Designing for Self-Configuration and Self-Adaptation in the Internet of Things

Arjun P. Athreya, Bruce DeBruhl, and Patrick Tague
Carnegie Mellon University
Email: {arjuna, debruhl, tague}@cmu.edu

***Abstract*—The Internet of Things (IoT) paradigm comprises a heterogenous mix of connected devices connected to the Internet. This promises a a wealth of opportunity for a large collection of distributed applications and services. However, the IoT introduces significant changes to the Internet model, largely in the form of billions to trillions of embedded devices that most likely will not be able to be managed centrally by cloud services due to lack of scalability. We suggest that the natural direction for IoT devices is to manage themselves, both in terms of their software/hardware configuration and their resource utilization. In this work, we descibe the underlying framework for self-managing devices, comprising measurement-based learning and adaptation to changing system context and application demands. In addition, we describe several upcoming research challenges in order to realize this self-management vision.**

***Index Terms*—Internet of Things; Self-management; Self-adaptation; Agent-based systems**

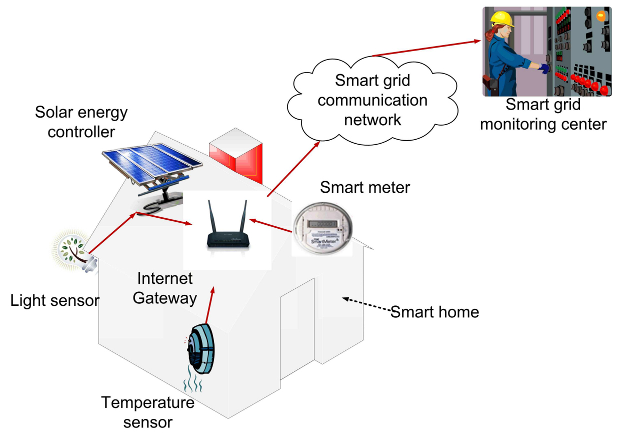

Fig. 1: An example deployment scenario for in-home and extra-home monitoring, analysis, and cloud service provision is illustrated in the context of smart energy management, e.g., the Smart Grid.

I. Introduction

The recent rise of the Internet of Things (IoT) paradigm comprising a heterogenous mix of devices referred to as things are connected to the internet. This paradigm has promised a wealth of opportunity for a seemingly unending collection of applications and services [1]. However, the IoT model introduces a vast array of user-less, interface-less devices into the Internet architecture, leaving these devices to either operate under remote or cloud-based control or to manage themselves. Due to the heterogeneity of devices – including sensors, actuators, storage devices, utility monitoring devices, mobile phones, network elements, and computers – and the sheer number of devices that are being connected to the Internet under the IoT umbrella, remote or cloud-based control appears to be a daunting task destined to suffer from limited scalability. Hence, the natural direction for IoT devices is to manage themselves, both in terms of their software/hardware configuration and their resource utilization (energy, communication bandwidth, medium access etc.). We call this capability of device managing themselves as *self-management*. An example deployment scenario for an IoT system is illustrated in Fig. 1.

In order for the various network components to operate and interoperate effectively, devices must be able to coordinate their management capabilities. However, the efforts taken by one device to optimize its performance should not negatively affect neighboring devices or services. Hence, a level of collaboration between software components operating on a device or across device platforms becomes a factor in resource-limited IoT domains. Moreover, effective self-management of IoT devices, resources, and services requires agile behaviors in all aspects of device software and configuration. This leads us to re-envision the software, hardware, and network architectures used for IoT designs.

In this work, we first take a preliminary look at these software, hardware, and network architectures involved in IoT systems. We then summarize recent research results that can be applied to IoT internetworking, and present significant challenges and opportunities for IoT research in the near future. We discuss these challenges by referring to a basic IoT architecture that can be envisioned for the IoT. Our efforts in this work are summarized in the following contributions.

- We present a generalized optimization framework that allows software agents to manage and control protocol parameters and behaviors.
- We describe various components of internetworked IoT devices that can be adaptively reconfigured using measurement, on-the-fly statistical analysis, and parameter tuning.
- We highlight several aspects of the IoT system architecture that must be re-evaluated to support the agent-based adaptation capability.

The remainder of this paper is outlined as follows. We discuss the inherent benefits of self-management and self-configuration in IoT systems in Section II, and we discuss the corresponding self-configurable IoT components in Section III.

COLLABORATECOM 2013, October 20-23, Austin, United States
Copyright © 2013 ICST
DOI 10.4108/icst.collaboratecom.2013.254091

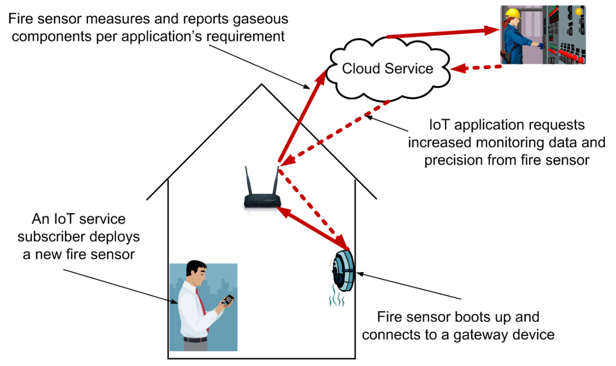
Fire sensor measures and reports gaseous components per application's requirement
Cloud Service
IoT application requests increased monitoring data and precision from fire sensor
An IoT service subscriber deploys a new fire sensor
Fire sensor boots up and connects to a gateway device

Fig. 2: We demonstrate the idea of self-management in the context of a fire sensor being deployed. The sensor boots up, connects to the gateway device and establishes communications with an application hosted in the cloud. It sends measurement data to the application at a pre-configured rate and precision. However, the application could monitor the local-environment and could request for the device to alter its sending rate or precision in data. To this request, the fire sensor responds through self-adaptation.

We propose a generalized IoT architecture to support self-configuration and self-adaptation in Section IV. We discuss significant research challenges in Section V and conclude with a few notes on future work in Section VI.

II. Self-Management in the IoT

The basic idea underlying self-management is that devices, software, or services can configure, evaluate, and adaptively reconfigure themselves in response to changes in performance, resource availability, or external dynamics. In the context of the IoT domain, this can entail reconfiguration of sensing parameters, network utilization, spectrum allocation, data reporting, data and control signal protection, interaction with cloud services, and a wide variety of software configuration details. In what follows, we highlight the primary aspects of self-configuration, measurement-based analytics, and self-adaptation in general, describing the high level techniques that enable these features and a coarse cost-benefit analysis of self-management.

A. Self-Configuration

The act of self-configuration takes place either immediately after deployment of new devices into a local IoT domain or after significant changes are made that warrant a clean-slate restart to the system configuration. Self-configuration primarily consists of the actions of neighbor and service discovery, network organization, and resource provisioning as shown in Fig. 2. However, the ways in which these actions occur for a particular device depend on the application or service scenario, the system context, the role of the device in the IoT domain, and the capabilities of the device. Since every device is typically aware of its own roles and capabilities within the system, the most important tasks of self-configuration are to coordinate with other devices in the local IoT domain and participate in network organization and resource provisioning.

B. Measurement-Based Self-Evaluation

Once the system is configured and operating, various performance aspects of the system operation can be monitored, either by the participating devices or by local network elements. As we will describe in more detail in Section IV, we suggest that each device be equipped with a software agent responsible for measuring performance aspects locally, either by passively observing the communication over the wireless medium or by querying software running on the device for a variety of performance metrics (e.g., packet data rate from a routing daemon). At a high level, a particular *protocol agent* responsible for evaluating and managing a single protocol can collect a set of measurements ϕ_m, $m = 1, \ldots, M$, corresponding to relevant observed events or a statistical summary of such events; more concisely, the measurements can be abstractly represented by a measurement vector $\boldsymbol{\phi}$. There are numerous observable events and statistical representations that can be used to evaluate and control each protocol, and each has different value to different protocols of interest. For example, per-device or per-link statistics about the queueing delay, received signal strength, and packet drop ratio can be useful to the routing protocol in selecting the most reliable routing path [2]. Any such desirable performance characterisitics can be captured by a suitable utilty function $\mathcal{U}(\boldsymbol{\phi})$, which provides a target objective function for optimizing the system on the fly.

C. Self-Adaptation Framework

The control parameters that can be tuned to optimize the protocol utility $\mathcal{U}(\boldsymbol{\phi})$ can be described as input parameters π_k, $k = 1, \ldots, K$, corresponding to a specification of the free or tunable parameters for the protocol of interest, similarly representable by a vector $\boldsymbol{\pi}$. As with the measurement parameters $\boldsymbol{\phi}$, there are numerous output parameters that can be evaluated for each protocol of interest. For example, the transmission power used for each link in a routing path can be tuned to achieve a desired tradeoff between resource expenditure and the resulting reliability of the routing path. As with any feedback system, there is a clear circular relationship between $\boldsymbol{\phi}$ and $\boldsymbol{\pi}$, namely that protocol options $\boldsymbol{\pi}$ used for communication influence the resulting observed statistics $\boldsymbol{\phi}$. Hence, the goal of a device trying to optimize its performance will estimate the mapping $\mathcal{M} : \boldsymbol{\pi} \mapsto \boldsymbol{\phi}$ and select options $\boldsymbol{\pi}$ to maximize a utility function $\mathcal{U}(\boldsymbol{\phi})$ that can also be expressed as a function $\mathcal{U}(\mathcal{M}(\boldsymbol{\pi}))$ of the design variable. However, this optimization formulation is also constrained by various factors, including resource availability, scheduling constraints, traffic capacity, etc., so additional constraints may be enforced on the selection of parameters $\boldsymbol{\pi}$. In general, the set of constraints can be represented by the vector inequality $\boldsymbol{\Gamma}(\boldsymbol{\pi}) \geq \boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is a vector of constants. The optimal protocol configuration is thus given by the solution

$$\boldsymbol{\pi}^* = \arg\max_{\pi} \mathcal{U}(\mathcal{M}(\boldsymbol{\pi})) \quad \text{s.t.} \quad \boldsymbol{\Gamma}(\boldsymbol{\pi}) \geq \boldsymbol{\gamma}. \tag{1}$$

We refer to this formulation as being context-aware since the utility function $\mathcal{U}(\cdot)$ depends on the context of the communication scenario and the constraints capture the broader context of the networked system.

In a realistic scenario, there are a number of practical aspects of this problem formulation that must be taken into account, some of which present significant challenges. First, due to network and resource dynamics, the system mapping $\mathcal{M}$, constraint function $\mathbf{\Gamma}$, and constant γ may vary in time. Hence, instead of a one-shot optimization problem that can be solved once during network initialization or pre-configuration, each protocol agent may continually update the parameters of its communication protocols. At a time instant t, each protocol agent must repeat the optimization problem of solving

$$\boldsymbol{\pi}_t^* = \arg\max_{\pi} \mathcal{U}(\mathcal{M}_t(\boldsymbol{\pi})) \quad \text{s.t.} \quad \mathbf{\Gamma}_t(\boldsymbol{\pi}) \geq \gamma_t. \tag{2}$$

Second, in a real network, the true values of the mapping $\mathcal{M}$, constraint function $\mathbf{\Gamma}$, and constant γ will be unknown to the device. This uncertainty thus requires the use of statistical techniques to provide a reasonable estimate for each unknown parameter and additional mechanisms to compensate for the uncertainty in the statistical estimate. Hence, statistical estimates $\widehat{\mathcal{M}}$, $\widehat{\mathbf{\Gamma}}$, and $\widehat{\gamma}$ and the corresponding uncertainty function η can be incorporated as

$$\boldsymbol{\pi}_t^* = \arg\max_{\pi\in\Pi} \mathcal{U}\left(\widehat{\mathcal{M}_t}(\boldsymbol{\pi})\right) - \eta_t(\boldsymbol{\pi}) \quad \text{s.t.} \quad \widehat{\mathbf{\Gamma}_t}(\boldsymbol{\pi}) \geq \widehat{\gamma}_t. \tag{3}$$

We believe that the constant learning and continual optimization procedures suffice to capture the near-real-time capabilities necessary to provide context-aware communication. In order to achieve these capabilities, however, several practical implications must be addressed in the repeated optimization formulation in (3). First, in order to set up the optimization in the first place, the device must be able to gather enough information through observation or measurement to construct an estimate of each unknown parameter and the corresponding uncertainty. Second, since the optimization procedure is repeated continually, the time required for parameter estimation and computation of the optimal parameter set $\boldsymbol{\pi}$ should be less than the (expected) interval Δt between adaptation steps. Third, the adaptation interval should be quick enough to provide timely parameter estimates, as network dynamics may cause estimates to become stale. Thus stale data means that the context under which the device reported the data would have also changed. With these considerations in place, an important aspect of our investigation is the tradeoff between accuracy of optimal solution and speed of heuristic selection.

III. Self-Managing IoT Components

Given the high level framework for self-management in IoT systems, we next turn our attention to describing some of the individual components of the IoT that can take advantage of the self-management capability. In order to discuss these components, we first describe a basic network model for the IoT. Then we describe how applications, communication and networking protocols, and security considerations can be incorporated as components into a self-managing IoT system.

A. Network Model

Our network model comprises three components, namely local environments, gateway devices and cloud services. *Local environments* constitutes a physical area with devices deployed that are communicating with a service through the internet. *Gateway devices* are those which can act as an interface between other devices of the local environment and the internet. *Cloud services* are software services hosted in the internet by third parties that communicate with devices in local environments and make meaningful representations of the data collected. In Fig. 1, we demonstrate this network model where the local environment is a home comprising of various devices such as sensors, controllers and the smart-meter. These devices connect to the internet via a gateway device such as a IEEE 802.11 router. These devices then connected to a service such as an energy monitoring application hosted by an utility provider. Thus the application can query these devices for data, which can then be used to let application subscribers know of their energy consumption and recommendations for saving energy based on historical consumption data.

B. Application

Applications in the IoT query various devices in local environments and project them on user interfaces that users can interact with. We discuss the various instances the application demands certain features in the IoT architecture.

1) Context-aware Adaptation: Several applications of the IoT are expected to query data from critical infrastructures or monitoring critical situations. Examples of these are bridges, power plants, hospitals and patients in intensive care. In all these examples, events could occur any time and could be critical [3]. These events to name a few could be natural disasters or abnormal changes in vital parameters. During these events, applications monitoring them would want to obtain a constant stream of data from these devices. This would aid in deploying immediate responses and also create awareness among neighboring environments of this critical event. Thus the constant stream of data from these devices means that communication resources need to be made available on-demand and has to be sustained till the event is over or the application ceases to need constant stream of data.

Thus making communication resources available on-demand could mean that rate-adaptation has to be done for other devices in the local environment. Alternatively, opportunistic communications that use radio resource diversity could also be used during these events [4]. In either of these approaches, network self-configuration to meet these demands of the application has to be done in a quick and reliable way. Therefore local awareness of the resource availability in the IoT is key to this process of network self-configuration. Thus this means that diversity in communication capabilities of devices and gateways in local environments benefit the application's demands during extreme events in local environments.

2) Re-programmable Interfaces: The IoT applications are expected to evolve based on the value the data creates. For example, an indoor weather monitoring application currently monitoring temperature, now needs to report humidity and air pressure data. Though the sensors on board could generate and send those data individually, the application might need it in certain formats that enables efficient processing in the back-end infrastructure such as the cloud. These message formats are dependent on how the device is programmed in the operating system. Thus any change to the format means that software code has to be changed and the code needs to be pushed down to these devices and reprogrammed on the fly. Re-programmable devices today are common, such as FireFly [5] and Electric-Imp [6]. Making this possible is also the availability of light-weight operating systems. Such operating systems enable programmers to change formats for data packets and even allow for simple pre-processing of data before being transmitted [7]. Therefore application programming interface (API) for such devices and computing platforms are needed along with the network bandwidth that allows for software code to quickly propagate to the devices in local environments. Here, we are not referring to a dozen devices, but such devices could be in thousands if the environments are large. Therefore scale and latency have to be addressed with efficient and reliable networks, while APIs allow for re-programmability of devices.

3) Energy-aware computing: Like any other man-made infrastructure, the IoT is also prone to device and network failures. Networks within local environments are expected to be hierarchical. Devices acting as parents to child devices in the hierarchical network aggregate and forward data to their parents which eventually reach the root device of the hierarchical network. Device and link failures in such networks means that data that was being forwarded in the region of failure will be lost and network connectivity could be disrupted. Additionally, if energy powered devices rely on battery to support their functioning, then energy-awareness among devices is very important.

Energy consumption of devices can be measured for all kinds of operations and the device can be trained to predict its energy costs for participating in network self-organization and other computing operations. Network self-organization aims at reconnecting devices when provision communications fail [8]. In our proposed architecture, devices which are aware of energy costs for any kind of operations (as a function of machine cycles or bytes of data for communication), they can vote to participate in specific functions requested by the IoT applications. This allows for improving network longevity and builds resilience into the system. It remains an open question if energy-awareness affects the application's performance due to restrictions posed by devices for their self-interests. Therefore game theoretic approaches that maximize the utility of the local environment's devices without totally compromising the device's interests would be one approach to solve this problem.

C. Communication and Networks

If the physical (PHY) and multiple access (MAC) layer will meet all our future needs they will look markedly different. Currently most networks are designed around single protocols at both the physical and MAC layers allowing for limited freedom. Likewise, these layers provide little or no feedback to the upper layers and are generally agnostic to application needs. This lack of cross-layer and application awareness at the lower layers severely limits the effectiveness of IoT networking. Current physical and MAC protocols also adapt in very limited ways to the complex and rich multi-node environment. Examples of current adaptation are limited to rate and modulation adaptation based purely on self-performance. Given these limitations we propose a change in the design paradigm to better allow for self-manageable IoT networks.

We propose that the future of the IoT will take advantage of intelligent and opportunistic MAC and PHY layers by being aware of context from surrounding nodes. By context aware we propose that a node can assess the surrounding environment and use the assessment to best accomplish its current goals in a similar manner to cognitive radio networks [4]. This is a useful modification to the current radio architecture because it allows for nodes to more intelligently use and reserve resources.

To allow for local context awareness we anticipate the use of flexible multi-radio and multi-band system systems that allow that allow for data for most applications to be sent in a plethora of ways. It is important to be able to send data in a multitude of ways to increase robustness, create diversity, and allow for coexistence in a crowded network. The use of multiple-bands allows for nodes to communicate in a more diverse way as well as allows for increases in security by improving interference avoidance. Similarly, having multiple radio protocols allow for adjustments based on the available bandwidth. To effectively use multi-radio, multi-band, multi-modulation transceivers we must design intelligent ways to assess the spectrum for usability and techniques for consensus spectrum management across many nodes.

Not only should the future IoT be aware of information from lower layers but it should also consider upper layer demands in determining its physical and MAC usage. For example, a video application that wants a constant rate of communication can be much better served by a contention-free medium access scheme (such as Time Division Multiple Access) at the PHY layer while a time impervious data application may be better served by getting continuous access to the channel. Because of this we propose that applications and networking protocols be allowed to request resources in the lower layers. This allows for the communications to occur in such a way that is best for a particular application. Given these advancements in communications a device must still provide interoperabiilty with legacy systems. To do this, radio protocols must be able to revert from intelligent adaptation and scanning and allow for legacy communications.

D. Security

With the devices in the IoT being envisioned to support a wide range of applications, sensitive data and commands are expected to flow to and from these devices through the network. This opens a Pandora's Box for all kinds of security problems in a self-configuring architecture for the IoT. The reasons for this are multi-fold.

The heterogeneity in the devices does not always guarantee support for all possible security features. It could be either limited in hardware or software capabilities to support only a subset of them. For example, sensor platforms could perform simple symmetric cryptographic computations such as the RC4, AES or DES. But they might not have the capability to support public-key cryptographic operations such as digital signatures even if there exists a key infrastructure for the IoT.

Authentication of remote commands from the application service or other devices in the IoT is a challenging problem. Authentication either in the form for a Message Authentication Code (MAC) or a digital signature requires pairwise shared symmetric keys or keying infrastructure in place respectively. Metrics are needed to evaluate the different levels of security needed to authenticate the remote control commands for devices in the IoT. These metrics will also help develop the necessary keying infrastructures for the desired security levels. Otherwise, external attackers can issue commands to devices that could alter the operations or goals of the interactions between devices in the IoT.

A root of trust has to be defined for the IoT to ensure that self-reconfiguration commands, software and firmware updates can be validated. Today, plenty of micro-controller based devices allow for code to be pushed to the hardware through the internet or be programmed via the cloud. An example of such a platform is the Electric Imp [6]. The root of trust in the IoT could be the application service hosting entities, hardware vendors, software vendors or these cloud based services. Thus the gateway of the local environment has to have a list of trusted entities whose commands or updates can be trusted and be allowed to interact with other devices in the local environment.

An important security property of a self-configurable architecture is the verfiability of code on the devices in the IoT. Applications can verify that their code was downloaded and installed correctly at the time of installation. However the device could have downloaded malicious code later that affects the functioning of trusted code [9]. Side channels, binary exploits and other attacks we have seen in software systems are still applicable to devices in the IoT. Hence, the applications in the IoT should be able to remotely verify code running on devices. If existing solutions for securing software systems that use Trusted Platform Modules and Platform Configuration Register extension mechanisms suffice the IoT's code verifiability needs is an open question.

As several applications could be querying the device for the same or different data, isolation of the application memory space and access controls on devices is needed. This ensures that though multiple applications access the same space, they cannot modify or read data that they are not supposed to. This also helps in preserving privacy of the data from a user's perspective.

IV. Architectural Support for IoT Self-Adaptation

Give the optimization framework and the examples of components that can take advantage of the self-management capability, we next turn out attention to the architectural needs to support self-managing or self-adapting IoT components.

A. Collaboration and Competition Considerations

In the above formulation, each adapting protocol agent continually solves an optimization problem relating its parameter choices to a utility function for the protocol of interest. In a real system, parameter choices by an agent i will affect the performance observed by a neighboring agent j, and vice versa. Hence, the optimization problems solved by neighboring agents i and j are coupled. If i and j are "team players", they may be interested in making choices that are mutually beneficial, or at least not disruptive to each other. Hence, instead of each agent simply maximizing its own utility $\mathcal{U}^{(i)}(\boldsymbol{\pi}^{(i)})$, the team of cooperating agents may instead choose to maximize a global utility $\mathbb{U}(\{\boldsymbol{\pi}^{(i)}, i \in \mathcal{I}\})$ using concepts such as proportional fairness [10]. Alternatively, if i and j are opponents on neighboring devices, they may not be interested in achieving a global utility. For example, if j is mounting a denial of service attack against i, then i may try to directly maximize its own utility while j may try to indirectly minimize i's utility, both subject to their own constraints. Even without knowing *a priori* whether the situation is cooperative, competitive, or adversarial, the process of learning the mapping $\mathcal{M}$ and system context will allow each agent to act accordingly. Moreover, if cooperating agents are allowed and able to exchange information about their differing contexts, a larger-scale distributed optimization problem can be solved by a higher-level coordinating agent, for example using the concept of network utility maximization (NUM) [11].

In addition to the cooperative/competitive aspects of the formulation, we must consider various cross-layer aspects of protocol adaptation, as some parameter choices have opposite effects on performance characteristics at different layers of the protocol stack. For example, increasing transmission power over a wireless link can improve the reliability or data rate of the link, but this increase power will lead to increased contention and longer forwarding delays over a larger neighborhood. Hence, even within a single device, various metrics and competing utility functions must be considered.

To address the problem of balancing utility across protocol layers and across devices, we propose the use of an *analysis and control agent* on each device that is reponsible for coordinating the individual protocol agent operations on the device. The analysis and control agent (ACA) is thus responsible for the entire process of measurement, inference, and optimization with respect to the entire network stack and using information and feedback from neighboring devices. This ACA will be made available to the various protocols running on the device through a common, open interface that can pull performance information from the network stack and push parameter values

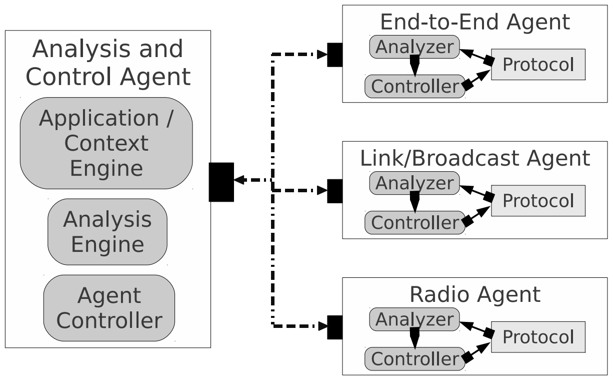

Fig. 3: We propose to modify the network protocol stack by replacing the collection of layers with an interactive collection of protocol agents and an overseeing analysis and control agent. Individual protocol agents can act using their own observations, possibly in coordination with other agents and the primary control agent. Existing protocols can be incorporated by instrumenting them with suitable data logging and reporting via interfaces to other agents.

into corresponding protocol agents and configuration files. The coordination between the individual protocol agents with the ACA allows the individual protocols to operate transparently while still getting the benefits of the context-aware adaptation capability. ACAs on neighboring devices can use in-band or out-of-band communication to coordinate their decisions, or each agent can rely on measurement and inference to learn and respond accordingly. The resulting agent-based adaptive protocol architecture is illustrated in Fig. 3.

B. Agent Measurements and Context Inference

In the previous sections, we have provided the groundwork for distributed, context-aware adaptation of protocol parameters across layers of the network stack using coordinated protocol agents. However, the optimization framework provided in Section II relies on the ability to learn the system context through measurement. In what follows, we describe our efforts toward efficient and accurate methods to enable context inference. As with the optimization framework, we present the statistical inference capability in general, noting that it can be applied for both offensive, defensive, cooperative, or competitive purposes.

We propose a statistical inference model in which each protocol agent makes observations ϕ about their own effect on the system as well as the previous actions that collaborating or opposing agents have made. Each agent i in our model observes samples of the individual utility function $\mathcal{U}^{(i)}(\cdot)$ as a function of its input parameters $\boldsymbol{\pi}^{(i)}$ and the inputs of other agents. Each agent then uses these observations to construct an estimate $\widehat{\mathcal{U}^{(i)}}(\boldsymbol{\pi}^{(i)})$ of the utility function. We refer to the collective information about utility functions and parameter history as the agent's knowledge, such that the more knowledge an agent has, the more likely it is to obtain an accurate estimate of the system context to optimize protocol parameters. However, whenever the system context changes, for example when an agent changes their parameters or objective function, knowledge becomes stale and can actually be detrimental and lead to sub-optimal parameter choices. Hence, each agent must continually update its knowledge by forgetting events of the distant past and continually updating its knowledge of system context.

In our preliminary work on this topic, we have identified two candidate approaches for measurement-based inference and context learning; we refer to these approaches as *weighted observation* and *universal approximation* [12]. The two formulations have different features in terms of computation time and accuracy, and evaluating these features will be part of the proposed task.

We first introduce the weighted observation approach which uses samples of the utility function to construct a weighted preference for different candidate parameters to construct a solution with better-than-average utility over time than that of a random parameter selection. In weighted observation, agent i constructs a vector $\boldsymbol{u}_i$ with each entry indicating the utility of the corresponding candidate parameter set $\boldsymbol{\pi}^{(i)}$. Using the utility vector $\boldsymbol{u}_i$, agent i then computes a weighting vector $\boldsymbol{\omega}_i$ by transforming and normalizing $\boldsymbol{u}_i$ as

$$\boldsymbol{\omega}_i = \frac{e^{\kappa \boldsymbol{u}_i}}{\|e^{\kappa \boldsymbol{u}_i}\|}, \tag{4}$$

where κ is a scaling constant to indicate the preference for choosing candidate solutions with higher estimated utility. The weighting vector $\boldsymbol{\omega}_i$ is thus used as a probability distribution for choosing protocol parameters $\boldsymbol{\pi}^{(i)}$, similar to the game theoretic concept of a mixed strategy [13]. After each parameter choice and action that agent i makes, it can then observe the resulting system utility $\boldsymbol{u}_{obs,i}$ corresponding to the play and update its utility estimate $\boldsymbol{u}_i$ accordingly. One possible method for this update is to fuse the observed utility with the previous estimate by simply replacing $\boldsymbol{u}_i$ as

$$\boldsymbol{u}_i \leftarrow \lambda \boldsymbol{u}_i + \beta \boldsymbol{u}_{obs,i}, \tag{5}$$

where $\beta, \lambda \geq 0$ determine the balance between memory of historical knowledge and preference for recent events. We note that a relatively smaller λ value means historical knowledge is forgotten more quickly and a relatively larger β value gives preference to recent events but may also introduce high variance in the utility estimate due to changes in the system. The updated utility estimate $\boldsymbol{u}_i$ is then used to compute the weighting vector $\boldsymbol{\omega}_i$ using (4), and the process repeats continually in near real time. We note that the weighted observation process has low complexity and can be tuned to quickly adjust the agents' knowledge when system dynamics are fast, providing the computation speed needed in near-real-time scenarios.

We next introduce the universal approximator approach based on biologically-inspired neural networks [14] to estimate the utility function $\widehat{\mathcal{U}^{(i)}}(\boldsymbol{\pi}^{(i)})$ based on observed utility samples $\boldsymbol{u}_i$ by training the neural network model. The neural network is trained by selecting a set of τ random parameter values $\boldsymbol{\pi}_i$, measuring the resulting utility $\boldsymbol{u}_i$ for each parameter $\boldsymbol{\pi}_i$, and setting neural activation functions using a universal neural network approximator. Once the neural network is trained, it provides a static estimate of the utility function that can be used for an appropriate duration. However, when the system is dynamic, the neural network must be continually re-trained to compensate for changes to the system context, so a periodic or random reset is required, similar to the previous process of forgetting knowledge, though less gradual. The value of the universal approximation technique thus relies on the relationship between the number of training samples τ and the rate of system dynamics, meaning it is likely more valuable for cases with slow dynamics.

In both of the above cases, extensions to an arbitrary number of agents and protocol levels can be incorporated into the statistical inference models, through the parameter space dimensionality quickly becomes problematic, so further investigation of these and other statistical techniques is needed.

V. Challenges for Self-Configurable IoT Systems

There are several significant research challenges between the current state-of-the-art and a realization of our agent-based self-management approach for on-the-fly tuning of protocol parameters in IoT devices. These span across almost every layer of the architecture we have described. In what follows, we highlight a few of these research challenges which comprise future work on the topic before a large-scale deployment.

Development of Suitable Metrics: As with any modern problem in networking, security, or system optimization, a lot of the effort boils down to designing the right metrics. In our framework, we abstract the idea of specific metrics and address only a generic utility function $\mathcal{U}$. While this utility function could simply incorporate standard performance metrics such as spectrum/bandwidth utilization, energy consumption, network throughput, or latency, many modern applications have very different types of demand that are not effectively captured by these classic metrics. For example, in a distributed sensing application, the data rates are very low, moderate latencies can be tolerated, and some data can be lost entirely due to temporal and spatial correlation in the sensor data streams. Hence, identifying the most suitable metric for a particular IoT system context remains a challenging problem.

Coordinated Contextual Intelligence: Even with our proposed statistical analysis techniques to provide context estimation and on-the-fly optimization of free protocol parameters, we have not addressed the issue of coordinating the optimization steps among agents or across devices. As previously described, parameter changes that can be highly valuable to one protocol can be highly detrimental to another protocol. The previously described example of power control (higher power providing more routing opportunity but increasing contention at the MAC layer) illustrates the potential conflicts or trade-offs that exist, even on the same device. Fundamental to any agent-based system, we need to address this coordination issue among agents. Most likely, this will come down to identification of a number of trade-offs that can possibly be controlled on-the-fly along with protocol parameters. Even if all devices are trusted or amiable this is a difficult resource allocation problem, but if any devices are malicious or greedy, this becomes a very difficult problem which needs novel solutions.

Energy Awareness: Energy-aware computing is essential to the IoT. With the scale of deployment envisioned, every unit of energy saved is paramount to the entire IoT being green as a system. Also, energy-awareness during times of failure helps improve longevity and support critical applications when they are needed the most. Thus, how energy calibration and energy models across the heterogeneous system of the IoT will need new algorithms and inter-device interaction protocols that considers energy as a constraint.

Incentives for Self-Management: Although not addressed explicitly in this paper, we have considered the problem of self-management from the economic perspectives of many of the parties involved in the IoT. For users, IoT self-management translates to increase service quality and cost savings, primarily through reduced resource consumption of in-home devices and increased overall performance and system utilization. Similarly for businesses, increased service quality and cost savings can be realized because of the increased overall performance and resource cost reduction. For network providers, increased system utilization means more service can be provided for the same unit of bandwidth or resource utilization, effectively translating to more revenue per unit investment. It remains an open challenge, however, to push these benefits to the point that they're worth the initial R&D investment required to develop the systems with the correct contextual intelligence to bring maximum value.

Resilience to Failures, Outages, and Attacks: While we have done some initial work on improving failure- and outage-resilience in IoT and Smart Grid systems [15], [16], [8], there is still a long way to go to make IoT systems truly resilient. Our intial work has proposed network mechanisms to re-organize the network after failure or outage events, but there are some limitations of our approaches and there are several trade-offs involved in our designs. The issue of failure-tolerance and attack resilience in self-managing IoT systems remains open.

Application Integration: Even with a deeply integrated self-management architecture for the IoT, there is no value if the applications are not designed to take advantage of the architecture. Hence, development of our self-managing systems requires application and service developers to use the set of APIs that we create, otherwise many of the benefits will be lost. Making context-aware networking usable is a major challenge. Designing usable and secure APIs that allow for fair and understandable use of such a system is a major problem. Work must be done to make the application framework and APIs usable, including aspects of how how the device's OS can confirm application requests, how a network of devices can verify the request, and how to provide this across a physical medium.

VI. Conclusion

In response to the emerging Internet of Things paradigm comprising a heterogenous mix of connected devices and numerous distributed application and services running over wireless networks of embedded devices, we have proposed the use of self-management and self-adaptation to cope with countless dynamics. We have presented an underlying framework for self-managing devices, comprising measurement-based learning and adaptation to changing system context and application demands. In addition, we described several upcoming research challenges in order to realize this self-management vision that comprise our future work on the topic.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '04, 2004, pp. 133–144.

[3] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emergency response: challenges and opportunities," *Pervasive Computing, IEEE*, pp. 16–23, 2004.

[4] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, no. 13, pp. 2127–2159, 2006.

[5] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, pp. 183–231, 2007.

[6] "Electric Imp," http://www.electricimp.com.

[7] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: an energy-aware resource-centric RTOS for sensor networks," in *26th IEEE International Real-Time Systems Symposium (RTSS)*, Dec. 2005.

[8] A. P. Athreya and P. Tague, "Network self-organization in the internet of things," in *IEEE International Workshop on Internet-of-Things Networking and Control (IoT-NC)*, to appear.

[9] B. Parno, J. M. McCune, and A. Perrig, "Bootstrapping trust in commodity computers," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP '10. IEEE Computer Society, 2010, pp. 414–429.

[10] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[11] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[12] B. DeBruhl and P. Tague, "Living with boisterous neighbors: Studying the interaction of adaptive jamming and anti-jamming," in *3rd International Workshop on Data Security and Privacy in Wireless Networks (D-SPAN)*, Jun. 2012.

[13] M. J. Osborne and A. Rubenstein, *A Course in Game Theory*. MIT Press, 1994.

[14] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, Dec. 2006.

[15] A. P. Athreya and P. Tague, "Survivable smart grid communication: Smart-meters meshes to the rescue," in *2012 International Conference on Computing, Networking and Communications (ICNC)*, Jan./Feb. 2012, pp. 104–110.

[16] ——, "Self-organization of a mesh hierarchy for smart grid monitoring in outage scenarios," in *Proceedings of the 4th IEEE PES International Conference on Innovative Smart Grid Technologies*, 2013.