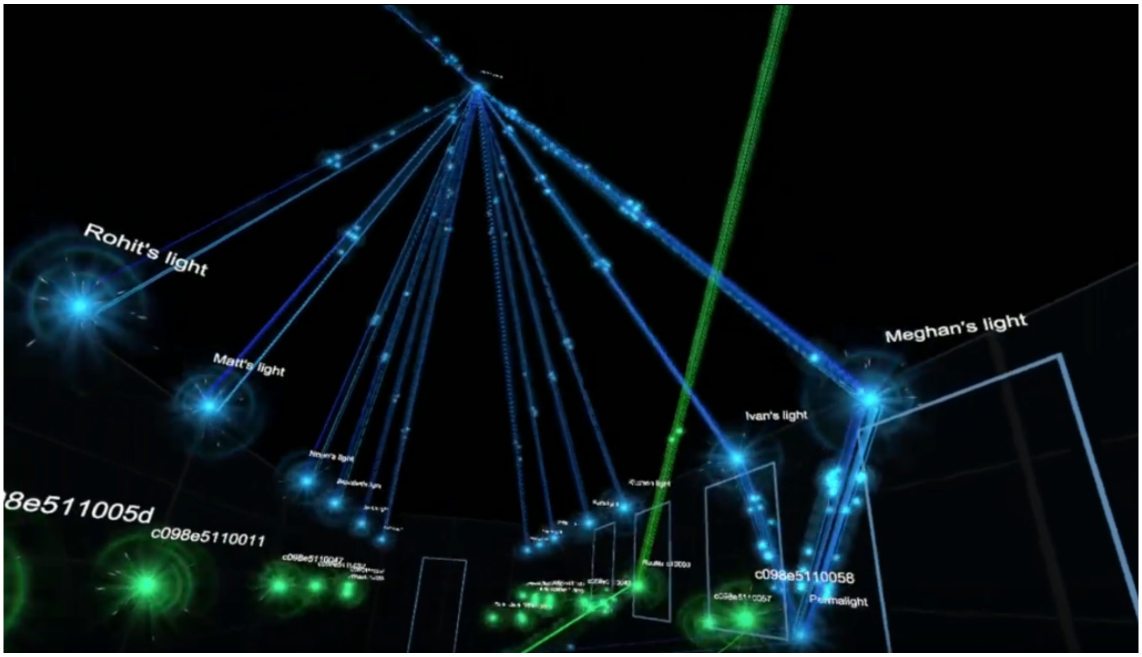


Figure 103: Visualization of Network Traffic demonstration at CONIX using WAVE

Other outreach included a paper and presentation at BuildSys, a paper entitled DataSet, and participating in a meeting at NREL regarding semantic interoperability models in buildings.

Refined Transition Plan

Milestone 14.2: *Present at quarterly review meeting a refined transition plan. Develop pitch deck that will be used to engage next-stage partners/funders. Present to DOE at quarterly review meeting. Report on progress against post-project transition plan.*

The startup HamiltonIOT is still selling Hamiltons. We have engaged with SRC and several member companies regarding the integration of WAVE MQ with MQTT. We have direct projects with Intel and ARM. At a recent UC Berkeley RISEcamp, there were demonstrations of Hamilton in front of 2 dozen member companies. We have engagements with financial services companies. We have continued building application demonstrations engaging CEC, Siemens, QuEST.

Hardware

Hamilton: sensor hardware (and data aggregation). HamiltonIOT continues to sell Hamilton hardware: https://hamiltoniot.com/; the sensor platform led to advancements in RiotOS and OpenThread.

Software

BTrDB (Berkeley Tree Database): high speed database: http://btrdb.io/index.html

Michael Andersen spent two years at PingThings, who is using BTrDB in their commercial offerings to utilities, touting "The world's fastest, nanosecond-precise, time series analysis and AI platform." [pingthings.io]

WAVE and WAVEMQ: secure middleware was used in the eXtensible Building Operating System (XBOS) (https://docs.xbos.io/), which formed the backbone of the $4M CEC-funded XBOS-DR project bringing demand response to small commercial buildings, and led to two other funded projects using XBOS (FlexHP (LBNL-led project, funded by CEC) and OpenBOS-NY (TRC-led project, funded by NYSERDA). Another opensource tool supporting WAVE is JEDI (Joining Encryption and Delegation for IoT): end to end encryption for WAVE (github.com/ucbrise/jedi-protocol-go).

TCP/IP and Thread: TCP/IP work by PhD student Sam Kumar, including MAGE: Efficient Execution Engine for Secure Multi-Party Computation. Sam was recently invited to speak at Google Thread Group virtual meeting on TCP/Thread work (Nov 2020).

Brick schema and Mortar: metadata tagging and query was further developed, and led to another DOE BENEFIT award.

Personnel

Trained thought-leaders: Michael Andersen (Andersen, 2019), Hyung Sin Kim, Moustafa Abdelbaky, Jack Kolb, Sam Kumar, Gabe Fierro

Refined and final transition plan

Hamilton IoT

Hamilton IoT is a startup company founded in 2017 to bring the Hamilton sensor technology to market. The company identified a number of use cases for low-cost multi-modal sensors across several market verticals including real estate, life sciences and research. It offers a unique value proposition of providing a low-cost, easy to deploy, sensor with instantly available data. Starting with open-source hardware and software, Hamilton IoT is focused on manufacturing, supporting, and managing cloud infrastructure to make deploying sensors easy to use.

Product

The Hamilton IoT solution consists of sensors, gateways and the Hamilton Cloud.

Sensors are wireless and battery backed (based on the open-source Hamilton designs) to make deployment as easy as possible, obviating the need to run power or data cables. This is a major benefit for many use cases. For example, in office buildings wired sensors require dedicated conduit to be run on the ceiling, an expensive task requiring an electrician and installation outside of standard occupied hours. In contrast, the Hamilton sensor can be clipped or magnetically attached to the ceiling by anyone and lasts for up to 5 years without service.

Sensors are offered in two configurations: with or without Passive Infrared (PIR) motion sensors. The ultra-low-power PIR sensors make up a large fraction of the overall device cost and are therefore offered as an optional feature.

The Hamilton Cloud is designed to provide instant access to data without requiring setup, hosting or complex cloud configuration. Sensors are pre-registered before being shipped to a customer so that data begins streaming to the cloud as soon as the sensors and gateway are powered on. A basic cloud hosting tier is provided at no additional cost with purchase of the sensors.

Key cloud features:

- Sample rate: all sensors sample three times per minute.
- Latency: how soon data is available to be viewed/analyzed/downloaded. The Hamilton Cloud supports up to real-time (sub-minute) latency for use cases that require rapid control decisions. Longer latencies are appropriate for historical analyses.
- Historic data: amount of data stored. Infinite storage is available leveraging AWS S3 and cold storage.
- Metadata: ability to add unlimited tags / descriptors to each device.
- Monitoring: ability to see the status and last reported data from each device.
- Export: ability to download data for offline analysis.
- API access: APIs for accessing data. Includes example code in Python and R.
- Web hooks: ability to register an API callback based on a data trigger (e.g. alert whenever temperature falls below 65 F)
- Email notifications: automated email notifications based on data triggers.

The Hamilton cloud is offered in three tiers aimed at common use cases. The Basic tier allows free data collection out of the box and is suitable for testing and short-term data collection. The Pro tier is designed for long-term data collection and analysis. The Control tier is designed for real-time use, integration into customer-developed code or modeling tools. Breakdown of features by tier:

	Basic	Pro	Control
Sampling rate	3 per minute	3 per minute	3 per minute
Data Latency	10 Minutes	5 Minutes	Real time (<1 Minute)
Historic data	1 month	Infinite	Infinite
Device metadata	Yes	Yes	Yes
Device monitoring	Yes	Yes	Yes
Export functions	-	Yes	Yes
API Access	-	200 calls per day	2000 calls per day
Web hooks	-	Yes (5 min latency)	Yes (real time)
Email notification	-	-	Yes

Use Cases

Hamilton IoT has a range of customers and use cases. For example, Michigan State University (MSU) uses Hamilton IoT for animal health monitoring in a life sciences laboratory. Specifically, the researchers at MSU are using rats for medical studies and require a controlled environment with fixed hours without disturbances for a prescribed sleep period. MSU used the Hamilton IoT sensors to detect disturbances using a combination of vibration, light and motion sensors. They required a high density of sensors with one sensor per cage. Battery power and wireless connectivity was essential for installation and maintenance.

TRC Companies Inc, a leading consulting, engineering and construction management firm, uses Hamilton IoT sensors for energy efficiency. They deployed Hamilton sensors in a high-density housing development to monitor occupancy and temperature. Based on this data, thermostats could be automatically adjusted to save energy. Having a gateway that could cover many sensors (up to 1000) was key for rapid deployment and managing costs.

Overall, Hamilton IoT supports a diverse range of applications by providing easy to deploy and easy to use wireless sensors.

Wireless Networking

The Hamilton project involved two significant marketable efforts in developing new software for network communications among low-power motes, including the Hamilton board itself. First, we implemented support for the OpenThread standard networking stack on Hamilton. Much of our code related to this effort has since been adopted as part of RIOT, an open-source operating system for Internet of Things devices with a highly active community of developers from around the world. We expect our software to remain part of RIOT for the foreseeable future and, because it is open-source, we anticipate other developers contributing to and improving upon this work going forward.

Second, we developed a fully-featured implementation of TCP specifically for low-power and embedded devices. While TCP is the dominant transport protocol in the modern Internet, historical concerns about its performance and power consumption had discouraged prior attempts at bringing it to the embedded setting. This meant that embedded devices generally could not freely interoperate with other computers over the Internet without something like an intermediate gateway device. We created a performant and power-efficient TCP implementation and presented this work at one of the leading academic conferences on computer networking. As a result, there has been significant interest from industry in adopting this work. Google, in particular, has shown interest and may be involved in bringing this work to the broader embedded market going forward.

XBOS

The XBOS project benefitted from a user survey and a large-scale pilot deployment to identify target markets and the needs of potential users. We found the following major value propositions for an automated building management system:

- Safety: adequate light, fresh air/ventilation, appropriate thermal conditions, furnishings
- Security: doors lock, controlled access
- Privacy: restricted access to building data and control
- Low risk: in terms of insurance for physical premises
- Comfort: thermal comfort, good light, little noise, good air quality
- Convenience: time savings measure (remote control, ease of maintenance)
- Cost: rent, salaries (more productive employees), energy demand charges, equipment, maintenance

All of those concerns have been addressed by XBOS in some form, and we believe all are important points to emphasize if a long-term interest in commercializing XBOS were to emerge. In the short-term, development and maintenance of XBOS is expected to continue through our collaborators at Lawrence Berkeley National Laboratory (LBNL). XBOS has served as the basis for several projects there, including a system for optimized control of grid-responsive buildings in the presence of local solar energy sources.

Brick

One aspect of XBOS that we expect to remain particularly active going forward is the Brick metadata effort. Brick offers a means of precisely describing the control elements and configuration of a building, allowing software to dynamically discover the building’s features and to account for them in its decisions regarding control, data collection, data analysis, etc. Brick has been supported by a consortium of academic partners at institutions including Carnegie Mellon University and the University of California, San Diego. Several companies have also become involved with Brick, including Johnson Controls. Finally, ideas from Brick are also being considered by ASHRAE, an association of building managers and engineers, in its ongoing standardization work.

DISCLAIMER:

References

Alston-Stepnitz, A., Sanguinetti, E., Peffer, A., & Alston-Stepnitz, E. (2020). Connected thermostats for low income households: Insights from user testing. *Summer Study for Energy Efficiency in Buildings*. https://escholarship.org/uc/item/61k1c32x

Andersen, M. P. (2019). *Decentralized Authorization with Private Delegation* [University of California Berkeley]. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-113.pdf

Andersen, M. P., Fierro, G., & Culler, D. E. (2017). Enabling Synergy in IoT: Platform to Service and Beyond. *Journal of Network and Computer Applications*, *81*, 96–110.

Andersen, M. P., Kim, H. S., & Culler, D. E. (2017). Demo abstract: Hamilton - a cost-effective, low power networked sensor for indoor environment monitoring. *BuildSys 2017 - Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, *2017-January*, 1–2. https://doi.org/10.1145/3137133.3141453

Andersen, M. P., Kolb, J., Chen, K., Culler, D. E., & Katz, R. (2017). Democratizing Authority in the Built Environment. *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys '17)*.

Andersen, M. P., Kolb, J., Chen, K., Fierro, G., Culler, D. E., & Popa, R. A. (2017). *WAVE: A decentralized authorization system for IoT via blockchain smart contracts*. https://people.eecs.berkeley.edu/~m.andersen/publications/wave2

Andersen, M. P., Kumar, S., AbdelBaky, M., Fierro, G., Kolb, J., Kim, H.-S., Culler, D. E., & Popa, R. A. (2019). WAVE: A Decentralized Authorization Framework with Transitive Delegation. *Proceedings of the 28th USENIX Security Symposium*, 1375–1392. https://www.usenix.org/conference/usenixsecurity19/presentation/andersen

Arens, E., Ghahramani, A., Przybyla, R., Andersen, M., Min, S., Peffer, T., Raftery, P., Zhu, M., Luu, V., & Zhang, H. (2020). Measuring 3D indoor air velocity via an inexpensive low-power ultrasonic anemometer. *Energy and Buildings*, *211*. https://doi.org/10.1016/j.enbuild.2020.109805

Fierro, G., Raftery, P., Peffer, T., Culler, D. E., Baky, M. A., Thomson, G., & Pritoni, M. (2018). Mortar: An open testbed for portable building analytics. *BuildSys 2018 - Proceedings of the 5th Conference on Systems for Built Environments*. https://doi.org/10.1145/3276774.3276796

Fierro, Gabe. (2021). *eXtensible Building Operating System (XBOS) Overview - XBOS Docs*. https://docs.xbos.io/

Fierro, Gabe. (2017). Demo abstract: HodDB - a query processor for brick. *BuildSys 2017 - Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, *2017-January*. https://doi.org/10.1145/3137133.3141449

Fierro, Gabe, & Culler, D. E. (2018). Design and analysis of a query processor for brick. *ACM Transactions on Sensor Networks*, *14*(3–4). https://doi.org/10.1145/3199666

Fierro, Gabe, Guduguntla, S., & Culler, D. E. (2019). Dataset: An open dataset and collection tool for BMS point labels. *DATA 2019 - Proceedings of the 2nd ACM Workshop on Data*

Acquisition To Analysis, Part of SenSys 2019, 40–42. https://doi.org/10.1145/3359427.3361922

Fierro, Gabe, Koh, J., Agarwal, Y., Gupta, R. K., & Culler, D. E. (2019). Beyond a House of Sticks: Formalizing Metadata Tags with Brick. *6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys)*.

Fierro, Gabe, Pritoni, M., AbdelBaky, M., Raftery, P., Peffer, T., Thomson, G., & Culler, D. E. (2020). Mortar: An Open Testbed for Portable Building Analytics (Journal Extension). *ACM Transactions on Sensor Networks*.

Kim, H.-S., Andersen, M. P., Chen, K., Kumar, S., Zhao, W. J., Ma, K., & Culler, D. E. (2018). System Architecture Directions for Post-SoC/32-bit Networked Sensors. *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems - SenSys '18*, 264–277. https://doi.org/10.1145/3274783.3274839

Kim, H. S., Ko, J., Culler, D. E., & Paek, J. (2017). Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey. In *IEEE Communications Surveys and Tutorials* (Vol. 19, Issue 4, pp. 2502–2525). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/COMST.2017.2751617

Kim, H. S., Kumar, S., & Culler, D. E. (2019). Thread/openthread: A compromise in low-power wireless multihop network architecture for the internet of things. *IEEE Communications Magazine*, *57*(7), 55–61. https://doi.org/10.1109/MCOM.2019.1800788

Kim, H. S., Paek, J., Culler, D. E., & Bahk, S. (2018). Do not lose bandwidth: Adaptive transmission power and multihop topology control. *Proceedings - 2017 13th International Conference on Distributed Computing in Sensor Systems, DCOSS 2017*, *2018-January*, 99–108. https://doi.org/10.1109/DCOSS.2017.23

Kumar, S., Andersen, M., Andersen, M. P., Kim, H.-S., Culler, D. E., & Culler, D. E. 2018. (2018). Demo Abstract: Bringing Full-Scale TCP to Low-Power Networks. *16th ACM International Conference on Embedded Networked Sensor Systems (SenSys 2018)*. https://doi.org/10.1145/3274783.3275196

Kumar, S., Andersen, M. P., Kim, H.-S., & Culler, D. E. (2018). *TCPlp: System Design and Analysis of Full-Scale TCP in Low-Power Networks*. https://www.researchgate.net/publication/328800924_TCPlp_System_Design_and_Analysis_of_Full-Scale_TCP_in_Low-Power_Networks

Kumar, S., Andersen, M. P., Kim, H.-S., & Culler, D. E. (2020). Performant TCP for Low-Power Wireless Networks. *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. https://www.usenix.org/conference/nsdi20/presentation/kumar

Kumar, S., Hu, Y., Andersen, M. P., Popa, R. A., & Culler, D. E. (2019). JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT. *Proceedings of the 28th USENIX Security Symposium*, 1519–1536. https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-sam

Wood, D. (2014). *Ethereum: a secure decentralised generalised transaction ledger*. https://www.semanticscholar.org/paper/ETHEREUM%3A-A-SECURE-DECENTRALISED-GENERALISED-LEDGER-

Wood/da082d8dcb56ade3c632428bfccb88ded0493214

Products Developed and Technology Transfer Activities

Publications

Andersen, M. P. (2019). *Decentralized Authorization with Private Delegation* [University of California Berkeley]. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-113.pdf

Andersen, M. P., Fierro, G., & Culler, D. E. (2017). Enabling Synergy in IoT: Platform to Service and Beyond. *Journal of Network and Computer Applications*, *81*, 96–110.

Andersen, M. P., Kim, H. S., & Culler, D. E. (2017). Demo abstract: Hamilton - a cost-effective, low power networked sensor for indoor environment monitoring. *BuildSys 2017 - Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, *2017-January*, 1–2. https://doi.org/10.1145/3137133.3141453

Andersen, M. P., Kolb, J., Chen, K., Culler, D. E., & Katz, R. (2017). Democratizing Authority in the Built Environment. *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys '17)*.

Andersen, M. P., Kolb, J., Chen, K., Fierro, G., Culler, D. E., & Popa, R. A. (2017). *WAVE: A decentralized authorization system for IoT via blockchain smart contracts*. https://people.eecs.berkeley.edu/~m.andersen/publications/wave2

Andersen, M. P., Kumar, S., AbdelBaky, M., Fierro, G., Kolb, J., Kim, H.-S., Culler, D. E., & Popa, R. A. (2019). WAVE: A Decentralized Authorization Framework with Transitive Delegation. *Proceedings of the 28th USENIX Security Symposium*, 1375–1392. https://www.usenix.org/conference/usenixsecurity19/presentation/andersen

Fierro, G., Raftery, P., Peffer, T., Culler, D. E., Baky, M. A., Thomson, G., & Pritoni, M. (2018). Mortar: An open testbed for portable building analytics. *BuildSys 2018 - Proceedings of the 5th Conference on Systems for Built Environments*. https://doi.org/10.1145/3276774.3276796

Fierro, Gabe. (2021). *eXtensible Building Operating System (XBOS) Overview - XBOS Docs*. https://docs.xbos.io/

Fierro, Gabe. (2017). Demo abstract: HodDB - a query processor for brick. *BuildSys 2017 - Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, *2017-January*. https://doi.org/10.1145/3137133.3141449

Fierro, Gabe, & Culler, D. E. (2018). Design and analysis of a query processor for brick. *ACM Transactions on Sensor Networks*, *14*(3–4). https://doi.org/10.1145/3199666

Fierro, Gabe, Guduguntla, S., & Culler, D. E. (2019). Dataset: An open dataset and collection tool for BMS point labels. *DATA 2019 - Proceedings of the 2nd ACM Workshop on Data Acquisition To Analysis, Part of SenSys 2019*, 40–42. https://doi.org/10.1145/3359427.3361922

Fierro, Gabe, Koh, J., Agarwal, Y., Gupta, R. K., & Culler, D. E. (2019). Beyond a House of Sticks: Formalizing Metadata Tags with Brick. *6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys)*.

Fierro, Gabe, Pritoni, M., AbdelBaky, M., Raftery, P., Peffer, T., Thomson, G., & Culler, D. E. (2020). Mortar: An Open Testbed for Portable Building Analytics (Journal Extension). *ACM Transactions on Sensor Networks*.

Kim, H.-S., Andersen, M. P., Chen, K., Kumar, S., Zhao, W. J., Ma, K., & Culler, D. E. (2018). System Architecture Directions for Post-SoC/32-bit Networked Sensors. *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems - SenSys '18*, 264–277. https://doi.org/10.1145/3274783.3274839

Kim, H. S., Ko, J., Culler, D. E., & Paek, J. (2017). Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey. In *IEEE Communications Surveys and Tutorials* (Vol. 19, Issue 4, pp. 2502–2525). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/COMST.2017.2751617

Kim, H. S., Kumar, S., & Culler, D. E. (2019). Thread/openthread: A compromise in low-power wireless multihop network architecture for the internet of things. *IEEE Communications Magazine*, *57*(7), 55–61. https://doi.org/10.1109/MCOM.2019.1800788

Kim, H. S., Paek, J., Culler, D. E., & Bahk, S. (2018). Do not lose bandwidth: Adaptive transmission power and multihop topology control. *Proceedings - 2017 13th International Conference on Distributed Computing in Sensor Systems, DCOSS 2017*, *2018-January*, 99–108. https://doi.org/10.1109/DCOSS.2017.23

Kumar, S., Andersen, M., Andersen, M. P., Kim, H.-S., Culler, D. E., & Culler, D. E. 2018. (2018). Demo Abstract: Bringing Full-Scale TCP to Low-Power Networks. *16th ACM International Conference on Embedded Networked Sensor Systems (SenSys 2018)*. https://doi.org/10.1145/3274783.3275196

Kumar, S., Andersen, M. P., Kim, H.-S., & Culler, D. E. (2018). *TCPlp: System Design and Analysis of Full-Scale TCP in Low-Power Networks*. https://www.researchgate.net/publication/328800924_TCPlp_System_Design_and_Analysis_of_Full-Scale_TCP_in_Low-Power_Networks

Kumar, S., Andersen, M. P., Kim, H.-S., & Culler, D. E. (2020). Performant TCP for Low-Power Wireless Networks. *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. https://www.usenix.org/conference/nsdi20/presentation/kumar

Kumar, S., Hu, Y., Andersen, M. P., Popa, R. A., & Culler, D. E. (2019). JEDI: Many-to-Many End-to-End Encryption and Key Delegation for IoT. *Proceedings of the 28th USENIX Security Symposium*, 1519–1536. https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-sam

Web site or other Internet sites that reflect the results of this project;

http://btrdb.io/index.html

https://docs.xbos.io/

Networks or collaborations fostered

We contributed to RiotOS and OpenThread

We presented WAVE 3 at HDSI IoT Security and Privacy Workshop.

Michael Andersen presented in a panel on security at the Energy Exchange Building Summit in Cleveland, OH in late August 2018.

Dr. Andersen worked for PingThings as the Chief Technology Officer (https://www.pingthings.io/). Ping Things has commercialized BTrDb and is using it in their business with electrical utility monitoring and analysis.)

Dr. Hyung Sin Kim, post doc on Hamilton project, worked for Google, now is a professor in Korea.

Technologies/Techniques

Hardware

Hamilton: sensor hardware (and data aggregation). HamiltonIOT continues to sell Hamilton hardware: https://hamiltoniot.com/; the sensor platform led to advancements in RiotOS and OpenThread.

Software

BTrDB (Berkeley Tree Database): high speed database: http://btrdb.io/index.html

WAVE and WAVEMQ: secure middleware was used in the eXtensible Building Operating System (XBOS) (https://docs.xbos.io/), FlexHP (LBNL-led project, funded by CEC) and OpenBOS-NY (TRC-led project, funded by NYSERDA). Another opensource tool supporting WAVE is JEDI (Joining Encryption and Delegation for IoT): end to end encryption for WAVE (github.com/ucbrise/jedi-protocol-go).

TCP/IP and Thread: TCP/IP work by PhD student Sam Kumar, including MAGE: Efficient Execution Engine for Secure Multi-Party Computation.

Brick schema and Mortar: metadata tagging and query

Inventions/Patent Applications, licensing agreements

Open source licenses for BTrDb

Other products

BTrDb