The Future Internet of Things: Secure, Efficient, and Model-Based

Joshua E. Siegel, *Member, IEEE*, Sumeet Kumar, and Sanjay E. Sarma, *Member, IEEE*

***Abstract*—The Internet of Things' (IoT's) rapid growth is constrained by resource use and fears about privacy and security. A solution jointly addressing security, efficiency, privacy, and scalability is needed to support continued expansion. We propose a solution modeled on human use of context and cognition, leveraging cloud resources to facilitate IoT on constrained devices. We present an architecture applying process knowledge to provide security through abstraction and privacy through remote data fusion. We outline five architectural elements and consider the key concepts of the "data proxy" and the "cognitive layer." The data proxy uses system models to digitally mirror objects with minimal input data, while the cognitive layer applies these models to monitor the system's evolution and to simulate the impact of commands prior to execution. The data proxy allows a system's sensors to be sampled to meet a specified quality of data target with minimal resource use. The efficiency improvement of this architecture is shown with an example vehicle tracking application. Finally, we consider future opportunities for this architecture to reduce technical, economic, and sentiment barriers to the adoption of the IoT.**

***Index Terms*—Emerging technologies, Internet of Things (IoT), networking architecture.**

I. Introduction and Contents

The INTERNET of Things (IoT) is a term describing a system of connected people, devices, and services [1]. The IoT allows computer-interfaced sensors and actuators to facilitate novel products and services by reducing costs, improving efficiency, and enhancing the usability of existing systems.

The benefits of connectivity are understood across industries, with connected cars and homes, smart factories, wearable devices, and intelligent infrastructure signaling the widespread adoption of the IoT. Few technical, economic, and social barriers, like support costs [2] and concerns about data privacy and system security, [3] limit this technology's opportunity space.

Today, power and bandwidth consumption challenge IoT's growth. The desire for rich data and information sharing dominates resource use, particularly challenging battery life and network loading for distributed wireless devices [4], [5]. A simultaneous proliferation of high-value connected devices makes the IoT a desirable attack surface [2], [6] and drives security-related resource requirements, demanding high-powered computation—lest a platform become unfavorable for mission-critical applications.

This paper builds upon the Siegel's dissertation [7] to demonstrate an approach leveraging scalable cloud resources to address efficiency, privacy, and security for next-generation IoT.

In Section II, we identify a need for IoT architecture improving system-wide efficiency and security, and discuss contemporary research. Then, we consider how people process, share, and protect information in Section III. In Section IV, we present a human-inspired model for data collection, synthesis, distribution, and protection. We develop a parallel IoT architecture utilizing process and measurement knowledge to reduce the cost of sampling sensors and transmitting data. This approach leverages system knowledge to provide security through abstraction and data privacy through remote sensor fusion.

We outline five enabling components of this architecture, and present the key innovations of "data proxies" and "cognitive layers" in detail. The data proxy is a model-based means of digitally mirroring objects using minimal input data, while the cognitive layer utilizes these same models to monitor system evolution and to simulate the impact of commands. Supporting the data proxy, we introduce the concept of the "quality of data," (QoD) a formalized quantitative metric used for intelligent resource management capable of assuring a high level of connected application performance. In Section V, this architecture's improvement on security and resource efficiency are demonstrated through an example application calculating vehicle distance traveled with sparse input data. Finally, Section VI concludes with a brief discussion and future work identification.

II. Prior Art

If one considers IoT as a design vocabulary, it necessarily must possess an alphabet of development considerations and enabling technologies. IoT's "ABC's" consider privacy, security, and resource efficiency, with a connected system's "A's" (safeguarded actuators and protected attributable data) ensuring a solid foundation for data storage, sharing, and use, and the "B's" and "C's" referring to the resource constraints of battery, bandwidth, bytes, and computation. Understanding these constituent letters allows developers to cultivate a vocabulary helpful for building safe, effective, and useful IoT solutions.

Manuscript received March 10, 2017; revised June 7, 2017 and September 5, 2017; accepted September 13, 2017. Date of publication October 2, 2017; date of current version August 9, 2018. *(Corresponding author: Joshua E. Siegel.)*

The authors are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: j_siegel@mit.edu).

Digital Object Identifier 10.1109/JIOT.2017.2755620

Creating a protected and efficient IoT architecture addressing these ABC's is not a new goal. In the following sections, we explore this alphabet and provide background information on security and efficiency improvements for connected systems. Additionally, contemporary connectivity architectures are explored.

A. Privacy and Security

Ensuring the safety and security of data and connected systems fulfills a critical need in a connected platform's implementation. IoT connects many personal or high-value things, which brings great opportunity and significant risks to privacy and security. These areas pose significant challenges to the deployment of cloud and other connected systems, with the privacy of sensitive user data a particular concern [8], [9]. In designing IoT platforms and services, addressing system security and data privacy must come first—the A's in our IoT alphabet including actuators must be protected, while sensitive attributable data must be also maintained adequately. Without these assurances, a connected platform will have difficulty gaining traction and sustaining long-term growth due to perception issues and the risk of data leakage.

Frequently, these privacy challenges revolve around data ownership and sharing policies [2]. While some platforms default to opt-out sharing, others have proposed relying upon opt-in sharing [10] and data visualization tools [11] to ameliorate user fears of data abuse. Such policies and tools are critical to improving user acceptance of IoT platforms and will be integral to an improved architecture addressing today's common concerns.

Though leaks resulting from permissive sharing policies are simple to solve, security vulnerabilities present more challenging threats. These vulnerabilities are especially critical to address in light of the proliferation of interconnected devices in sensitive locations with access to potentially harmful actuation capabilities. There is a need for attack resilience, data authentication, and access control to ameliorate these problems and security approaches applied to conventional networks must be improved before being applied to IoT [9].

This problem of undersecured, overly sensitive connected devices is due in part to IoT's rapid growth. The rapid rollout of connected technologies led many systems to rely on "security through obscurity" due to short development cycles [12], [13]. Strict cost targets led developers to eschew authentication, encryption, and even message integrity checks [14], as the computational overhead for cryptography require processors with higher memory and speed requirements [12]. For these reasons, many products on the market have little to no built-in protection, and the hardware may not have sufficient computational overhead or update capabilities to support future improvements while meeting real-time performance requirements [15].

Consider three household IoT devices highlighting IoT's fragmented security: Philips Hue lightbulbs rely on a whitelist of approved controllers and transmit data in plaintext; Belkin WeMo outlets use plaintext SOAP communication without authentication; and NEST smoke alarms use encrypted traffic to communicate with a remote server, with changing OAuth2 tokens to ensure the integrity of the connection [14].

These devices demonstrate a range of system complexity and security. Improving the less-protected devices is not a simple matter; the device designs themselves must be changed. Intensive encryption may not be compatible with already-deployed WeMo hardware, for example, leading Belkin to stop developing for Apple's HomeKit standard [16].

Recently, groups have made an effort to standardize communication protocols and data exchange to improve security [14]. Without legislation, unifying manufacturers and developers will prove challenging. Further, standardization only addresses future devices—a solution compatible with past and present devices is preferable.

Considering the constraints of embedded devices, researchers have proposed intermediate, network-level solutions for "security as a service" allowing dynamic communication rules in intermediate layers [14], [17]. Others suggest creating crowd-sourced repositories for users to share their device information to aid in identifying attack signatures and creating abstracted device models for fault detection [18]. Multilayer Cloud security frameworks have also been suggested as a means of implementing firewalling, access control, identify management, and intrusion prevention [8].

These solutions improve upon business as usual, but have their own challenges in service management, rule creation, scalability, and incentivizing data sharing.

B. Resource Efficiency

Connected systems must optimize for a number of resource inputs. In our IoT alphabet, the B's and C's refer to resource efficiency in terms of battery, bandwidth, bytes, and computation. Battery refers to device or system power consumption; improved energy efficiency allows systems to run longer without service interruptions [9]. Bandwidth refers to data transmitted or routed; reduced data needs limit network congestion and reduce system operating costs. Bytes refers to the amount of data required to be stored; limiting the amount of data stored lowers costs and simplifies analysis and information sharing by avoiding the trap of Big Data. Computation describes the processing needed in constrained nodes; processing can take time and consume power, assuming a device's processor is even capable of executing particular code. In these ways, common design considerations from wireless sensor networks [5] apply in contemporary IoT implementations, as these use similarly constrained nodes for data collection and actuation [4].

In implementing a system, these problems are often coupled, for example, transmitting data frequently has a more substantial impact on the battery life than sampling a sensor [2].

Addressing these needs, researchers have demonstrated routing optimization, power minimization, and efficient computation for wireless sensor networks and other connected systems.

To optimize routing, self-organizing data dissemination algorithms using data-centric storage to minimize search energy and bandwidth expenditure [19].

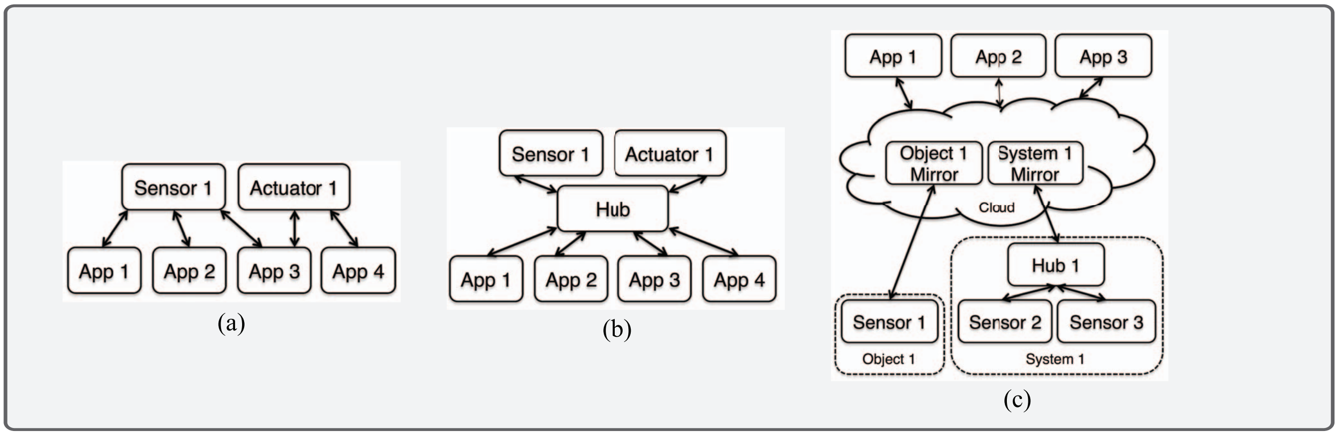

Fig. 1. Comparison of commonly used architectures for connected systems. (a) In direct connectivity, an application engages directly with a device or devices. (b) With hub or gateway connectivity, devices and applications connect to a hub. This hub moderates the flow of information. (c) Cloud connectivity relies on remote computation. Devices and applications connect to a network, and Cloud resources manage data storage and flows of information and commands.

Other approaches minimize the sampling rate on networks with a finite bandwidth. Adlakha *et al*. [20] designed a sensing system making use of a Kalman filter to account for missing observations and identify the rate at which an event of interest must be sampled to maintain a target estimate error covariance.

Hu *et al*. [21] used linear programming to predict intermediate sensor data, minimizing energy requirements by reducing the number of sensors and sampling rate required to instrument a system.

Jain *et al*. [22] treated sensor management as a filtering problem, using a device-run Kalman filter to meet error targets. This minimizes bandwidth at the expense of computation a constrained node.

Compressed sensing exploits *a priori* sparsity information to reconstruct signals from sparse samples. Data are compressed at an end node, transmitted, and reconstructed at a "fusion centre," trading bandwidth for computation [23].

Another approach to minimizing resource spend identifies the most critical sensors to minimize worst-case errors [24]. By deploying only critical sensors, the energy or bandwidth requirement for a network may be reduced.

These approaches tend to optimize single elements of a larger system, and often compromise one challenge for another (e.g., bandwidth for computation).

There have been some efforts to optimize for multiple components in aggregate, e.g., energy cost and security. Li *et al*. [25] developed a resource optimization architecture for minimizing the energy cost in data centers while assuring system security and meeting delay targets. This approach considers the use of frameworks for adaptively timing computation to capitalize on utilizing resources that are the lowest cost or alternately most available.

Similarly, an architecture may be created to optimize for resource use in aggregate while targeting improved security and privacy relative to business as usual, while meeting broader data quality and delay targets.

An approach not relying on costly computation and jointly addressing security, efficiency, and scalability is needed.

C. Foundational Architectures

Connected systems employ one of several connectivity architectures. Each has advantages and disadvantages ranging from complexity to resilience to scalability. We discuss three common approaches to connectivity: 1) direct; 2) hub; and 3) Cloud.

1) Direct Connectivity: In direct connectivity, an application queries and controls a system's sensors and actuators directly. An example pairs a mobile phone to Bluetooth environmental sensors and lightbulbs.

This topology, shown in Fig. 1(a), is efficient for a single application used in conjunction with few devices. Temperature information is sent only when it is needed and the lamps process all incoming commands. However, this architecture scales poorly. Each additional application adds data queries or sends new command requests. If an application samples a device at n Hz, and m copies of that application are running, the devices are queried $m \times n$ times per second, consuming additional bandwidth and power despite these samples conveying similar information. In the worst case, the network becomes saturated.

The use of low-cost constrained nodes causes insecurity due to their inability to run credentialing services and timely encryption. Should a malicious agent join the network, these nodes are incapable of limiting access.

Though quick to develop and test, this approach is unsuitable for scalable deployment [26] or use in safety-critical systems.

2) Hub Connectivity: With hub connectivity, shown in Fig. 1(b), data requests and control commands pass through a master node capable of translating and moderating the flow of information. An example of this is a ZigBee-enabled home lighting system, which uses a hub to bridge several ZigBee lights to WiFi.

Gateway systems may have limited sampling intelligence to perform local data aggregation and preprocessing, [27] reducing redundant data collection and transmission. A simple example of scaling considers an application requesting at n Hz and one requesting at m Hz, with the gateway polling at the ceiling of these two request rates.

Hubs may run basic Firewalls, encrypt communications, and validate credentials, simplifying the blocking of malicious agents.

While hub-based systems improve scalability over direct architectures, there are still limitations. Resource constraints mean hub architectures work best for small to medium networks with known application payloads.

3) Cloud Connectivity: The cloud approach shown in Fig. 1(c) effectively extends the hub model with infinitely scalable resources between end nodes and applications. A cloud system mirrors one or several devices or systems, storing information centrally for multiple use. These mirrors may combine data from different sources, applying additional processing to filter data, and aggregate results [10].

As with a hub, data, and control requests are abstracted from physical devices.

Cloud connectivity is known for its scalability and extensibility [28], multiple-use, and ability to abstract devices from applications.

III. Human Inspiration

Reviewing prior art shows several unaddressed needs. For example, efficiency must be optimized at the most constrained nodes, while security must allow real-time data access and control. Today's solutions are also application-specific, whereas an architecture should support dynamic scalability and extensibility.

In evaluating these needs, we present the view that humans themselves present a good analogy for secure and efficient connected architectures. We use context and cognition to gather, share, and act upon data. We synthesize data from multiple sources to provide enhanced information and we minimize effort in acquiring and fusing data with estimation. We even protect ourselves and our resources through abstraction.

Consider a scenario consisting of two people talking as they wait at a train station. The person making requests for information is the client "application," and the individual collecting, synthesizing, and moderating the flow of information the "proxy." Our proxy has access to a wristwatch and a train schedule.

When an application asks the current time, the proxy considers a number of factors prior to collecting information and formulating a reply. The proxy notes who is asking, the history of previous interactions, and the application's apparent need for timely and accurate data. A typical request "what time is it?" is met with a reply addressing average needs for timeliness and precision, "it is about 10:30."

In the following sections, we illustrate how humans apply cognition to formulate context-appropriate replies.

A. Varied Request Priorities

Applications have varied request priorities. One application may have little interest in information, so timeliness and precision are noncritical. Estimates are acceptable and replies may wait until the proxy has free time to process the request, as is the case with a child nagging a parent.

Another application may be high priority and require a precise and timely reply. The proxy must expend additional effort to immediately and directly acquire precise data. An example application is a train conductor who wants to avoid delaying passengers. The proxy knows the conductor has a critical need to know the time, and therefore chooses to get a direct measurement from his or her watch. The additional accuracy is conveyed directly, e.g., by saying "it is 10:30 exactly... now."

B. Data Synthesis

Beyond acting as a valve for the flow of information, proxies may synthesize data from multiple sources. In our train station example, an application may make a request for processed information such as "how long until the train to Alewife arrives?" The proxy may reply using information from multiple sources to formulate the appropriate response: "the train schedule says the train arrives at 10:47 and it is 10:30. You have 17 min."

C. Multiple-Use of Replies

Multiple applications may need the same information, and proxies allow reply sharing. In our example, a nearby passenger, another potential application, overhears the proxy's reply to the first application and no longer needs to make a dedicated request. This saves resources and allows low-priority applications to benefit from high-priority applications' replies.

D. Malicious Request Blocking

Requests can become annoying. In the case of a nagging child asking the time, the proxy may initially give coarse estimates to save the effort of directly acquiring a measurement. Eventually, the proxy may stop responding entirely. This limits data access for malicious and annoying applications.

E. Resource Safeguarding

Proxies have access to valuable information. If an unsavory application asks to access a data source (in this case, a watch), the proxy applies judgment to moderate access to resources (hiding the watch) and related data.

F. Command Simulation

Proxies simulate the future. In our example, consider an application requesting that a proxy look after his bag for the remainder of the day. The proxy considers first the source of the request, then mentally simulates the result of executing the command. If the command seems strange (a day is a long time to watch a bag), it may be verified and the application given a chance for correction. If the command is validated but would conflict with another objective (watching the bag means missing the train), the request may be denied.

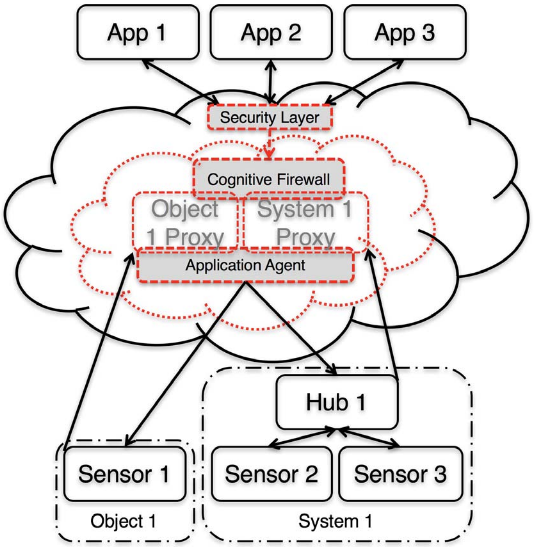

Fig. 2. This architecture builds upon the cloud connectivity approach, adding elements to improve efficiency, privacy, and security.

G. System Supervision

The proxy may supervise his own system measurement instruments and the behavior of their environment. Consider the case of a proxy checking his watch an hour apart, and seeing the time has not changed. The proxy knows the measurement has failed (a dead watch battery) or the environment is not behaving as expected (traveling at the speed of light). In either case, the source of the fault may be learned from, and if possible, remedied.

IV. System Embodiment

One sees that the human model can improve the security and efficiency of data collection and distribution. We seek to build this contextual cognition using scalable and extensible Cloud infrastructure.

Considering how humans handle data, we propose the creation of data proxies based on statistical and physical system models, allowing connected systems to meet prescribed applications' QoD requirements. Data proxies then digitally duplicate physical systems from intelligently sampled inputs. The proxy's model and sampling rate selection are run by an "application agent" in the Cloud, allowing constrained nodes to conserve power and bandwidth resources.

Additionally, we present the concept of a cognitive layer using the proxy's model to respond to system behavior changes and to evaluate the impact of commands. As mirroring abstracts digital from the physical systems, the use of an intermediate "security layer" combines with the cognitive layer to moderate connections, protecting the proxy's reference system from direct access.

The five new elements shown in Fig. 2 extending the cloud architecture are as follows.

1) The QoD accompanies application data requests, specifying accuracy and timeliness targets. The QoD has an associated element in replies, providing confidence intervals.
2) The *security layer* moderates incoming data requests and actuation attempts, allowing only approved connections through to the private cloud.
3) The *cognitive layer* observes and anticipates system behavior, applying context information to determine when system behavior is anomalous or when a command may lead to a fault.
4) The *data proxy* applies process and measurement knowledge to estimate the system state from limited input data.
5) The *application agent* uses prior simulation and learned models to optimize input sampling rates based on acquisition costs and aggregate QoD needs.

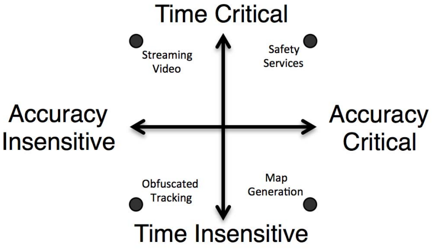

Fig. 3. QoD is distilled into two axes: timeliness and accuracy.

These elements mirror human cognition. This "new IoT" optimizes sampling to meet QoD objectives while minimizing resource use and related cost subject to constraints. These constraints include physical constraints, like minimum and maximum sensor sampling rates, as well as application constraints, like minimum and maximum freshness and/or error.

In an example scenario with m applications requesting samples at n Hz, this architecture's intelligence would eliminate unnecessary requests and sampling at n Hz. Using the proxy's system model in conjunction with the application agent, intelligent sampling will save bandwidth and power while providing security through abstraction.

A. Quality of Data

QoD is a quantitative metric used to provide an objective for the scheduler and is included in data requests as well as replies. A system's QoD metrics may vary-based upon the type of information uploaded to the Cloud and typical application types, so we focus this section on an example embodiment.

To formulate this sample QoD, we examined elements of quality of service, experience, and information (QoS, QoE, QoI) and focused on those parameters both improving application functionality and useful in informing scheduling.

Traditionally, QoS may consider service time, delay, accuracy, load, priority, reliability, efficiency, and sensing precision [29], [30]. QoE evaluates user experience based on perception [9], while QoI looks at the value of the data itself. From these metrics, we derive a canonical QoD that distills into timeliness and accuracy requirements, with applications across a spectrum as shown in Fig. 3.

Applications target a particular QoD, and the proxy replies with data meeting or exceeding those targets as well as confidence intervals. The inclusion of confidence data along with the requested parameters aids applications in working with imperfect estimates and measurements and may be considered the computational equivalent of the human model's use of the words "approximately," "precisely," and similar. In the case where it is not possible to reach QoD targets, the reply will return the closest possible data and denote that the returned information does not meet the requested objective. An example where this might occur is when requesting information from a sensor with a limited resolution or insufficient maximum sampling rate.

Common QoD objectives and example rationales appear below.

1) *Instantaneous Accuracy:* The allowable error between the model and real system at a particular time. Maximum error is useful in the case where a factory manager must know the power used by a machine at a particular moment that another might come online, while minimum error is useful for tracking websites where locations must be kept imprecise by design.
2) *Periodic Accuracy:* The allowable error between the model and real system at regularly spaced intervals. This is useful when generating "snapshot" reports to determine energy trends throughout the day.
3) *Average Accuracy:* The allowable sum of errors between the model and real system between two target times, which is useful in supervising equipment over the long term.
4) *Maximum Latency/Freshness:* The acceptable temporal recency of direct sensor measurement. Maximum latency aids in maintenance operations, e.g., for determining when a machine is cool enough to touch. Minimum latency is useful where obfuscated data are preferable, as is the case with public-facing websites where "fresh" data presents a security risk.
5) *Threshold Detection:* The acceptable delay between a state being reached and notification being sent to the server. Useful for temperature monitoring.

These metrics are not exhaustive and vary by an application's needs. For example, these may be extended by the QoD parameters from Wu *et al.* [31], wherein data has measurements contain information about accuracy, truthfulness, completeness, and up-to-dateness, or might be an a composite metric consisting of several of these objectives.

QoD is coupled closely to resource costs. In calculating costs for IoT systems, we consider expenses and resource use. Example costs include computation and the three "b's": battery, bandwidth, and bytes (power consumption, data transmission, and storage), with each element having fixed and variable costs and possible geotemporal modifiers (location or time dependent resource use). Some costs may be coupled, like computation and power, or computation and bandwidth, while others are independent.

In our architecture, we typically optimize costs while attempting to meet a QoD target, or optimize QoD while staying within a resource budget. We consider IoT as a simple system with applications connected to a middleware querying data from sensors. We assume that sensors collect data directly measuring a process or event, and that optimizing the sampling rate, accuracy, and latency of a single sensor has a direct correspondence to optimizing the monitoring of a process or event. Joint optimization of these sensors' sampling rates in the context of maximizing the QoD allows for significant resource cost reductions while maintaining state estimate accuracy.

B. Security Layer

Contemporary approaches to security frequently offer solutions to single problems, whereas layered approaches offer enhanced threat resilience [8]. Therefore, we choose to implement privacy- and security-protecting elements in our architecture at multiple points, using a combination of credentialing, firewalling, and supervisory models. The first element in our tiered approach is the security layer.

The security layer validates incoming connections' credentials and protects an encapsulated Private Cloud nested inside a Public Cloud from invalid data and actuation requests. This layer moderates access to the data proxy and cognitive layer described in the following sections, preventing an unauthorized application from directly accessing these elements. This layer is the IoT equivalent of a security guard at a private event—just as a guard checks IDs and moderates access to an event, the security layer uses an encrypted connection and Cloud computational resources to check credentials before allowing appropriate incoming connections and blocking malicious connections. Similar to how a guard may call a supervisor to check an out-of-state ID, the security layer seeks human-in-the-loop validation before allowing or blocking a connection of unknown provenance.

The security layer uses Cloud resources to allow for rapid authentication, enhancing timeliness. It may rely on predefined rules or machine learned rules, and may support conventional security approaches such as certificate authentication, credential revocation, or command and request validation.

C. Cognitive Layer

Providing a second layer of threat resistance beneath the security layer, the cognitive layer provides firewalling and supervisory elements to assess system performance and evaluate the impact of commands for undesirable effects.

This cognitive layer makes use of data proxy's system models to observe the system's evolution and to test incoming commands. This layer embodies Isaac Asimov's third law, for the system to protect itself, and consists of both a cognitive Firewall, for simulating the impact of a command to ensure it is safe prior to execution, and the cognitive supervisor, to monitor the system's evolution over time in the absence of commands. These are visualized in Fig. 4(a). Both cognitive elements are checked against known, machine-learned, and human-in-the-loop limits to raise alarms or prevent actuation when anomalous behavior is detected.

The cognitive Firewall acts on incoming commands, evaluating the impact of a command prior to execution by forward-simulating the system's future state and raising alarms when the potential evolution endangers the system. Two possible system evolutions are shown in Fig. 4(b).

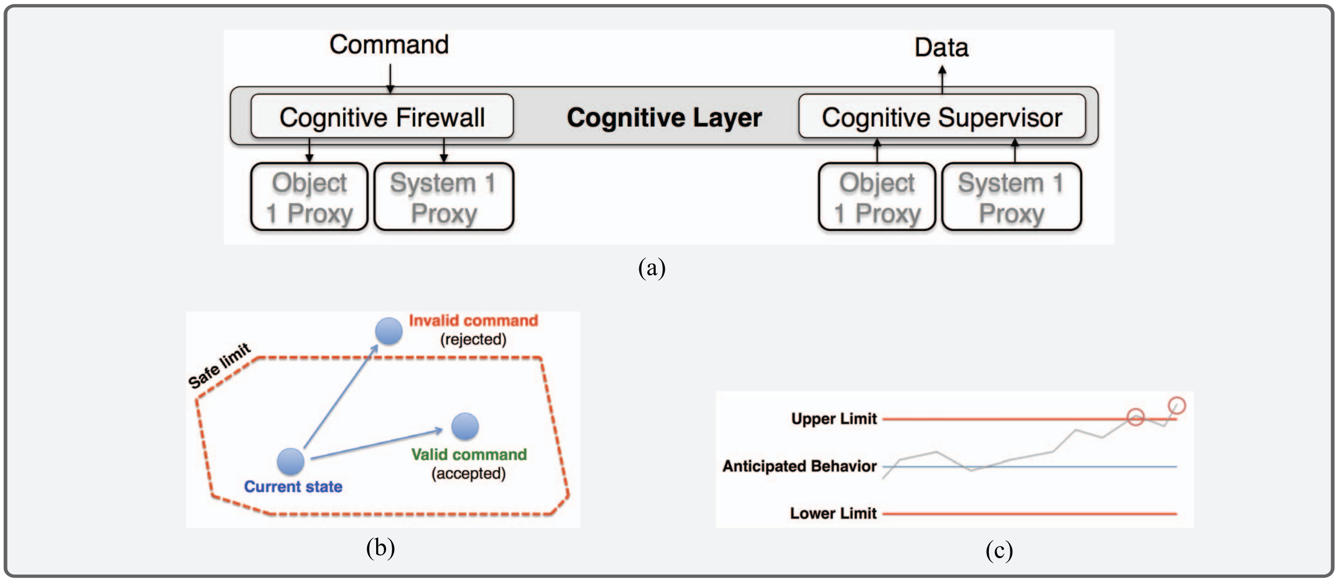

Fig. 4. Cognitive layer ensures a system is behaving as expected and that commands do not bring about harm. The data proxy's model is used to apply context information for monitoring faults and validating incoming commands prior to execution. (a) Cognitive layer consists of a Firewall and a supervisor. The supervisor monitors the system; the Firewall checks commands for validity and safety prior to execution. (b) This simplified example shows how the cognitive Firewall uses a system model to evaluate the impact of a request against control limits, rejecting invalid commands. (c) Cognitive supervisor monitors the system against known and learned limits, and raises an alarm when the system approaches an undesirable state.

This Firewall uses the system model to filter out inputs with potential for negative consequence based on rules, machine learning, or human-in-the-loop validation. For example, a cognitive Firewall may be used to simulate the impact of a command received over the Internet by a robotic arm's data proxy to check for collisions and other undesirable effects prior to acting upon the command.

This is similar to how people think ahead before taking actions; if a passenger in a taxi asks the driver to accelerate through a red light, the driver resists because of the risk of an accident or ticket. Similarly, the Firewall applies context and cognition to detect and avoid probably undesirable scenarios.

While the Firewall tests commands in the Cloud before relaying these to a physical system, the cognitive supervisor examines the system in the absence of commands and monitors the system's true versus anticipated evolution, raising alarms when the behavior is not as expected or exceeds control limits. This may be used to identify potential faults in the physical system, the system model, or the measurement equipment at an early stage, and is similar to statistical process control. An example "process out of control" is shown in Fig. 4(c).

The supervisor may be used to identify faults in a machine or process. For example, a mill which expects to require a particular amount of power during a machining operation might identify that the power to cut a material is out-of-line with learned or modeled expectations. Such a fault could indicate a failure in the proxy's model, a failure in the system's sensors, or a fault in the system itself, such as using material that does not meet specifications. With a cognitive supervisor, faults may be detected and responded to early, e.g., by contacting the material supplier to replace the machine's feedstock before any widgets leave the factory or fail in the field.

This is similar to how people learn to understand cause and effect. If a person's house is 20 °C and they set the thermostat to 22 °C, they expect the house to be 22 °C within two hours. If the temperature is below 22 °C, the person knows that either the heater is broken, the sensor is wrong, their model is wrong (the temperature outside is far colder than expected), or perhaps a window was left open.

D. Data Proxies

Data Proxies are the "brain" of our approach to cognitive IoT. Proxies are Cloud-run digital duplicates, using observers, estimators or probabilistic modeling tools to digitally mirror a physical system or process. These models take sparse data inputs (as determined by the application agent in response to targeting a particular QoD) and create a rich digital representation of an object or system in the Cloud. These replicas provide an estimate of system states and their evolution based on intermittently measured inputs and outputs synthesized by a context-aware system model.

The canonical proxy uses the digital duplicate to take an imperfect, periodically sampled view of a system's state, filling in gaps based on trusted models for anticipated system behavior. Applications communicate with the data proxy and it is related elements rather than the physical device or system it mirrors—the use of an intermediate proxy allows the isolation of devices from requestors, while data in-fill (interpolation, estimation, or another means of gap filling) reduces the requisite sampling rate and resource requirements to maintain high-quality data.

In IoT, applications often involve monitoring a process (machining in a factory) or an event (overheat detection). Control theory's observers and estimators, Kalman filters, probabilistic graph models, and neutral networks may be applied to model key system characteristics. Often, such modeling require historical data and machine learning techniques to train and calibrate an appropriate system model.

While the proxy is central within this architecture, the proxy's implementation may change based on its integration with data sources and other resources, as well as based on the proxy's coupling with applications. For example, if an IoT system supports a particular always-on application the proxy may be designed with biases to ensure that this application is always addressed as efficiently as possible. Alternatively, the proxy may be designed for generalized application payloads. Altering the proxy's construction allows for it to remain central within the architecture while still closely moderating its performance for different input resources, end users, or applications.

E. Application Agent

The application agent is a Cloud-run query manager responsible for aggregating QoD requirements and optimizing resource use for specific data proxy models. The agent forward-simulates a sample data set to minimize the cost of sampling while ensuring timeliness and accuracy targets are met.

All application requests share a QoD with the application agent. The application agent aggregates each QoD and simulates the system using downsampled historic data, comparing the downsampled QoD to a fully sampled baseline. This allows the agent to determine optimal sampling schemes meeting the combined QoD. The agent then sets the sensors' sampling rates to value identified by simulation to minimize resource expenditure. As applications join and leave the system, the sampling rate is dynamically recalculated to ensure scalability and efficiency. This process is automatic, and may occur based on a schedule (e.g., at regular intervals), using machine learning to identify and anticipate changing QoD requirements, event-based reconfiguration, or other automated, and semi-automated techniques.

V. Example Application of Data Proxy

Usage-based insurance (UBI) is considered by many to be an equitable solution to insuring infrequent drivers [32]. In this model, drivers pay for insurance based on a number of factors including distance traveled. A similar application is vehicle miles traveled (VMT) taxation, which charges drivers for their use of public roadways [33]. For these applications, it is necessary to precisely know the distance traveled by a vehicle.

A mobile phone is an excellent candidate for distance measurement. Many drivers already own a mobile phone, and these devices possess sensors like GPS and accelerometers, as well as the ability to wirelessly interface with vehicle on-board diagnostics (OBD). These sensors provide the information necessary to enable UBI and VMT, but this use case presents a challenge: sensors consume power, and mobile devices are battery-constrained. Additionally, distance estimates are often derived from location, and users may not wish to share this sensitive information.

Data proxies allow us to apply a vehicle dynamics model to efficiently estimate trajectory and distance traveled, while the use of data abstraction and sensor fusion allows drivers to maintain the privacy of their location data.

We formulate the problem by considering costs and objectives: we aim to minimize the power cost of acquiring GPS, OBD, and accelerometer data while meeting a target accuracy provided by insurance companies or the government.

A. Proxy Model

The first step in developing a proxy is to select a system model. The accuracy and inputs to this model determine the performance of the data proxy.

For illustration purposes, we apply the vehicle motion model from Kumar *et al.* (2013) [32]. The car is modeled as a unicycle with the constraint that it moves along a trajectory with no slip. A Kalman filter is applied to estimate the location and the distance traveled by the vehicle (called system states here). These estimates are performed at a baseline frequency while the model allows GPS and OBD speed measurements to be incorporated at varying frequencies in the estimation model. Additionally, a scaling and bias correction are applied to accelerometer measurements for improved estimation accuracy.

To collect data, a vehicle was driven in a loop while recording information at the reference 10 Hz from a GPS device, an accelerometer, and an OBD interface.

We identified an optimal reference trajectory using fully sampled data run through our tuned Kalman filter. We then iterated filtering using differing downsampling rates for OBD and GPS and calculated the costs and QoDs for each run, as described in the following sections.

B. Costs

In our case, we consider only the per-sample energy cost of acquiring sensor data from GPS and OBD. GPS is the most costly sensor, while OBD is less expensive but still significant (we assume use of a WiFi OBD interface). Accelerometer acquisition is near-negligible, so we always sample this sensor at the maximum rate. This simplifies the model, as the accelerometer will be an input rather than a variable.

Our cost function with a per-sample cost becomes the linear combination

$$c_{\text{total}} = \lambda_{\text{GPS}} * n_{\text{GPS}} + \lambda_{\text{OBD}} * n_{\text{OBD}} \tag{1}$$

where n_{GPS} and n_{OBD} are the numbers of GPS and OBD samples, while their respective λ's are per-sample costs. In the base case we assume, $\lambda_{\text{GPS}} = 10\ \mu\text{W/sample}$ and $\lambda_{\text{OBD}} = 3.3\ \mu\text{W/sample}$.

C. Objectives and Constraints

Our objective is to minimize power use while accurately measuring the trajectory and the distance a vehicle travels. Further, we wish to maintain tamper-resistance (security) and driver privacy through the use of aggregate statistics.

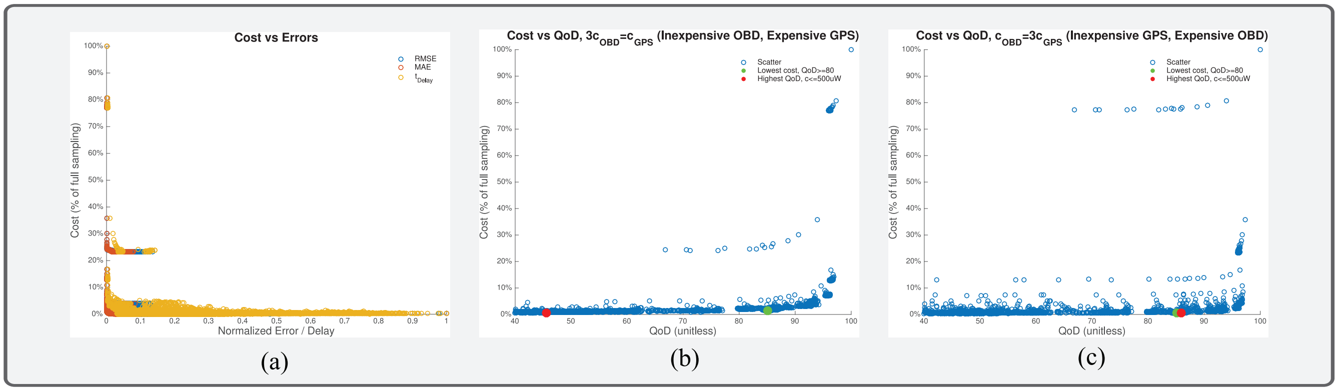

Fig. 5. Cost as a percentage of full sampling plotted against error and QoD for the UBI example system. (a) Cost versus normalized RMSE, MAE, and average threshold cross detection delay. (b) Cost versus aggregate QoD; $\lambda_{\text{GPS}} = 10\ \mu$W/sample and $\lambda_{\text{OBD}} = 3.3\ \mu$W/sample. Lowest cost for error target in green; lowest error for target cost in red. (c) Cost versus aggregate QoD; $\lambda_{\text{GPS}} = 3.3\ \mu$W/sample and $\lambda_{\text{OBD}} = 10\ \mu$W/sample; lowest error for target cost in red.

We define our quantitative QoD metric as a measure of error in process monitoring and event detection. In this example, we combine two measures of position estimation error with a distance threshold crossing delay metric

$$\begin{aligned}\text{QoD} = m_1(100 - \text{RMSE}_{\text{position}})\\ + m_2(100 - \text{MAE}_{\text{position}})\\ + m_3(100 - t_{\text{Delay}}).\end{aligned} \tag{2}$$

$\text{RMSE}_{\text{position}}$ and $\text{MAE}_{\text{position}}$ are the root mean squared error and the mean average error of the trajectory estimates with respect to the reference trajectory calculated using all sensor data at 10 Hz. t_{Delay} is the average, absolute value of the delay in detecting a series of threshold crossing events (in this case, when the distance traveled crosses $d = n * 500$ m and $n \in \{1, 2, 3, 4, 5\}$). Sampling configurations resulting in undefined threshold cross delays are discarded.

m_1, m_2, and m_3 are tuning parameters chosen to normalize each error type by its maximum value from a sample set while providing equal weighting.

This QoD behaves intuitively; tuning constraints $m_1 + m_2 + m_3 = 1$ make the zero-error case result in QoD $= 100$ with lower numbers representing increased error relative to the reference trajectory. Note that a QoD may be negative when errors are significant.

We are constrained by the maximum 10 Hz sampling rates of the sensors. Minimum sampling period of GPS is constrained to 100 s, and OBD is constrained to 50 s.

D. QoD Optimization Results

We demonstrated the data proxy's utility in maximizing data richness and reducing resource use in the context of our IoT architecture by optimizing the QoD for a target cost and optimizing the cost for a target QoD. Further, we showed that the optimal sampling arrangement varies with sensor costs and QoD type.

The QoD varies based on weighting factors and constituent inputs. The normalized error elements RMSE, MAE, and t_{delay} are plotted against cost in Fig. 5(a).

These errors follow a $1/x$ profile, with the minimal clustering indicating low error variability for a given cost. Striation would suggest stronger dependence on certain samples. Changing the weight of each error in the QoD will change the shape and therefore the optimal sampling arrangement.

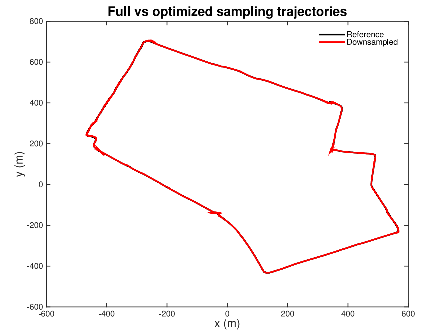

Fig. 6. Reference trajectory (black) and downsampled estimate (red) show minimal positional error. This sampling scheme offers QoD = 95.5 with a 92.9% reduction in cost relative to the QoD = 100 reference case.

Next, we studied the impact of cost on QoD. A base case with high cost GPS and inexpensive OBD is plotted in Fig. 5(b).

Visually, one sees that some sampling schemes are not feasible due to high cost or poor QoD. Noteworthy is the steep increase in costs for QoD >= 95, illustrating that slight compromises in QoD can lead to significant cost savings.

The knee in this curve indicates that our reference trajectory is significantly oversampled—once the estimate is reasonably accurate, additional data offers diminishing returns. Therefore, the horizontal portion of the plot offers improved return on sampling. The use of simulation-based optimization ensures that the optimal cost/benefit relationship is identified, even without a-priori knowledge.

In UBI and VMT, using simulated data to determine the optimal return on sampling could dramatically increase battery life with minimal loss in state estimate accuracy. For example, in the high-cost GPS case, accepting a QoD of 95.5 instead of 100 allows the resource expenditure to be reduced by 92.9% relative to full sampling. A comparison of the reference trajectory and this less-expensive scheme appears in Fig. 6.

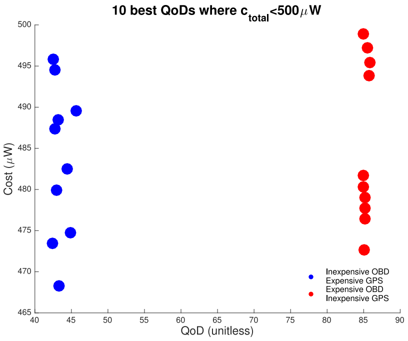

Fig. 7. Ten best QoDs meeting $c_{\text{total}} < 500\ \mu\text{W}$ are shown for the case where $\lambda_{\text{OBD}} = 3\lambda_{\text{GPS}}$ and $3\lambda_{\text{OBD}} = \lambda_{\text{GPS}}$. Note that when GPS is relatively less expensive, it is sampled more frequently dramatically increasing the QoD.

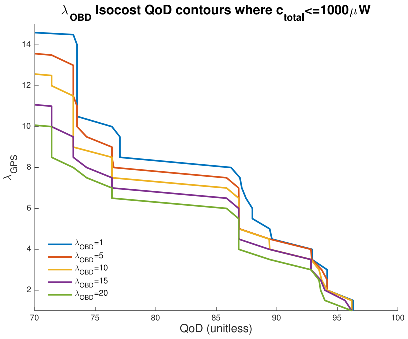

Fig. 8. OBD isocost contour lines plotted against QoD and GPS sampling cost demonstrate how varied costs and cost ratios can impact a proxy's QoD.

We then simulated a different model with low cost GPS and high cost OBD. Cost is plotted against aggregate QoD for this solution in Fig. 5(c).

Note that as the costs of each sensors change, so does the clustering of the cost/QoD plots. These differences can be significant; for example, the highest possible QoD for $c_{\text{total}} < 500\ \mu\text{W}$ is QoD ≈ 45 in the low-cost OBD case, whereas in the high-cost OBD scenario the QoD ≈ 86. In the latter case, the optimal sampling shifts to include more GPS results as acquiring these data are relatively less expensive.

To further illustrate this point, consider Fig. 7, which shows the ten best QoD's meeting the constraint $c_{\text{total}} <= 500\ \mu\text{W}$ for both cost models.

From this, we see that increasing the cost of OBD relative to GPS shifts the possibility space to include additional GPS samples, raising QoD. We also see that for a cost target, a range of QoDs are possible and that the highest-cost solution is not necessarily the most accurate because we are querying a process estimate rather than an individual sensor.

The best-possible QoD for a given cost depends heavily on the sampling rate of each sensor. To highlight this, we create a contour plot indicating the QoD for differing values of λ_{GPS} and λ_{OBD} with a maximum allowable cost $c_{\text{total}} = 1000\ \mu\text{W}$. Fig. 8 shows constant-OBD-cost curves relating QoD and GPS sampling cost. As expected, decreasing GPS cost increases QoD by making direct GPS sampling more feasible. We see a similar trend when examining OBD costs, where low-cost sampling leads to increased QoD. The decision to sample a particular sensor occur at inflection points, leading the contours to appear like step functions. Note that the curves for low-cost sampling totally envelop the high-cost sampling curves, which shows that the optimization works as intended.

This section demonstrates that the QoD and cost optimization models work as anticipated, and proves the value in using forward-simulation to choose the sampling schema maximizing QoD for a fixed target cost, or vise-versa.

Other data proxies generate similar cost versus error plots [7]. Transforming these error to a QoD of the form presented here therefore results in a similar shape function for QoD, indicating that an optimality should exist for most model-based abstractions underlying a physical process.

E. Multiapplication Optimization

The previous example considered how a quantitative QoD metric and an estimator-backed data proxy minimize the need for direct sensor sampling while meeting the performance requirements of a single application. Often, several applications with different performance requirements may require access to the same piece of information. One application may require low-latency reporting; another may perform best with tight state estimate error bounds. A third may work well with coarse, occasional data.

A key benefit of cloud connectivity is that a device or service can upload data once and multiple applications may benefit. The data proxy architecture scales similarly, and though the three connected applications above have different QoD targets, information still need only be uploaded once.

When multiple applications with different QoDs request direct or estimated data, the application agent switches to an active scheduling role. Rather than optimizing for the QoD from Section V-C, the agent determines the optimal aggregate set of constraints designed to meet the most stringent applications' QoD requirement. Through the use of historic data and simulation, the application agent evaluates the impact of different downsampling approaches and determines the most efficient querying schema capable of meeting the applications' performance requirements. The agent is able to use scalable Cloud resources to dynamically simulate and adjust sensor sampling rates as applications join or leave the platform, ensuring optimal resource use no matter the application payload.

This dynamic optimization allows a range of applications across industries to benefit. For example, such a system could leverage connectivity to democratize healthcare between constrained hospitals and hospitals where resources are freer [34], by utilizing cognitive models at network extremes to allow for low-latency, high-reliability control. Similarly, the reduction in resource requirements would allow sensing and connectivity to migrate into very low-power wearable devices, reducing the cost and complexity of contemporary health-aware clothing [35].

F. System Supervision

Connected systems possess several possible failure modes. In passive (data-only) systems, sensors can fail, connections can drop, or a proxy's model might be proven inaccurate or incomplete. In active systems, the possibility exists for malicious incoming commands.

In the data proxy architecture, the cognitive layer protects the system against threats that manage to breach the security layer. The cognitive layer applies knowledge of the data proxy's model to identify and respond to a fault condition or to send notification to a secondary system or reviewer.

In our UBI example, sensors such as GPS could fail, the proxy model could break down when used outside the designed parameters (e.g., to monitor a helicopter's travel instead of a wheeled vehicle), or a user could generate false information to "spoof" the distance traveled. In each case, a cognitive supervisor applies system and process knowledge to determine that the real system's behavior is divergent from the simulated system or that the relationship between inputs and states (e.g., the accelerometer's reading and the next position) are incongruous. Further, rule-based or learned limits may be implemented to identify specific faults, for example, if a vehicle accelerates unreasonably fast, that sensor's readings might be flagged as spurious and discarded. A similar approach could utilize a cognitive model tailored for a different set of applications, e.g., a supervisor optimized for healthcare supervision, similar to that suggested by Chen *et al.* [34].

In an application with sensitive actuators or with the potential for causing damage or injury, such as is the case with remote vehicle control, this same model-based or rule-based failure identification could be used to limit the impact of malicious commands. For example, an autonomous vehicle may allow remote cellular throttle modulation. In this vehicle, all incoming commands would be simulated using the proxy's model and tested against known and learned limits as part of a cognitive Firewall. If a malicious command requests 100% throttle for 10 s, this Firewall would use the proxy's model to simulate the impact of the request and block it from reaching the vehicle upon identifying a nonzero likelihood of injury, damage, or discomfort.

VI. Conclusion

We identified opportunities to improve the IoT, proposing the creation of a new architecture with QoD targets, security and cognitive layers, mathematical-model-based data proxies, and an application agent to optimizing sampling costs or minimizing error subject to constraints.

Building upon the human model of applying context and cognition to data management, our architecture abstracts physical from digital systems to improve security and efficiency. It applies context information to supervise systems and to protect them against malicious commands, fuses data to provide difficult to obtain measurements, and uses estimation to minimize sampling cost. Together with clear ownership policies and data sharing visualizations [10], [11], this architecture's use of abstraction and creation of "black boxed" aggregate data addresses privacy concerns.

Using the practical application of UBI, we demonstrated that proxy models which are well calibrated to an underlying physical process may allow us to reduce the energy necessary to represent that process in the cloud. We demonstrated that querying information does not require one-to-one sampling of the sensors instrumenting that process, and showed that it is possible to substantially minimize costs without significantly increasing measurement error. This level of abstraction and sensor fusion improves security by eliminating applications' direct access to physical systems and preventing the long-term storage of sensitive data. Further, this same technique may be used to minimize data transmitted, conserving costly bandwidth. This approach to cloud mirroring ultimately reduces technical, economic, and consumer sentiment barriers to the deployment of connective technologies. Ultimately, with the reduced bandwidth costs, computational requirements, and improved security facilitated by a context-aware, cognitive architecture for the IoT, networking will become tenable on more devices in more places, helping to achieve the idealized vision of a fully connected world.

Some challenges remain to be addressed. Model selection, for example, will remain an active domain of research, with a focus on characterizing and controlling for noise and model evolution. Other challenges relate more to system implementation—actuation latency and data accuracy may suffer due to the reduced sampling rate of data proxies, so research is needed to quantify the impact of these delays and accuracy losses. Relatedly, current data representations must be extended so that applications may account for the varied accuracy of information received in response to a request. A probabilistic extension to the data proxy may facilitate this accuracy reporting and ensure that returned data are sufficient to ensure a high degree of application performance.

By developing an architecture allowing more devices in more places to join the IoT, we will ultimately support the next generation of products and services improving industry, transportation, healthcare, and quality of life. The data proxy's efficiency improvements will allow even the smallest, most resource-constrained device to join the ranks of "Big Data" systems, while this architecture's security improvements will enable new modalities for actuation never before possible.

A. Future Work

Future work will examine how best to define QoDs for various application types, how best to build and adapt data proxy models for a system in realtime, and how to quantify a proxy's performance statistically. Additional work will focus on implementing a functional cognitive Firewall to protect smart homes and connected cars, while the cognitive supervisor will be used to enable "cognitive prognostics" capable of identifying system faults early, reporting these automatically and providing rich information to aid in their repair. The use of this low-cost architecture will lead to the deployment of connected devices in more places, creating richer data mirrors and supporting enhanced pervasive sensing prognostic opportunities by reducing the amount of data needed to identify a fault. This architecture will also be adapted to work at the local network level, for example to apply an in-car cognitive Firewall and to reduce loading on constrained networks such the vehicle's controller area network linking a vehicle's electronic control units.

We further aim to extend this paper from mirroring physical processes using sparse input data to include algorithmic processes dedicated to software monitoring, fault detection, and automated error correction in high-criticality systems that are not instrumented today. These systems include smart factories, infrastructure, and collaborative vehicle navigation systems. The cognitive elements of this architecture have the potential to transform how and what we connect to the Internet, affording greater opportunities and lower risks than conventional systems. This highly efficient and secure connectivity has the potential to transform all products with connected data in the design, manufacturing, and use phases.

REFERENCES

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[3] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[4] C. C. Aggarwal, N. Ashish, and A. Sheth, *The Internet of Things: A Survey From the Data-Centric Perspective*. Boston, MA, USA: Springer, 2013, pp. 383–428, doi: 10.1007/978-1-4614-6309-2_12.

[5] K. Bhaduri and M. Stolpe, *Distributed Data Mining in Sensor Networks*. Boston, MA, USA: Springer, 2013, pp. 211–236, doi: 10.1007/978-1-4614-6309-2_8.

[6] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of industrie 4.0: An outlook," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, p. 7, 2016, doi: 10.1155/2016/3159805.

[7] J. E. Siegel, "Data proxies, the cognitive layer, and application locality: Enablers of cloud-connected vehicles and next-generation Internet of Things," Ph.D. dissertation, Dept. Mech. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, Jun. 2016. [Online]. Available: http://hdl.handle.net/1721.1/104456

[8] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," *IEEE Trans. Services Comput.*, vol. 9, no. 1, pp. 138–151, Jan./Feb. 2016.

[9] L. Li, M. Rong, and G. Zhang, "An Internet of Things QoE evaluation method based on multiple linear regression analysis," in *Proc. 10th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Cambridge, U.K., Jul. 2015, pp. 925–928.

[10] E. Wilhelm *et al.*, "Cloudthink: A scalable secure platform for mirroring transportation systems in the cloud," *Transport*, vol. 30, no. 3, pp. 320–329, 2015.

[11] S. Mayer and J. Siegel, "Conversations with connected vehicles," in *Proc. 5th Int. Conf. Internet Things (IoT)*, Seoul, South Korea, Oct. 2015, pp. 38–44.

[12] O. Arias, J. Wurm, K. Hoang, and Y. Jin, "Privacy and security in Internet of Things and wearable devices," *IEEE Trans. Multi Scale Comput. Syst.*, vol. 1, no. 2, pp. 99–109, Apr./Jun. 2015.

[13] M.-H. Maras, "Internet of Things: Security and privacy implications," *Int. Data Privacy Law*, vol. 5, no. 2, pp. 99–104, 2015.

[14] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, "An experimental study of security and privacy risks with emerging household appliances," in *Proc. IEEE Conf. Commun. Netw. Security*, San Francisco, CA, USA, Oct. 2014, pp. 79–84.

[15] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2015, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2744769.2747942

[16] J. Clover. (Apr. 2016). *Macrumors*. [Online]. Available: http://www.macrumors.com/2016/04/01/belkin-wemo-homekit-compatibility-on-hold/

[17] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[18] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things," in *Proc. 14th ACM Workshop Hot Topics Netw. (HotNets XIV)*, Philadelphia, PA, USA, 2015, pp. 1–7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834095

[19] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 137–142, 2003.

[20] S. Adlakha, B. Sinopoli, and A. Goldsmith, "Optimal sensing rate for estimation over shared communication links," in *Proc. Amer. Control Conf.*, New York, NY, USA, Jul. 2007, pp. 5043–5045.

[21] L. Hu, Z. Zhang, F. Wang, and K. Zhao, "Optimization of the deployment of temperature nodes based on linear programing in the Internet of Things," *Tsinghua Sci. Technol.*, vol. 18, no. 3, pp. 250–258, 2013.

[22] A. Jain, E. Y. Chang, and Y.-F. F. Wang, "Adaptive stream resource management using kalman filters," in *Proc. ACM SIGMOD Int. Conf. Manag. Data (SIGMOD)*, Paris, France, 2004, pp. 11–22. [Online]. Available: http://doi.acm.org/10.1145/1007568.1007573

[23] S. Li, L. Da Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2177–2186, Nov. 2013.

[24] A. Das and D. Kempe, "Sensor selection for minimizing worst-case prediction error," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, St. Louis, MO, USA, Apr. 2008, pp. 97–108.

[25] Z. Li *et al.*, "Energy cost minimization with job security guarantee in Internet data center," *Future Gener. Comput. Syst.*, vol. 73, pp. 63–78, Aug. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16307634

[26] C. D. Marsaon, "IAB releases guidelines for Internet-of-Things developers," *IETF J.*, vol. 11, no. 1, pp. 6–8, Jul. 2015.

[27] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, *From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices*. Berlin, Heidelberg: Springer, 2011, pp. 97–129, doi: 10.1007/978-3-642-19157-2_5.

[28] F. Li, M. Vogler, M. Claessens, and S. Dustdar, "Efficient and scalable IoT service delivery on cloud," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Santa Clara, CA, USA, Jun. 2013, pp. 740–747.

[29] R. Duan, X. Chen, and T. Xing, "A QoS architecture for IoT," in *Proc. Int. Conf. Internet Things 4th Int. Conf. Cyber Phys. Soc. Comput.*, Dalian, China, Oct. 2011, pp. 717–720.

[30] L. Li, S. Li, and S. Zhao, "QoS-aware scheduling of services-oriented Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1497–1505, May 2014.

[31] Q. Wu *et al.*, "Cognitive Internet of Things: A new paradigm beyond connection," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 129–143, Apr. 2014.

[32] S. Kumar, J. Paefgen, E. Wilhelm, and S. E. Sarma, "Integrating on-board diagnostics speed data with sparse GPS measurements for vehicle trajectory estimation," in *Proc. SICE Annu. Conf.*, Nagoya, Japan, Sep. 2013, pp. 2302–2308.

[33] J. E. Siegel, "CloudThink and the Avacar: Embedded design to create virtual vehicles for cloud-based informatics, telematics, and infotainment," M.S. thesis, Dept. Mech. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 2013. [Online]. Available: https://dspace.mit.edu/handle/1721.1/92230

[34] M. Chen, J. Yang, Y. Hao, S. Mao, and K. Hwang, "A 5G cognitive system for healthcare," *Big Data and Cognitive Computing*, vol. 1, no. 1, p. 2, 2017. [Online]. Available: http://www.mdpi.com/2504-2289/1/1/2

[35] M. Chen *et al.*, "Wearable 2.0: Enabling human-cloud integration in next generation healthcare systems," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 54–61, Jan. 2017.

Joshua E. Siegel (S'14–GS'14–M'16) received the bachelor's, master's, and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2011, 2013, and 2016, respectively.

He is a Research Scientist with the Mechanical Engineering Department, Massachusetts Institute of Technology. His current research interests include connected vehicles, pervasive sensing, and secure and efficient architectures for the Internet of Things.

Dr. Siegel was a recipient of the Lemelson-MIT National Collegiate Student Prize in 2015.

Sumeet Kumar received the bachelor's degree from the Indian Institute of Technology Kanpur, Kanpur, India, and the master's and Ph.D. degree in mechanical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2009 and 2014, respectively.

He was a Data Scientist with Facebook, Menlo Park, CA, USA, where he focused on developing machine learning algorithms and product analytics. His current research interests include machine learning, statistical signal processing, data science, sensor systems, and quantitative investments.

Dr. Kumar was a recipient of the Academic Excellence Awards in 2005 and 2006 and the Pappalardo Fellowship.

Sanjay E. Sarma (M'07) received the bachelor's degree from the Indian Institute of Technology Kanpur, Kanpur, India, the master's degree from Carnegie Mellon University, Pittsburgh, PA, USA, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA.

In 1941, he was the Fred Fort Flowers Professor and the Daniel Fort Flowers Professor of Mechanical Engineering and the Vice President for Open Learning, Massachusetts Institute of Technology (MIT). He co-founded the Auto-ID Center, MIT, and developed many of the key technologies behind the EPC suite of RFID standards now used worldwide. He was also the founder and the CTO of OATSystems, which was acquired by Checkpoint Systems (NYSE: CKP) in 2008. He has authored over 75 academic papers in computational geometry, sensing, RFID, automation, and CAD.

Dr. Sarma was a recipient of numerous awards for teaching and research including the MacVicar Fellowship, the Business Week eBiz Award, and the Informationweek's Innovators and Influencers Award.