A Vision of a Future IoT Architecture Supporting Messaging, Storage, and Computation

Van Nguyen and Audrey Gendreau

***Abstract*—In the future, things in the Internet of Things should be able to provide solutions for any task that we ask them to perform. Imagine that we would have hundreds of billions of devices connected to the Internet, each of these devices possibly has hundreds of gigabytes of storage and memory, a magnitude of giga instructions per second, and a bandwidth of gigabits per second. When these devices are connected in order to holistically work together, we would almost have an unbounded capacity. This paper introduces an architecture that utilizes all the available resources of the things on the Internet in hope to provide an unprecedented computational power to benefit society. There are three important features that a thing in this architecture could support: messaging, storage, and/or computation. With this architecture, a thing would be capable of discovering, working, and sharing with other things on the Internet. This architecture is based on the service-oriented concept to provide ease of use and implementation.**

***Index Terms*—Internet of Things, messaging, storage, computation, distributed computing, machine-to-machine, services, virtualization, communications.**

I. Introduction

The Internet of Things (IoT) is a network of objects connected to each other and their owner. These object applications are weaved into the fabric of everyday life to make it easier. According to [1], by 2022 the IoT evolution will have resulted in a trillion IP addresses. Fundamentally, this new technology is based on a vast distributed heterogeneous network that reports on the physical environment and each other. It combines Local Area Networks (LANs), Wide Area Networks (WANs), Wireless Sensor Networks (WSNs), Cellular Phones, Radio Frequency Identification (RFID), and the Internet to form for the first time a network of the most widely distributed technologies with different software and hardware capabilities. According to [1], the IoT's coverage will be so extensible that everything will be monitored and tracked to provide a new kind of knowledge and awareness. To reach this goal, services that can indiscriminately share both the networks physical and virtual resources will be important in order to effectively perform at this unprecedented level of distributed computing as shown in Fig. 1.

Already, Service-Oriented Architecture (SOA) is the subject of interests in the IoT's research community [2], [3], [4], [5]. For one reason, services have traditionally been used to provide generic facilities that may be used by a wide variety of distributed technology. As a middle-ware, services architectures allow application objects to communicate with one another irrespective of their programming language, their hardware and software platforms, or their networks. The ability to mash-up services further utilizes the diversified IoT's infrastructure [4]. Another reason to use a SOA and the focus of this paper is to enable concurrent processing and to exploit the available computational resources. However, the IoT's unique data driven mobile platforms have limited computational capabilities. For the sensors to be close to the physical environment that they are monitoring requires them to reside on tiny computers; therefore, distributed computing agents analyze and process events to collectively achieve the applications task. Regardless of these resource limitations, we would like to propose a SOA that can apply similar distributed computational techniques to dissimilar networks and objects within the IoT. For example, using discovery services and context awareness, a phone placed in a designated geographic location can share its computational capability with the other local application objects. Moreover, it is proposed that these services will be extended to discover both virtual and physical capabilities over the Internet within a network that has selected to opt in to publicly shared resources in the IoT.

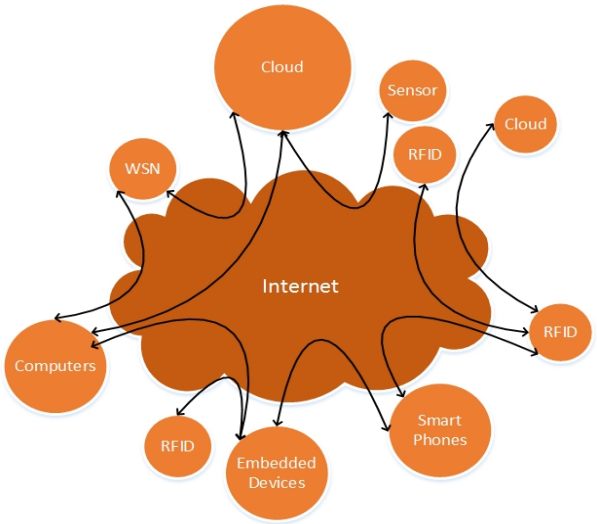

Fig. 1. The IoT.

While this concept initially may seem a good idea; as discussed today's technology is resource constrained, e.g., with limited communication, storage and computing capabilities. Therefore, the premise of our SOA approach is that the rapidly evolving technology for the IoT will soon overcome the current platform constraints. It may be bold, but this is not an unrealistic expectation. Introduced just two

Manuscript received August 22, 2014; revised October 24, 2014.

The authors are with Department of Computer Science, Saint Leo University, USA (e-mail: van.nguyen, audrey.gendreau@saintleo.edu)

DOI: 10.7763/IJFCC.2014.V3.337

decades ago, WSNs and RFID communication technology was originally driven by the need for embedded sensing devices to communicate with one another in order to track objects over time [6]. At the turn of the 21st century researchers determined that the unique physical characteristics of sensing devices demanded new communication requirements be included in the protocol stack. Soon after, specialized communication protocols for WSNs began to be researched and published, as discussed in a WSN survey paper by [7]. Since then, many specialized communication protocols have been developed to form the prevalent WSN architecture used today. Currently, the technological focus is on the IoT. Not long ago, the volumes of information generated by these data driven networks and the limited number of addresses available were barriers to its success. To overcome the first barrier Cloud computing has provided a method for the anticipated volumes of generated data to be stored and analyzed. The second barrier was resolved with the creation of IPV6 and the abundance of addresses available for the objects. It enabled 6LoWPAN, an IPv6 over Low Power Wireless Personal Area Networks protocol, to recently be adopted as a standard for WSNs and RFID readers to communicate over the Internet [2]. Hence, again the major issues that prevented the targeted technology from evolving were addressed in order to establish the first phase of the IoT.

As the trend to overcome hardware and software barriers continues, it is surmised that the power of the IoT will unfold to be able to share messaging, storage, and computational resources over the internet. According to [8], by 2025 the objects, and thus their resources on the Internet may reside in everyday products such as food packaging, furniture, paper documents as well as many others. While this may create new problems, it also poses a great potential for new reserves of storage, and computational power to be utilized by the services. Our SOA, denoted as Hecate, will regulate the new object resources by facilitating message services supplied by a family of IoT software products. These software products will allow independent applications to communicate with one another and attain resources wherever available. Moreover, they will provide the necessary tools to be the foundation for other services to build upon.

This paper is structured as follows. Section II present a brief review of the literature needed to set the foundations of the presented work. Section III is the framework of the Service-Oriented Architecture named Hecate presented in this study. Section IV compares features and examines the architecture from another perspective. Section V presents examples of the devices using the proposed architecture while section VI presents a summary of the conclusions as well as avenues of future work.

II. Related Work

Most Internet of Things (IoT) architectures [9], [10] support basic features in which devices send data to a collector via the Internet or they allow services on the devices to be invoked remotely. The data can be temperature from sensors, health status from patients, or setup data for the devices to configure and apply according to the parameters. Pintus *et al*. [11] have proposed Service-Oriented Architecture (SOA) paradigms to build complex distributed systems via the composition of atomic, loosely-coupled software modules called services by using a Web Services Description Language (WSDL) standard.

The authors in [14] have proposed a middleware solution in which a device offers its capabilities as services. Our proposed architecture is based on this principle, but we extend it to provide more of a universal computing unit that is not limited to predefined services. It supports streaming services such as live-feeds from cameras. It also allows new services to be injected into devices remotely and scales the services according to their required load.

Architecture for IoT gateway centric in [12] has been proposed to have mobile clients and smart/non-smart devices to exchange real-time data. It has introduced new features, such as dynamic discovery of machine-to-machine (M2M) devices and endpoints, managing connections between non-smart devices, using Sensor Markup Language (SenML) to associate metadata with sensor and actuator measurements, and allowing mobile clients to control devices.

Readers interested in the subject may read a survey on IoT in [13]-[19] in which the authors have raised some major issues that research community has still been facing. The authors in [8] has addressed the complexity and cost of deployment of IoT by studying the applications of the RFID standard framework. The research has presented a blueprint to make the adoption of the standard less complex.

III. The Proposed Architecture

The strength of the proposed architecture is that, to the best of our knowledge, this is the first work in which a device (software or hardware) can be used as (i) data collector (messaging) such as sensors or actuators, (ii) storage device, something similar to bitTorrent Mainline Distributed Hash Table (DHT) [20], (iii) computational unit that allows any distributed software to run on. Our architecture supports multifaceted solutions. For example, A RFID device provides tag data, and it can utilize the messaging service. A refrigerator uses messaging service to download recipes. Moreover a smartphone can implement all three features to provide a complete solution. An established device in this architecture does not have to be a physical entity. It can be a computer instantiating many software instances and each acts as a device through the use of virtualization.

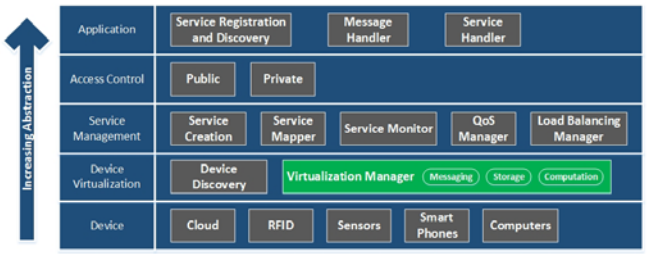

Fig. 2. The hecate architecture.

The architecture consists of five layers (components) as featured in Fig. 2, application, access control/security, service management, device virtualization, and device. Each of the layers contributes a necessary feature for the architecture.

A. Application

This layer provides an interface for applications to use the device's services. It consists of three main elements: service registration and discovery, message handler, and invocation handler.

Service registration and discovery allows a device to register its services or discover services provided by other devices. It also enables users to determine what services to register. This element registers services to public registries so that they can be searched by other devices. An example of a registry is Universal Description and Discovery and Integration (UDDI) to register web services. Using UDDI is a complex process and not suitable for dynamic and large services. An alternative would be using Device Profiles for Web Services (DPWS) for seamless communication between devices. The authors in [3] have proposed RSDPP for service discovery and provisioning process in conjunction with DPWS, a search query refine process which can be used for efficient discovery for this element.

The next element is Message Handler, which allows services to exchange messages. As with the implementation in [4], this element can implement WS-notification standard in which services act as producers and consumers to push and pull messages to/from other services. Producers provide a subscription interface and consumers provide a notification interface. SOAP can be used as a format for exchange messages.

Service handler is an element that invokes services by remote requests and maintains the connection between services. It takes messages from the message handler and passes them to other components of the architecture for further processing. It can act on behalf of the local services to request information from remote services.

B. Access Control

This component gives devices protection to share their services. A device can share some of its services to the public and have other services confined to a private group of users. Access control policies can be used to provide access to intended users and prevent malicious activities to the device. In the IoT, devices are mostly self-administrative in IoT. Also a security breach can be devastating as a breach can affect a significant number of devices. We utilize Web Service (WS) suite to address access control and security for our design. WS-policy [21], [22] would provide adequate security for communications between devices. Devices use WS-policy to describe security policy constraints. The WS-policy specification recognizes the following security attributes: privacy attributes, QoS selection, encoding considerations, security token requirements, and supported algorithms.

C. Service Management

Services need to be managed and monitored to maintain good and efficient operations. After a service is created, it will have been assigned categorical information such as name, identification, type, and description for search purposes. Depending on the types of service, they will have types of different Quality of Service (QoS). Fig. 3 shows the interactions between the elements within the service management layer. For example, a service is mapped to a virtualized object via the service mapper. Following the mapping, the QoS Manager receives information about a service's condition and quality via the service monitor. Before passing a request to the next layer, the load balancing manger works with the QoS manager and service mapper to decide whether a new virtualized object should be created or an established one should be removed.

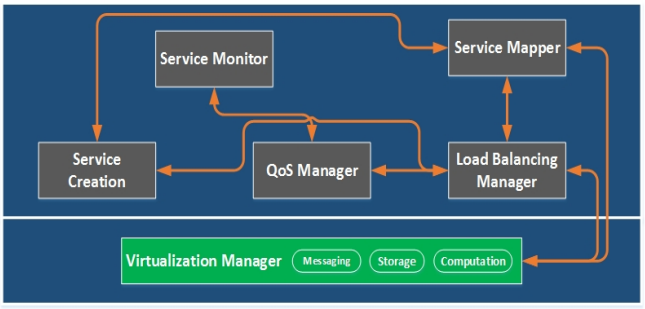

Fig. 3. Service management interactions.

Service creation creates services. After a service is created, the service will then be mapped to a virtualized object. Also, this element stores information about the services in a table named srv_tbl. The stored information will enable the services to be categorized and looked up. Each service has a data entry in the srv_table to store its information. It can be name, support functionalities, or capacity. Searching could be easier and more precise if there are more relevant information available. Information in this repository can be used by the application component to register services to a registry.

Service mapper links services to virtualized objects. A remote entity connecting to a service needs to go through the application layer, in which a service can be enabled or disabled by the device's user based on one's priority and preference. Another feature that the service mapper's supports is that a service can have many instances, for example, a computational unit can be confined to one remote entity. If the device capacity is large, it can have many instances of the computational unit to support many external entities simultaneously. The service mapper will then map each external entity to one instance.

Service monitor is an element that monitors the inbound and outbound traffic that the service processes rely on within a device. It has two main tasks: (i) sends traffic information to QoS manager and (ii) reports service status to respective entities. The first task allows the QoS manager to control the traffic based on the information that it provides. The second task allows it to initiate messages related to services. In the event of service failure, the service monitor can send messages to inform the remote entity.

QoS manager allows services to operate optimally over the grand scale of the Internet. The manager can adjust the QoS dynamically based on a given set of rules. Since the way traffic is processed and managed in the IoT could be much different from the traditional traffic, new requirements and support systems have to be defined and specified. A device of this architecture can have three types of service running concurrently: messaging, storage, and computation. The manager can set lower quality for computation than messaging to leverage the priority of the service. However when there are two or more services of the same type (e.g.,

messaging) then the manager needs to have some capacity to correctly prioritize the more important service.

Load balancing manager (LBM) controls the load for virtualized objects. When LBM detects too much load on a virtualized object it invokes the service creation and virtualization manager to create more services and virtualized objects. LMB can perform the following procedures: create and remove new services and virtualized objects, configure services and virtualized objects, and discover balancing issues with virtualized objects.

D. Device Virtualization

This component virtualizes devices and acts as an interpreter between devices and services so that a uniform interface is presented for exchanging messages and information. The virtualization process will be able to provide an easier way to implement the architecture and to integrate services. It also introduces heterogeneity for devices. Each device has its own specification and characteristics, it would not be practical to have an implementation of the architecture updated when a new device is plugged in. It hosts virtualized devices, which in turn provides hosting for services. This component helps discover devices when they are connected and creates virtualized objects for each of them. Depending on the workload requirement, a device can have more than one virtualized instances.

Device discovery is an element that allows plug-and-play functionality. Each device has its own capacity. There are two instances where a device may use this proposed architecture: (i) a device that has the capacity to implement this architecture (ii) a device that plugs into another device to be part of the network. For the first approach, the discovery feature is only needed to get identification and capacity. For the second approach, when a device is plugged into the host, which implements the architecture, the host will discover and extract information about the device in order to virtualize the device.

Virtualization manager is the main element of this architecture. There are many types of communication devices including RFID, sensors, computers, tablet, cloud computing, smartphones, and more. Each provides different functionalities. They can transmit just collected data (RFID, sensors, smartphones), share storage (computers, cloud computing) and computation (smartphones, computers, clouding computing). The virtualization process creates unified interfaces that allow any connected device to be part of the network. They would be able to interpret/decode and encode messages from/to remote services without loss of meaning. We separate device capabilities into the three main types that they can offer: messaging, storage, and computation. Each type provides a different interface. It is not necessary that a device can offer all three types simultaneously.

E. Messaging

This type of capabilities allows services to send messages and stream contents (video streams). An RFID device can send collected information about temperature, locations, or medical data. A smartphone can stream video from the camera to remote location. The interface for this capability has similar specifications as in [4], [9].

F. Storage

This is a dedicated feature that is only used for storage. Since the size of Internet is large, the logical way of utilizing the storage of devices is by implementing a form of bitTorrent DHT. Devices provide information about their storage after coming online, based on this information a virtual object will be created and the size of the storage can be set from the application component. This storage objects can have many instances based on the access control specified by the users. A public object can be used by Internet users and a private object can be used among a user's devices.

G. Computation

One of the distinctions in our architecture is that a device in this architecture can contribute the surplus computational power to the outside world. Our purpose of computational unit in this architecture is to allow connected devices to be able to act as a neuron(s) that can work with other devices in a distributed manner. Each device receives computational tasks which they execute regardless of the underlining hardware architecture. In order to facilitate this functionality, the computational unit has to support one common programming language and a set of common features. A user can send a task out and it will get distributed to the devices that are open for computational contribution. Depending on how a task is coded, it can replicate itself into multiple instances on the same local machine or communicate with other devices, which in turn perform the replication for the vast of the Internet. One important note is that this computational unit can execute any code programmed for it unlike a web service that executes only a predefined set of data.

It uses similar principles as in distributed computing, i.e., communicating and coordinating tasks by passing messages with the following characteristics: concurrency of components, lack of a global clock, and independent failures of components [23]. Because devices are virtualized, a virtualized object can have many instances. Each of these instances runs in memory in a protected location in order to prevent any unintended actions. In this architecture, an instance can only host one computational task. If there are more tasks requested, more instances could be instantiated. This approach is pursued to protect one computation task from another. Even though computation is shared to the outside world, local computation always has higher priority than the shared ones. Moreover, at any time a user can disable an instance.

IV. Features Comparison

We compare our proposal with the current architectures in literature. The characteristics of IoT devices as specified in [24] are heterogeneity, automation, load balancing, dynamicity, zero-configuration, messaging, storage, computation. Heterogeneity accepts dissimilar devices to be in the same network. Automation allows sending and receiving data automatically. Load balancing gives IoT devices the ability to adapt. Dynamicity gives the devices the freedom to move around and still maintain the service. Zero-configurations supports a simple integration with the

IoT. There are other characteristics that IoT devices should have to provide a complete solution such as storage and computation capabilities. In Table I the proposed work is compared with DIAT [9], SOCRADES [4], and IoT-A [25].

TABLE I: FEATURES COMPARISON

Features	*HECATE*	*DIAT*	*SOCRADES*	*IoT-A*
Heterogeneity	✓	✓	✓	✓
Automation	✓	✓	✓	✓
Load Balancing	✓			
Dynamicity	✓	✓	✓	✓
Zero-configuration	✓	✓		✓
Messaging	✓	✓	✓	✓
Storage	✓			
Computation	✓			

V. USABILITY

With unimaginable projected earning, the business community is actively exploring the IoT. According to [1], by 2016 a great many of these businesses expect to be able to use it. Currently, more mobile devices equipped with cameras, accelerometers, Gyroscopes, GPS, and microphones are sharing their content on the cloud. Another example of the existing IoT evolution are home computers that can be accessed over the Internet from a tablet to control appliances. The next technological phase will be when this heterogeneous network of things becomes one humongous network of objects and products. In 2020 it is projected that Web 3.0 will be a new service-oriented approach based on the Internet of Services and Internet of Things [25]. The main obstacles preventing the next technological phase from developing are the immaturity of the services and standards [1].

Working to build more sophisticated services and standards, our vision of a future IoT architecture is based on the ability to globally commission virtual and physical resources, e.g., share messaging, storage, and computation capability across the IoT. To do this Hecate will focus on creating generic tools and programming interfaces for service development in a humongous network. To benefit both object applications and people, some applications of Hecate include: BitTorrent, a peer-to-peer file sharing protocol that by utilizing shared storage large volumes of data will be distributed across the IoT; security, encryption algorithms which are computationally extensive could be much more widely used than they are today; messaging, once a new device is plugged into an established device in the IoT, Hecate's messaging service would become available to the new device; and Grid Computing, a distributed computing technology would be greatly enhanced by this framework. Anticipating the development of unique requirements intrinsic to this domain, using Hecate as a foundation will enable other new services to rapidly evolve.

VI. CONCLUSION

We have proposed a vision of an architecture that allows IoT devices to send data and share resources. The architecture is created to provide three important features: messaging, storage, and computation. We have incorporated service oriented architecture approach for ease of use and implementation. Our proposal meets all the characteristics and features of a standard IoT architecture and is able to provide more functionalities such as storage and computation. For future work, we will implement the architecture in our test-bed and in a real-world environment to evaluate the feasibility and effectiveness of our design.

REFERENCES

[1] K. Pretz, "Smarter sensors," *The Institute*, vol. 38, no. 2, pp. 6-7, March 2014.

[2] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless embedded Internet*, 1st ed.; John Wiley & Sons Ltd.: Chichester, UK, 2009.

[3] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based Internet of Things: Discovery, Query, Selection, and on-demand provisioning of web Services," *IEEE Transactions on Services Computing*, vol. 3, pp. 223-235, 2010.

[4] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza, and V. Trifa, "SOA-based integration of the internet of things in enterprise services," in *Proc. IEEE International Conference on the Web Services*, July 2009, pp. 968-975.

[5] M. M. Komarov and M. D. Nemova, "Emerging of new service-oriented approach based on the Internet of Services and Internet of Things, e-Business Engineering (ICEBE)," in *Proc. the International Conference on IEEE 10th*, Sept. 2013, pp. 429-434.

[6] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *Proc. the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, vol. 31, no. 2, April 2001, pp. 20-41.

[7] I. F. Akyildiz, W. Su, Y. Sankasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey computer networks," *Communications IEEE*, vol. 40, no. 8, pp. 102-114, August 2002.

[8] D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, REST and mashups to simplify RFID application development and deployment," in *Proc. the Second International Workshop on Web of Things*, ACM, June 2011, p. 9.

[9] C. Sarkar, Nambi, R.V. Prasad, and A. Rahim, "A scalable distributed architecture towards unifying IoT applications," *IEEE World Forum on Internet of Things (WF-IoT)*, March 2014.

[10] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet: the Internet of Things architecture, possible applications and key challenges," in *Proc. the 2012 10th International Conference on Frontiers of Information Technology, IEEE Computer Society*, December 2012, pp. 257-260.

[11] A. Pintus, D. Carboni, A. Piras, and A. Giordano, *Connecting Smart things through Web Services Orchestrations*, Springer Berlin Heidelberg, vol. 6385, pp. 431-441, 2010.

[12] S. K. Datta, C. Bonnet, and N. Nikaein, "An IoT gateway centric architecture to provide novel M2M services," *IEEE World Forum on Internet of Things (WF-IoT)*, pp. 514-519, March 2014.

[13] L. Atzori, A. Iera, and G. Morabito, "The social internet of things (SIoT) –when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, 2012, pp. 3594-3608.

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

[15] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, Sept. 2013.

[16] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the internet of things," *Internet Computing*, vol. 14, no.1, pp. 44-51, 2010.

[17] D. Bandyopadhyay and J. Sen, "Internet of things: applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49-69, 2011.

[18] S. Turber, J. Brocke, O. Gassmann, and E. Fleisch, "Designing business models in the era of internet of things," *Advancing the impact of Design Science: Moving from Theory to Practice*, vol. 8463, 2014.

[19] J. Arias and Y. Barajas, "Wireless sensor system according to the concept of IoT –internet of things," *International Journal of Advanced Computer Science and Information Technology*, vol. 3, no. 3, 2014
[20] Mainline DHT Specification. [Online]. Available: http://www.bittorrent.org/beps/bep_0005.html
[21] WS-Policy. [Online]. Available: http://www.w3.org/TR/ws-policy/
[22] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila, "Two-phase authentication protocol for wireless sensor networks in distributed IoT applications," in *Proc. IEEE 14th International Conference on Wireless Communications and Networking (WCNC)*, pp. 2770-2775, April 2014.
[23] F. Coulouris, J Dollimore, T. Kindberg, and G. Blair, "Distributed systems: concepts and design," 5th ed. Perason Education, 2012.
[24] G. M. Lee and J. Y. Kim, "The Internet of Things: A problem statement," *Information and Communication Technology Convergence (ICTC)*, pp. 517–518, 2010.
[25] European Lighthouse Integrated Project. [Online]. Available: IoT-A. http://www.iot-a.eu/

Van Nguyen is an assistant professor of Computer Science at Saint Leo University, Florida. He obtained his Ph.D. from the Center for Advanced Computer, Studies at the University of Louisiana at Lafayette, where he participated in a multi-million dollar project (UCoMS) funded by the Department of Energy. His research areas are in computer wireless networks, performance evaluation, protocol design, and data mining. He also specializes in computer security and computer management. His vision of a future computer world is that there would be one massive computer consisting of all the computing devices in the world that can solve any problem presented to it.

Audrey A. Gendreau is assistant professor of Computer Science at Saint Leo University, Florida. She recently completed her Ph.D. in Computer Information Systems with a concentration in Information Security at Nova Southeastern University. She has obtained the NSA certification in graduate work, the Global Information Assurance Certification Forensic Examiner (GCFE), and is Java certified. Her main research topic of interest is in the area of the security of wireless sensor networks (WSN) and especially as they pertain to the Internet of the Things (IoT). When she is not working on her research on sensor networks and the IoT, she enjoys volunteering to help other computing organizations such as participating in the Anita Borg Institute Grace Hopper – Celebration
Scholarship as a grant reviewer.