Proteus: Towards a Manageability-focused Home-based Health Monitoring Infrastructure

Mengjing Liu*, Mohammed Elbadry*, Yindong Hua*, Zongxing Xie, Fan Ye
{mengjing.liu,mohammed.elbadry,yindong.hua,zongxing.xie,fan.ye}@stonybrook.edu
Electrical and Computer Engineering
Stony Brook University, NY, USA

ABSTRACT

A data collection infrastructure is vital for generating sufficient amounts and diversity of data necessary for developing algorithms in home-based health monitoring. However, the manageability—deployment and operation efforts—of such an infrastructure has long been overlooked. Even a small size of a dozen homes may incur enormous manual efforts on the research team, including installing, configuring and updating of sensor, edge devices; continuous monitoring for faults and errors to prevent data losses, and integrating new sensing modalities. In this paper, we present Proteus, an easily managed infrastructure designed to automate much of the work in deploying and operating such systems. Proteus includes: *i)* scalable, continuous deployment and update of devices with automatic bootstrapping; *ii)* automatic fault and error monitoring and recovery with watchdogs and LED feedback, and complementary edge and cloud storage backups; and *iii)* an easy-to-use data-agnostic pipeline for integrating new modalities. We demonstrate our system's robustness through different sets of experiments: 3 sensor nodes running for 24 days sending data (17.3 Mbps aggregate rate), and 16 emulated sensors (92.8 Mbps aggregate rate). All such experiments have data loss rates less than 1%. Further we reduce human efforts by 25-fold and code required for adding new data modality by 25-fold. Our results show that Proteus is a promising solution for enabling research teams to effectively manage home-based health monitoring at small to medium sizes.

CCS CONCEPTS

• **Computer systems organization → Data flow architectures**.

ACM Reference Format:
Mengjing Liu[*], Mohammed Elbadry[*], Yindong Hua[*], Zongxing Xie, Fan Ye. 2023. Proteus: Towards a Manageability-focused Home-based Health Monitoring Infrastructure. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23), September 3–6, 2023, Houston, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3584371.3613006

1 INTRODUCTION

Home-based health monitoring has the potential to revolutionize the way we manage our health. The continuous collection and analysis of multi-modal sensing data could lead to early and precise interventions and management of a broad spectrum of diseases [8]. However, the development of robust, generalizable algorithms and models critically depends on the amount and diversity of data from real homes, not well controlled lab environments [4]. The lack of a *manageable* infrastructure that can easily collect such data from real homes is a fundamental barrier to practical home-based health monitoring.

The *manageability* in deploying and operating such an infrastructure is a great challenge, especially for most research teams of small sizes (e.g., one faculty member plus a few students). Significant manual efforts and special expertise are needed throughout all stages of the process: *i)* configuring sensor, edge devices in multiple homes, installing data collection and analytic software to create the testbed, and constantly updating the algorithms and models running on the sensor, edge and cloud upon improved software or changed health conditions; *ii)* automatic monitoring of the system status and performance, and timely detection, resolution of network, device faults and errors to ensure minimal data loss in 24/7 data collection; and *iii)* integrating heterogeneous sensing hardware, modalities with varying data formats and rates (e.g., from Kbps of activities to tens of Mbps of RF baseband) which may require separate and distinct engineering efforts for data streaming, processing, and storage.

While there exist a few research infrastructures for health monitoring purposes (e.g. [1, 2, 5, 6, 10, 14]), they do not consider manageability issues of intensive human efforts required in deployment (configuring, installing) and operation (updating, troubleshooting), and they support mainly traditional sensors (e.g. video, wearable devices, ambient sensors such as temperature and humidity) or sensors for specific diseases (e.g. breath sensor for asthma monitoring), not easy to extend and add new modalities. Existing data collection or IoT infrastructures from industry (e.g. KAA [13], Thingspeak [9]) support new modalities by data-agnostic transportation and storage utilizing MQTT [7] and HTTPS protocols. However, they mostly focus on cloud-side features such as data transfer, storage and analysis. Intensive manual efforts for configuring sensor, edge devices, and frequent update of algorithms or models needed for home health-monitoring, are not addressed. In addition, these works do not consider sufficiently the faults, errors that happen frequently (e.g. network outage, device faults) in home environments, causing serious data losses and manual efforts to remedy.

In this paper, we present Proteus [1] , a manageability-focused data collection infrastructure that minimizes manual efforts so a small research team can efficiently deploy and operate longitudinal home-based health monitoring. To minimize human efforts, we develop new design components, and identify, combine mature technology pieces, engineering practices to optimize the overall

BCB '23, September 3–6, 2023, Houston, TX, USA

ACM ISBN 979-8-4007-0126-9/23/09…$15.00
https://doi.org/10.1145/3584371.3613006

[1]This work is supported in part by NSF grants 1951880, 2119299.

Table 1: Comparison with existing work. Our proposed system (in cyan), Proteus, advances the state of the art by enabling a comprehensive set of functions as a whole to allow research teams to effectively manage home-based health monitoring.

System	[6]	Welcome [2]	SPHERE [14]	KAA [13]	ThingSpeak [9]	Proteus
In-home	✗	✓	✓	✓	✓	✓
Automatic Installation	✗	✗	✗	✗	✗	✓
Automatic Connectivity	✗	✗	✗	✗	✗	✓
OTA Deployment/Update	✗	✗	✓	✗	✗	✓
Edge Status Monitoring	✗	✗	✓	✓	✓	✓
Watchdog on Edge/Cloud	✗	✗	✗	✗	✗	✓
Complementary Edge Storage	✗	✗	✗	✗	✗	✓
Data-agnostic Pipeline	✓	✗	✓	✓	✓	✓
End-to-end Data Loss	—	—	—	—	—	< 1%
End-to-end Latency	—	—	—	—	—	< 3.65 seconds
Reduction in Manufacturing Time	—	—	—	—	—	25-fold

workflow, including: *i)* a scalable, continuous deployment and management pipeline with IoT device management solution (DMS) for remote, batch deployment and frequent update, automated node registration and networking setup upon first boot, and automated connection on edge; *ii)* automatic monitoring and watchdog mechanisms on edge for timely recovery from errors, and complementary edge storage backup for resilience against network failures; and *iii)* a data-agnostic pipeline that uses publish-subscribe (pub-sub) to transfer heterogeneous data formats to the cloud and store the data in database appropriately.

Preliminary experiences show that we can achieve 25-fold reduction in manual configuration of 7 sensor nodes from 420 minutes to 17 minutes. A small scale stress test shows that Proteus works reliably with 16 simulated sensor nodes, collecting data at an aggregate rate of 92.8 Mbps, with negligible data loss. The infrastructure we deploy in a one-bedroom, one-bathroom simulated home environment runs 3 real sensor nodes continuously for 24 days, and 10 real sensor nodes for 2 weeks without glitches. Despite many faults and errors (e.g. 6 sensor nodes running for 1 month, 2 network outages, 11 low data transfer rates, lasting 2 hours in total), the watchdog mechanisms can always quickly restore the normal operation, leading to minimal data loss (< 1%). Results show that greatly improved manageability of Proteus enables small research teams to efficiently deploy and operate longitudinal, continuous data collection systems.

We summarize our contributions as follows:

- We identify three sources of intensive manual efforts in manageability challenges for deploying and operating longitudinal home-based health monitoring: *i)* configuring sensor nodes and edge servers, installing and updating their software and analytics; *ii)* detecting, resolving network and hardware faults and errors to ensure minimal data loss; *iii)* integrating new, unforeseen data modalities.
- We design new components and identify, combine mature techniques to minimize human efforts for greatly improved manageability for small research teams to deploy and operate such an infrastructure, including *i)* automated, continuous deployment, operation and update of the infrastructure with automatic self-bootstrapping; *ii)* automatic monitoring and recovery with watchdogs and complementary edge storage, requiring minimal manual efforts while ensuring minimal data loss; and *iii)* easy integration of new, unforeseen modalities with a data-agnostic pipeline.
- We have deployed the infrastructure in a simulated home environment. We find the manual configuring efforts are cut down by 25 fold, and lines of code for new modality integration is cut down by 25 fold. 10 real sensor nodes run continuously for 2 weeks without any errors, and 16 simulated nodes collecting data at an aggregate rate of 92.8 Mbps have negligible losses.

2 BACKGROUND AND RELATED WORK

We identify and provide a brief description of existing technologies that are critical for designing Proteus.

Enterprise Device Management Solutions. There exist enterprise solutions [11] for IoT device management and field deployment, like Amazon's AWS Greengrass IoT, Azure IoT device Management, which provide services like monitoring node status remotely with a cloud dashboard, pushing updates to remote nodes, containerized deployment of codes, version control, watchdog services for automatic quick-recovery, and etc.

Containers and Microservices. Containerizing the code (e.g., docker) is a popular cloud technology that packages a full userspace environment with complex library, code dependencies so it can be accurately and easily duplicated in large quantities with simple scripts in forms of microservices.

Pub-Sub Communication. A pub-sub transport provides robust asynchronous communication where data is cached and resent under intermittent connections, and flexibility to support data of different formats/content as opaque loads tagged with different topics. There exist mature solutions for constrained nodes at edge (e.g., MQTT). With Quality of Service (QoS) 2, MQTT guarantees once and only once data delivery over network.

Related Work. While these technologies help with manageability of the intended infrastructure, building and running such a system still requires significant efforts in an error-prone environment with imperfect embedded devices. We categorize and compare the related work in Table 1.

Home-based data collection infrastructure has been created, but not focused on manageability [3]. VitalCore [1] has developed an analytics and support dashboard and eliminated the tall pipeline of reading HL7 format health data. Multi-modal sensor infrastructure [14] supports various sets of data sent to the cloud, but does

*These authors contributed equally to this work

not handle edge computing, or maintaining units in the field. [2, 6] focus on clinical data and device management. However, they are not specifically designed for home-based health monitoring, thus intensive human efforts for installing, monitoring devices and remedy on failures (e.g. network outage, hardware failures) which can easily occur in home environments are not considered. Furthermore, they do not provide a quantitative estimation of the infrastructure's robustness and efficiency. Many such data collection works could use Proteus to reduce manual efforts for greatly improved manageability (e.g., Parkinson data collection [12]).

There are also existing IoT platforms from industry (KAA [13], Thingspeak [9]), which support multi-modal data transportation with MQTT or HTTPS protocols, and various data analysis and visualization tools on cloud. However, they do not consider manual efforts in configuring, installing individual sensor, edge devices, and detecting, troubleshooting for faults, errors (e.g. network outage, device faults) in edge environments, which are fundamental barriers for small research teams to conduct longitudinal home-based health monitoring.

3 SYSTEM DESIGN

3.1 Goals and Assumptions

3.1.1 Goals. The goals of our infrastructure design are to minimize manual efforts of a research team in different stages: *i)* deploying the infrastructure for initial setup and continuous operating for data collection and analysis. *ii)* detecting network, device faults and errors, and recovering timely to facilitate longtime, continuous running with minimum data loss. and *iii)* coding work on the pipeline to integrate multiple, possibly unforeseen types of sensors (e.g., base band radio data) for health monitoring needs.

3.1.2 Assumptions. We make the following assumptions: *i)* there exists sufficient wireless connection to the Internet at homes to handle the data throughput; *ii)* most of the data to be collected are time series and events (e.g., vital signs, activities with timestamps) that can be stored in relational and time series databases (e.g., MySQL, InfluxDB); and *iii)* for certain sensitive data, the user may prefer storage, processing locally on edge servers, not cloud.

3.2 System Overview

In this section, we first describe the 3-layer framework of Proteus including the data path and control path of the pipeline. We then provide a comprehensive description of the efforts required to deploy and operate the infrastructure, and our design to minimize such efforts.

Figure 1 shows the three-layer structure of Proteus and its data and control paths. In each home (i.e., edge), embedded systems (e.g., Raspberry Pi's) gather data directly from connected sensors (e.g., Ultra-Wide Band (UWB) radios, or Neulog sensors via USB), and send data through the home WiFi to an edge server. The edge server, a data transfer gateway, aggregates data from all sensor nodes in one home, pushes data to the cloud. It temporarily stores the data in a local database when the Internet is disconnected, and resends new incoming data upon reconnection. Sensitive data not allowed to leave the home is also stored locally. Further, analytic algorithms and machine learning models can be deployed on the edge server, enabling edge processing whenever needed. After data is sent to a cloud MQTT broker in a DMZ (Demilitarized Zone facing public

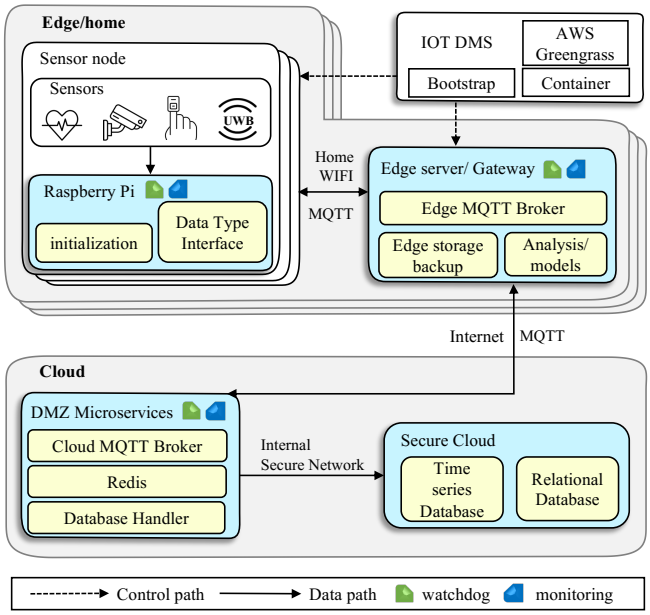

Figure 1: Proteus Design Overview. In each home, there are multiple Raspberry Pi's connected to sensors. They use MQTT to publish readings to an Edge Server Gateway, which then aggregates the data and sends it to the cloud MQTT Broker which stores it in databases appropriately. Both edge and cloud have watchdogs, cloud monitoring and Over the Air (OTA) update capabilities.

Internet and blocking potential malicious traffic), some microservices receive and push the data to appropriate databases (either time-series or relational) in the secure cloud behind DMZ. The control path consists of two parts: *i)* IoT DMS to remotely deploy and update containerized codes and models on edge, sensor nodes, *ii)* System-wide monitoring and watchdogs to monitor system status and performance to ensure timely detection and recovery from faults, errors (e.g. network outage).

We divide our design into three parts: *i) Automated and Continuous Deployment* to minimize human efforts in the setup of the infrastructure, *ii) Maintenance* to monitor system performance, detect and restore from unexpected faults, error automatically, and *iii) Data-agnostic Pipeline* to support easy integration of new modalities.

3.3 Automated and Continuous Deployment

The deployment of the infrastructure can be divided into three phases:

Installation. Manually installing operating systems and software (e.g. Docker for containerized code, software for LED control and WiFi configuration, etc.), or setting up the environment for an algorithm (especially a deep learning model) for embedded systems, are all laborious. We estimate that it takes an expert familiar with embedded systems 10 minutes to set up a sensor node. Instead of repeating such manual effort for every single sensor node, we duplicate them based on a *"prep"* master image with all the needed packages using a flash cloning machine, which costs only 3 minutes to duplicate 7 units in a batch.

Connectivity. After installation, we need to configure the network settings of sensor devices so they can find the IP of edge server and establish pub/sub connections for data transmission. We reduce configuration time by *i) automated WiFi set up.* We configure

the *"prep"* image to open a WiFi portal on boot so that we can connect to it using a mobile App and send over the WiFi identifier and password of the Home WiFi network to embedded systems of sensor nodes. *ii) Dynamic Edge Server Connection.* To establish pub sub connection between the sensor nodes and edge server, the only information needed is the IP address of the broker, which is located in the edge server and can change over time. To achieve dynamic IP discovery mechanism, we create a lookup table in the cloud relational database to associate the edge server of each home with its MAC address, which is generally considered to be static. To establish pub sub connection, the sensor nodes only need to query the relational database to get IP address of the broker in the corresponding home. In addition, we configure the home ID of the home where the sensor is deployed in the *"prep"* image for record.

OTA Deployment and Update. Deploying code (e.g., algorithms and models) across tens to hundreds of sensors and edge devices is labor-intensive. The workload is compounded by constant code update on sensors and edge upon bug fixes, software improvements, or changing health conditions. To alleviate such manual efforts, we build on enterprise DMS (e.g., AWS Greengrass) which has a dashboard to easily deploy and update code on many sensor nodes and edge servers via a few clicks. However, for a sensor device to be managed by a DMS, it needs to register and obtain its unique ID from the DMS. We develop an automatic self-registration script, where upon the first boot, the script generates a unique ID for the edge device, registers the device to the DMS, and creates a record in our relational database (including its unique ID and the home ID) for subsequent tracking with the home ID stored in the *"prep"* image beforehand. Then the self-registration of dozens nodes in one home can be done in parallel automatically on first boot without any manual efforts.

3.4 Maintenance

Our system has multiple embedded devices and multi-hop network transport. Many problems (e.g. network interruption, accidental unplugging of sensors, hardware failures) may occur in a home environment, resulting in critical data loss. Although software such as AWS Greengrass and PM2 provides dashboard for remote monitoring of device status and watchdogs to restart programs if they exit abnormally, they do not provide detailed per-device performance metrics (e.g. data transfer rate per hop) which are significant for gaining more insights of system performance and troubleshooting. Furthermore, when such application level restart cannot bring the system back, more action (e.g. rebooting the device) is required.

Thus, we design watchdogs combined with LED chipsets for automatic monitoring and recovery to detect, notify (with different LED lighting modes) and remedy on *i) network issues* and *ii) hardware issues* in time to ensure seamless operation of the infrastructure.

Network Issues. We have a local database on the edge server to temporarily store data when Internet is disconnected. However, there are network issues on sensor nodes that can cause data loss before data is transferred to the edge server, requiring extra efforts to operate correctly.

With pub-sub communication, when data publishing fails due to network issues, during which the data will be kept in the publisher queue in the sensor node memory, waiting to be re-published until the publishing (i.e., four-way handshake) finishes successfully. While new data keeps coming in, data piles up in the queue. Once the maximum queue size is reached, new data cannot be appended in the queue and are dropped. Thus, we identify two key metrics to monitor sensor nodes: memory usage and publisher queue size over time, which can reflect data transfer rates and reveal abnormal network issues occurring.

We monitor the network connection, notify users with an LED light pattern (blue light) and reboot the device to recover if the network outage lasts longer than a threshold. For low data transfer rates, we monitor the publisher queue size and reboot the device when publisher queue reaches the maximum limit (configured corresponding to memory limit) and data transfer rate remains low within a time window.

Hardware Failures. Multiple hardware failures (sensors being unplugged, power supply failure, file system corruption and edge server going offline) can happen on edge, requiring mandatory human operations. We integrate the embedded system with an LED chipset whose lighting modes can indicate the sensor states, aiding non-expert users and the research team to conduct troubleshoot and remedy on errors without physical travel. The states and corresponding recoveries are *i)* Green: the sensor device is running normally, no operation required; *ii)* Red: the sensor needs to be re-plugged. *iii)* Yellow flashing: the sensor node can not discover the edge server MAC address, requiring to record edge server MAC in relational database. *iv)* Blue flashing: the sensor node can not discover the edge server IP address, requiring to connect edge server to home WiFi. *v)* Red flashing: the sensor node can not connect to the broker on the edge server, requiring to reboot the edge server. *vi)* LED is off: file system corruption, requiring OS image reflash for recovery. *vii)* A power light off on sensor nodes indicates a power failure and requires the power supply to be re-plugged.

More events can be supported by additional lighting patterns of the LEDs. Compared to having to read logs to analyze the problem, which is cumbersome, and feasible only by the research team, LED feedback offers an easy, direct visual means for a few most common problems, allowing even non-expert users to help the research team troubleshoot remotely (e.g., via phone calls).

3.5 Data-agnostic pipeline

The utilization of heterogeneous sensing devices and modalities is essential to meet comprehensive research needs and effectively monitor home health. While data transmission over the network independent of the data type can be easily achieved with pub sub transportation, extra effort is needed to deal with storage of data from heterogeneous sensing modalities to facilitate subsequent data analysis. We design *interfaces for data types* to minimize the required coding effort.

When adding a new sensor modality, the developer will need to update code for streaming data from the sensor (according to respective libraries), and passing the opaque, tagged data in the format they wish to store in the database. Two categories of metadata are defined to accommodate the new data type: *i) database type and name*, which instructs the pipeline the type of the database (i.e. time-series, relational) and name of database to store data within; *ii)* tags (i.e., topics) to store each respective field (e.g. "*baseband_timestamp*" indicates two tags to store baseband data field and timestamp field of when the data is generated) appropriately within the database. We use the database tags as topics in pub sub

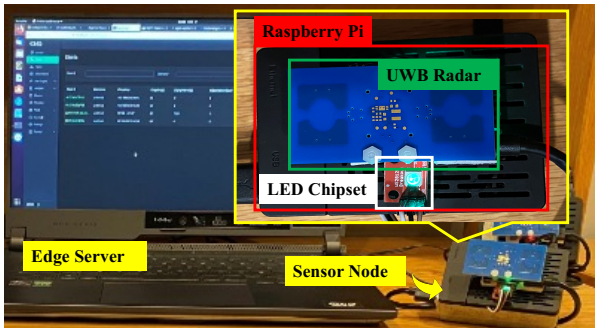

Figure 2: Each home has a gaming laptop based edge server and multiple sensor nodes each consisting of a Raspberry pi, UWB sensor and LED chipset.

transportation, so only microservices subscribing on those topics for respective databases receive the corresponding data.

4 EVALUATION

We implement our infrastructure's cloud components on a HIPPA-compliant data center using standard virtual machines running Red Hat Enterprise Linux 8.4.1. For each sensor node, we use a commodity Raspberry Pi 3B+ connecting a Ultra-Wide Band (UWB) radio sensor (5.8 Mbps data rate of baseband frames) and a LED chipset. We take gaming laptops (Intel Core i7, 16GB RAM, 1 GPU, 512 GB hard drive) as edge servers. We deploy one MQTT broker on the edge server to aggregate data from all sensors per home, and another in cloud for all homes. We leverage AWS Greengrass to control the sensor/edge devices and push containerized updates to them from a cloud dashboard. We deploy InfluxDB as our time-series database and MySQL as relational database. We configure our sensor nodes and edge server (Raspberry Pi's and laptops) with automatic bootstrap and connectivity to minimize human efforts.

We summarize the comparison between Proteus and existing home-based health monitoring infrastructures and IOT platforms in Table 1. We are the first to introduce a manageability focused infrastructure that minimizes manual efforts in deployment and operation. This enables small research teams to efficiently manage longitudinal home-based health monitoring systems. Furthermore, we conduct quantitative evaluations to assess the end-to-end loss, latency, and reduction in manufacturing time, providing concrete evidence of the robustness and efficiency of our infrastructure.

4.1 Automated and Continuous Deployment

25-fold reduction in manufacturing time from 420 minutes to 17 minutes. Originally, setting up 7 sensor nodes costs 420 minutes (60 minutes per unit in total, a rough estimate of the effort, including 10 minutes for OS and software installation, 5 minutes WiFi setup, 30 minutes dynamic edge server connection, 3 minutes deploy code on edge, 12 minutes update code on edge. Empirically, home WiFi breaks 1/day, thus 30 edge server connections/month. Code updates 1/week, thus 4/month). Through configuration and batch OS flash image cloning, automatic WiFi setup and connectivity, automated self-bootstrapping and OTA update, we reduce 420 minutes to 17 minutes (itemized as OS image cloning 3 minutes, WiFi setup 7 minutes, dynamic edge server connection 2 minutes, code deployment 1 minute and code update 4 minutes per month for 7 nodes in total). This saves significant time and makes it manageable by a small research team to deploy dozens of homes in a matter of hours.

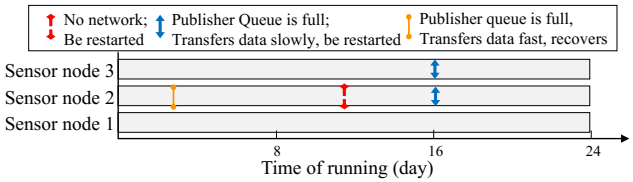

Figure 3: Abnormal events occur on sensor nodes during a 24-day run.

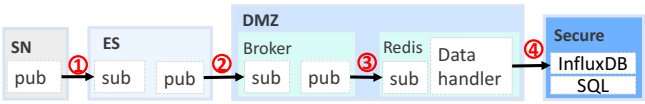

Figure 4: The hop-by-hop data flow from the sensor node to the cloud database.

4.2 Maintenance

During infrastructure testing, abnormal events occur on several sensor nodes, but our watchdogs are able to recover them from errors and ensure continuous operation. In Figure 3 we present the events on three of the sensor nodes through 24 days of running. Sensor node 1 runs smoothly with no issues, while on sensor node 2, the publisher queue is full after 2.5 days briefly (no reboot or issue) and the node is rebooted after 11.5 days due to a network disconnection. After 16 days, node 2 is rebooted due to having publisher queue full for over 10 minutes. Meanwhile, sensor node 3 encounters the same event and is rebooted automatically. Our results confirm that multiple issues happen in continuous operating systems in the field, and that our watchdog can recover the nodes and prevent data loss (0.03% data loss on edge in 24 days).

During running, we also observe 2 times of file system corruptions on sensor nodes and 2 times of edge server going offline because it has been running for months without being restarted. Due to our monitoring with LED, we can be notified of such hardware failures and remedy timely to prevent data loss.

4.3 Easy integration of new modality (25× reduction in LOC)

We demonstrate the adaptability of our pipeline to support new data modalities. For time-series data like UWB baseband data and Neulog sensor data, we pack their data into MQTT messages for data transfer and store them in corresponding measurements in InfluxDB on cloud. For example, for Activities of Daily Living (ADL) recognition, the sensor nodes collect baseband data (containing all original information for developing better ADL recognition algorithms) from UWB sensors and package them with timestamps into MQTT messages to send under the topic "*BasebandData_timestamp/Node_ID*". On the cloud, a measurement named "*BasebandData_timestamp/Node_ID*" in InfluxDB should be created, including "*BasebandData*" and "*timestamp*" fields so that the data handler can read messages and write timestamps and baseband data into the database correctly. With our system, we can implement it within 4-5 LOC (2-3 LOC parse data from sensor, and 2 LOC pass it to our API) instead of hundreds of LOC :15 LOC to create a publisher or subscriber, 2 LOC to publish and subscribe to topics of interest, 20 LOC on the subscriber side to parse messages and decide what to do with them (e.g. push to the next hop or store in edge database), 55 LOC to parse data and convert it to the appropriate format before writing to the database, 15 LOC to write data into database in multi-threading. If programming languages used on the edge and cloud are different, the LOC count can easily double.

Table 2: Data Loss Rate Hop by Hop

# of SN	R	ES	Sub	Red	DB	EDB	T
4	23.2	0.05%	0	0	0.06%	8%	15D
7	40.6	0.27%	0	0	0.26%	12%	6D
10	58.0	0.5%	0.17%	0	0.03%	3.8%	14D
16	92.8	0	0	0	0.10%	1.4%	2H

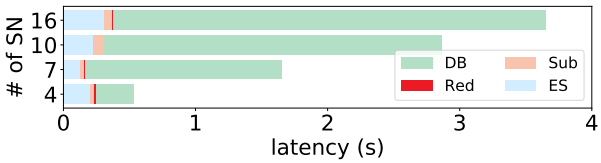

Figure 5: Average Latency Hop by Hop. With 16 simulated sensor node connections sending data at 92.8 Mbps, the end-to-end latency is 3.65 seconds, slightly higher but still enough for home health monitoring (e.g., fall detection).

4.4 Overall Performance

We evaluate the overall performance of the infrastructure in terms of the data loss and latency with different number of sensor nodes running concurrently for an extended period of time. Each sensor node sends 1 MQTT message per second, including 80 UWB base-band data frames, where the message data size is 0.72MB. In addition, we run multiple threads simultaneously on a gaming laptop to simulate multiple sensor node connections to stress test the system for data loss and latency. We estimate the data loss and latency from one hop to the next to get a insight of the system performance. The data flow and 4 hops of the infrastructure are shown in Figure 4. We estimate the data loss and latency hop-by-hop between the publishers on sensor nodes (SN), subscriber on the edge server (ES), subscriber of broker on cloud (Sub), subscriber of Redis on cloud (Red) and InfluxDB (DB) on cloud. We measure the loss and latency with different data rates (R) in Mbps, numbers of sensor nodes, and continuous running times (T, in the units of days denoted by "D" and hours "H"). We summarize the test results of data loss in Table 2, including the ratio of data temporarily stored in the edge database as a percentage of all data (EDB). Results for 4, 7, and 10 nodes are from real sensor nodes, and the others are node connections from multithreaded simulations.

Results in Table 2 show that Proteus can support data collection with 16 sensor nodes sending data at 92.8 Mbps aggregate rate, with negligible data loss (0.10% end-to-end), sufficient for the needs of home health monitoring. In the test with 10 real sensor nodes running for 2 weeks, we observe 0.7% end-to-end data loss. Given the resilient pipeline and watchdogs, there are negligible data loss between SN and ES during rebooting of sensor nodes. In our preliminary experiment using 3 sensor nodes for 40 days, we observe no data loss and no errors from sensor nodes to the edge server. Results show that Proteus is reasonably stable and promising for continuous, longitudinal data collection.

EDB in Table 2 demonstrates the fraction of data backup on edge server at up to 12% upon network failures, which proves such backup storage is necessary. We also conduct offline tests (e.g., manually disconnecting the network of the edge server for 2 hours) and observe no data loss because of edge storage.

Figure 5 shows the average hop-by-hop latency with real and simulated sensor nodes. In the 15-day run test with 4 real sensor nodes, the average latency from sensor node to each hop of the infrastructure is 0.2, 0.23, 0.24, 0.53 seconds respectively. With more sensor nodes and higher data rates, we observe the end-to-end latency slightly increases due to higher data transfer and storage requirements. But even with 16 simulated sensor nodes to connect to the system and send data at 92.8 Mbps, we observe 3.65 seconds end-to-end latency, still sufficient for home health monitoring (e.g., detecting falls within seconds).

5 CONCLUSION

In this paper, we present our experience in developing and pilot deploying an infrastructure for home-based health monitoring. We recognize the human efforts needed in the full cycle of deploying and operating such infrastructure. Our system cuts down on labor efforts by 25X and reduces code necessary for deployment by 25X. Our system ran end to end (from sensor on edge to database on cloud) and had 0.7% data loss over 14 days experiment. We share experiences and lessons hopefully valuable for other research teams, and we plan to open-source the infrastructure once it becomes mature.

REFERENCES

[1] Hyonyoung Choi, Amanda Lor, Mike Megonegal, Xiayan Ji, Amanda Watson, James Weimer, and Insup Lee. 2021. VitalCore: Analytics and Support Dashboard for Medical Device Integration. In *IEEE/ACM CHASE 2021*. IEEE, 82–86.

[2] Ioanna Chouvarda, Nada Y Philip, Pantelis Natsiavas, Vasilis Kilintzis, Drishty Sobnath, Reem Kayyali, Jorge Henriques, Rui Pedro Paiva, Andreas Raptopoulos, Olivier Chetelat, et al. 2014. WELCOME—innovative integrated care platform using wearable sensing and smart cloud computing for COPD patients with comorbidities. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 3180–3183.

[3] Advantech Co. 1983-2023. Advantech Co-Creating the Future of the IoT World. Retrieved August 7, 2017 from https://www.advantech.com/en-us

[4] Prafulla N Dawadi, Diane J Cook, and Maureen Schmitter-Edgecombe. 2013. Automated cognitive health assessment using smart home monitoring of complex tasks. *IEEE transactions on systems, man, and cybernetics: systems* 43, 6 (2013), 1302–1313.

[5] Matti Hämäläinen, Lorenzo Mucchi, Stefano Caputo, Lorenzo Biotti, Lorenzo Ciani, Dania Marabissi, and Gabriele Patrizi. 2021. Ultra-wideband radar-based indoor activity monitoring for elderly care. *Sensors* 21, 9 (2021), 3158.

[6] Andrew Hornback, Wenqi Shi, Felipe O Giuste, Yuanda Zhu, Ashley M Carpenter, Coleman Hilton, Vinieth N Bijanki, Hiram Stahl, Gary S Gottesman, Chad Purnell, et al. 2022. Development of a generalizable multi-site and multi-modality clinical data cloud infrastructure for pediatric patient care. In *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. 1–10.

[7] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. 2008. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In *COMSWARE'08*. IEEE, 791–798.

[8] Yingcheng Liu, Guo Zhang, Christopher G Tarolli, Rumen Hristov, Stella Jensen-Roberts, Emma M Waddell, Taylor L Myers, Meghan E Pawlik, Julia M Soto, Renee M Wilson, et al. 2022. Monitoring gait at home with radio waves in Parkinson's disease: A marker of severity, progression, and medication response. *Science Translational Medicine* 14, 663 (2022), eadc9669.

[9] The MathWorks. 2023. IoT Analytics - ThingSpeak Internet of Things. Retrieved June 10, 2023 from https://thingspeak.com/

[10] Ho-Kyeong Ra, Asif Salekin, Hee Jung Yoon, Jeremy Kim, Shahriar Nirjon, David J Stone, Sujeong Kim, Jong-Myung Lee, Sang Hyuk Son, and John A Stankovic. 2016. Asthmaguide: an asthma monitoring and advice ecosystem. In *2016 IEEE Wireless Health (WH)*. IEEE, 1–8.

[11] Arvind Ravulavaru. 2018. *Enterprise Internet of Things Handbook: Build end-to-end IoT solutions using popular IoT platforms.* Packt Publishing Ltd.

[12] Shehjar Sadhu, Dhaval Solanki, Nicholas Constant, Vignesh Ravichandran, Gozde Cay, Manob Jyoti Saikia, Umer Akbar, and Kunal Mankodiya. 2022. Towards a telehealth infrastructure supported by machine learning on edge/fog for Parkinson's movement screening. *Smart Health* (2022), 100351.

[13] KaaIoT Technologies. 2023. Enterprise IoT Platform with Free Plan | Kaa. Retrieved June 10, 2023 from https://www.kaaiot.com/

[14] Przemyslaw Woznowski, Xenofon Fafoutis, Terence Song, Sion Hannuna, Massimo Camplani, Lili Tao, Adeline Paiement, Evangelos Mellios, Mo Haghighi, Ni Zhu, et al. 2015. A multi-modal sensor infrastructure for healthcare in a residential environment. In *IEEE ICCW 2015*. IEEE.