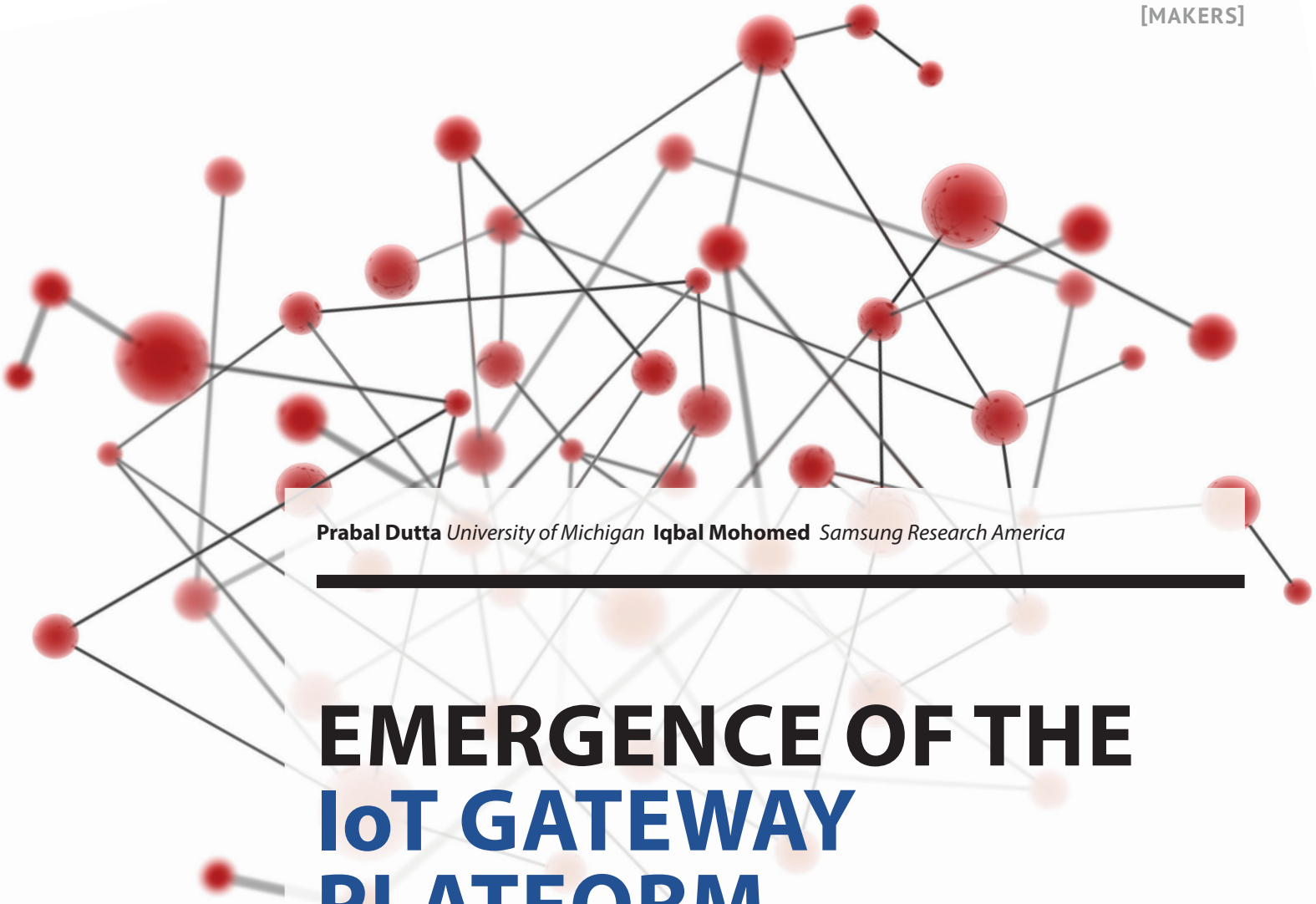

Prabal Dutta *University of Michigan* **Iqbal Mohomed** *Samsung Research America*

EMERGENCE OF THE IoT GATEWAY PLATFORM

CONNECTING ALL THE THINGS

Many believe that the Internet of Things (IoT) represents a future in which trillions of devices will be connected to the Internet within the next decade or two. Today's IoT devices are connected wirelessly in three main ways: (i) tethering to Bluetooth Smart (or "BLE") capable smart-phones via device-specific apps, (ii) connecting through app-specific hardware gateways over a variety of different radio interfaces, and (iii) using shared Wi-Fi access points located in homes and offices. Smartphones, custom gateways, and access points have thus far provided adequate connectivity for current IoT devices, but the limitations of this approach are becoming clear.

Today, users must install a new app on their smartphones for each new device with which they intend to interact. This is akin to requiring one to install a new web browser for each distinct website that one visits. While this may have been tolerable when the web was in its infancy, and the number of websites could be counted on one hand, the approach would be considered seriously broken as the number of sites increased even slightly. Yet, users are willing to tolerate exactly this situation today with IoT devices. The difference is that the notion of casually browsing the physical world is an idea whose time has not yet come but is technologically ripe.

Many IoT devices are bundled with their own proprietary gateway that connects the device to the Internet. The Kevo door lock from Kwikset comes with its own gateway. So does the August lock, Hue lights, Sonos sound system, and a myriad of other IoT devices. What's unfolding today is no different than requiring a unique access point for each website one visits. Once again, while this approach may be tolerable with just a few devices, it is untenable with the expected proliferation of IoT devices.

Ultimately, phones and gateways provide three key functions for an IoT device: (i) a user interface, (ii) network access, and (iii) in-network processing. Currently, these functions are packaged within mobile apps and custom gateways but there is no reason that they cannot be provided generically to all applications in way that requires neither custom smartphone apps nor proprietary gateways. Indeed,

Illustration, istockphoto.com

such an approach would allow a much more fluid interaction of phones and devices, by obviating the need to download myriad apps, and also keep the shoe closet from becoming a tangled mess more reminiscent of the wiring closet.

Wi-Fi connectivity is one way to avoid these problems, but it too is not enough. Many imagine a world in which their IoT devices are marshaled into an application that spans several of them and executes an application across the device ensemble. Unfortunately, deploying such applications today requires low-level programming with software like The Thing System [1] or inter-cloud interactions like the Kevo-triggered Nest control mediated by UniKey's cloud service. The former is difficult to use and latter fails whenever Internet connectivity is lost (and leaks private data to the cloud).

Instead of this patchwork quilt of largely incompatible smartphone apps, stove-piped proprietary gateways, and unnecessary cloud mediation, imagine if there were a principled, open standards-based approach to connecting all the things, both to the Internet, and to each other. Indeed, such things are now on the horizon.

MOBILE GATEWAYS

Today's IoT smartphone apps conflate a user interface, network access, and in-network processing, integrating these functions in a single, monolithic piece of software. This has two major drawbacks. First, every new device requires an app install, which makes *casual interactions* between phones and devices impossible. Second, limited operating system support for cross-application communications makes *device-to-device data sharing* and orchestration difficult to achieve, even when mobile apps have been pre-installed. Some recent efforts, including iBeacon, UriBeacon, and EddyStone have sought to address aspects of this problem.

Apple's iBeacon allows BLE beacons to advertise specially formed packets that are received by nearby smartphones and, potentially, trigger actions on the phones, based on the data contained in the beacon. An iBeacon payload contains three key fields: a 16-byte UUID, a 2-byte Major identifier, and a 2-byte Minor identifier. In one usage model, a retailer generates its own UUID, assigns a unique Major id to each of its retail locations, and a Minor id to each distinct area within a location. The retailer would also release a smartphone app that it would encourage its customers to download. This app, run in background mode, would register with the phone's OS to receive only those beacons that contain a specific UUID. The result is a retailer's smart phone app will receive beacons that identify the phone's approximate location within a store, enabling context-specific advertising using lock screen notifications.

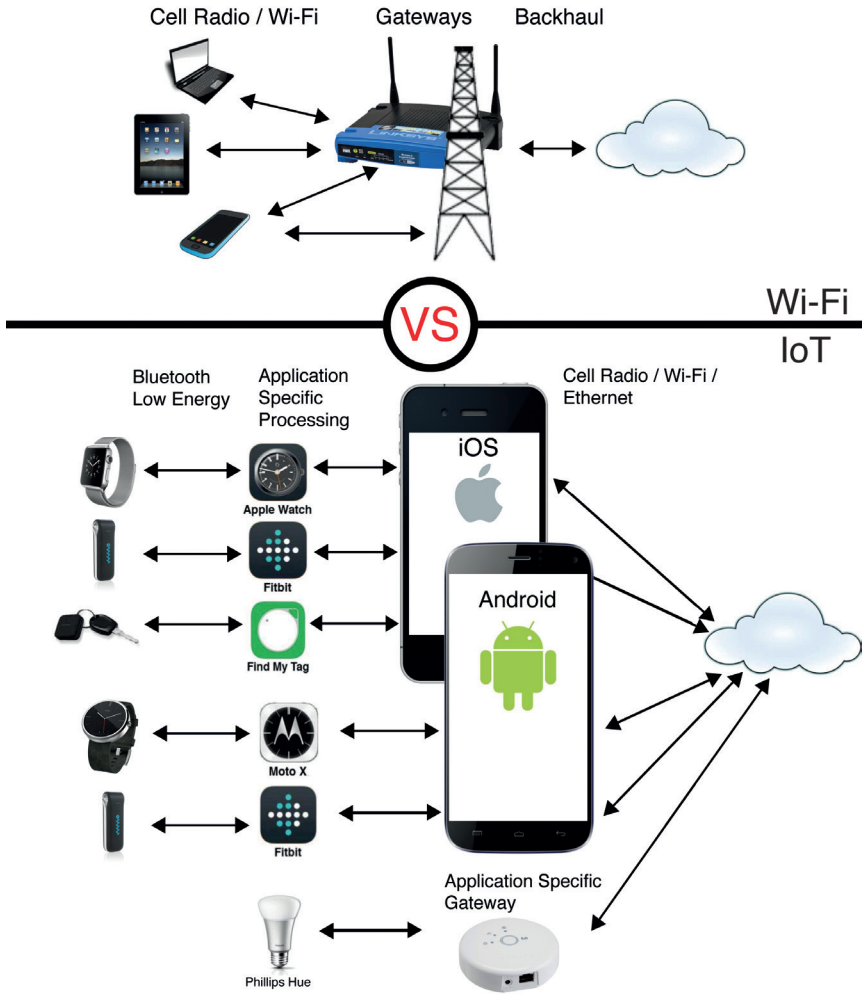

FIGURE 1. The stove-piped nature of today's IoT: Unlike mobile, tablet, and phone platforms that all use the same network layer gateways, namely cellular and WiFi, today's IoT devices use a myriad different application specific gateways that exist as device-specific smartphone apps or custom gateway hardware. This situation detracts from the user experience, results in unnecessary hardware deployment, hampers orchestration across multiple devices and creates new failure modes.

iBeacon technology is focused on enabling retail micro-location services. The iBeacon model assumes that customers are willing to install retailer-specific apps – one for each retail chain or independent store – on their phones to receive the beacons advertised by only that retailer. These restrictions make using iBeacon in more general settings difficult. First, we would expect many BLE devices offered by many different vendors that all use different UUIDs to often be collocated in home and office settings for applications beyond micro location services. Second, neither a single UUID nor a single app for all such devices would be viable, and indeed would be antithetical to the iBeacon model.

Google's UriBeacon enables BLE beacons to advertise URNs or URLs that a nearby smartphone can receive and follow

to obtain additional data about the beacon or, more likely, the environment in which the beacon is situated. Used in this way, an UriBeacon provides a cyber-physical hyperlink, enabling broadcast beacons to contain an index to information or services relevant to the current physical context.

EddyStone, the successor to UriBeacon, is used in Google's Physical Web Project to provide additional functionality supporting ambient device interactions. In particular, an EddyStone-enabled app or browser can navigate to an advertised web page, providing peripherals devices with rich user interfaces served from the cloud, but not interactively with the device itself. A drawback to UriBeacon and EddyStone technologies, ostensibly by design, is that the beacons themselves transmit limited data – typically the URL/URN – but do not appear to support direct interactions between the smartphone and beacon. Rather, interactions are indirect, with the smartphone interacting with the cloud, and the beacon left to find an independent path to upload its dynamic content to the cloud.

But this design point may not be needed. Dynamic but encrypted content from the beacon could be transferred through a smartphone to the cloud (to be stored for later use), and then unencrypted with a cloud-provided key and rendered in the smartphone once cloud delivery finishes. This creates a symbiotic relationship: the phone gets dynamic data and the beacon gets an infrastructure-free cloud pathway.

This approach eliminates the need to provision a data backhaul for dynamic beacons – a costly and painful step in deploying persistently connected end devices today – and opens the door to richer and more dynamic interactions. Of course, in other cases, it may not be necessary to encrypt data at all. For example, smartphone-hosted apps could be allowed direct interaction with a BLE beacon if the phone can prove close proximity, perhaps using an optical or radio-based challenge-response handshake protocol. Such proximal access would mirror the kind of access patterns observed in physical environments – proximity to a light switch, for example, affords one the opportunity to manipulate it while higher-order social mechanisms exists to resolve contention from concurrent accesses from multiple users.

The key to making such designs work revolves around new networking services and incentives. Smartphone-based transport services with strongly verifiable delivery semantics are needed. Moreover, these services must be broadly available in smartphones to be generally useful. And there's no reason to limit these services to just smartphones – they may be just as useful in immobile gateways.

IMMOBILE GATEWAYS

The widespread adoption of BLE-enabled smartphones has fueled the consumer IoT space. The phone provided two key, but missing, pieces – a "last inch" network and a "remote" user interface – accessible easily from a nearby purse or pocket. With phone in hand, accessing embedded and wearable IoT devices became possible for the masses and ushered in consumer experimentation with IoT by technology early adopters. But, some of these IoT devices clearly had value even in the absence of a nearby smartphone to act as its display and network. A great example is the BLE-enabled door lock that opens when one comes home by communicating with one's smartphone. But what if one wanted to unlock the front door when an out-of-town visitor called to say that she had arrived early? Such scenarios require persistent, rather than intermittent, access to the wide area network, underscoring the need for an immobile, always-on gateway.

Lacking a standardized, widely deployed network for BLE and various flavors of ZigBee and 6LoWPAN over 802.15.4, IoT device vendors – from Phillips to Sonos to Kevo – have resorted to shipping their own gateway hardware, turning shoe closets into wiring closets. Recognizing the chaos of this approach, a number of recent efforts have sought to converge on smaller, more capable gateways. Home Depot & Wink's Hub, Lowe's Iris Hub, Samsung's SmartThings Hub, and Staples Connect are all examples of commercial gateways that support a variety of radio protocols including a mix of WiFi, ZigBee, Z-Wave, and Lutron, among others. Unfortunately, many of these systems are essentially closed, to varying degrees, that at best provide a small amount of programmatic access and at worst constrain hackers, makers, and users to the set of devices and software hand-selected by the vendors.

In contrast to the standalone hub model, there appear to be efforts underway to integrate the hub/gateway functionality into devices that offer some other utility. For example, Nest acquired Revolv – a startup company focused on building a general-purpose, multi-protocol gateway – with the likely goal of integrating the basic gateway functionality into their own products that have already been deployed (e.g. Nest Thermostat), are now emerging on the market (e.g. Google's OnHub), or will soon emerge, presumably with support for emerging industry standards like those being defined by the Thread Group and Bluetooth SIG for standardizing IP-based access to IoT devices.

Devices like Nest's Learning Thermostat, Fantem's Oomi Cam and Amazon's Echo use a home's existing Wi-Fi to gain network access. What's interesting about some of these devices, like the Nest Thermostat or the Oomi Cam, is they include additional radios. Once a dual- or multi-radio device connects to a home's Wi-Fi network, it can then be a gateway for other devices – like BLE, 802.15.4, or Z-Wave devices that cannot connect to a home's Wi-Fi network directly – with nothing more than a remote software upgrade. With IPv6 adaptation over 802.15.4 and, recently, over BLE links using 6LoWPAN, it may soon be possible for embedded devices to both gain wide-area access for the kind of scenarios envisioned above and for these devices to interact with each other over the local network, enabling the Intranet of Things.

The multi-radio IoT gateway/hub is a new component whose role is being explored and defined largely in the commercial sector with limited opportunities for academic tinkering on the platforms, protocols, security services, orchestration model, or user experience – at least for the products currently available on the market. One reason vendors seem to be "rolling their own" is due to the lack of established standards (and in some cases, too many competing standards for the same thing). Furthermore, security and privacy concerns have caused local communication between devices and gateways to often be encrypted, making reverse engineering all the more challenging. Unfortunately, history has shown that this is likely to lead to poor design choices, resulting in incomplete

protocols, security vulnerabilities, and user frustration.

In contrast with vendors focused on either stove-piped gateways that are a component of a vertically integrated IoT system, like Kevo's BLE door locks and gateway, or stand-alone/device-integrated gateways, like Revolv/Nest Thermostat, that may soon support "standardized" protocols like Thread, others vendors including Intel are offering reference gateway platforms that can be customized by manufacturers. Intel's IoT gateway software stack includes support for connectivity, manageability, security and execution, making it easier for third parties to build custom IoT gateways on the base platform. However, most of the upper layer protocols that could be used to enable generic, multi-radio, multi-device gateways are not provided.

ENSEMBLE ORCHESTRATION

Although many IoT systems are standalone today, much of the expected value of the IoT depends on orchestrating a wide range of different devices into new applications. Unfortunately, orchestration today is a tangle of competing architectures and ecosystems. One model involves cloud-to-cloud interactions between vendor clouds. This is the approach taken today to link a Nest Thermostat and a Kevo door lock, for example. While it may be easier to structure cloud-to-cloud interactions between a few clouds and a small number of devices, the approach scales poorly since the number of possible interactions grows quadratically with the number of devices. Reliability also suffers due to the dependence on multiple remote clouds over frequently unreliable access networks, rather than local computing resources.

Two other approaches may be better suited to reliable, distributed orchestration. The first of these involves standardizing protocols within an ecosystem and encouraging vendors to build compliant devices, which of course limits the choices to the supported devices in the ecosystem, but ensures compatible devices. This is the approach that AllJoyn, ZigBee, Insteon, and EnOcean took. The second of these models involves adapters to devices to support programmatic or rules-driven orchestration, which is the most flexible but requires hardware dedicated to hosting the adapters and applications, as many devices may be incompatible with one another. This is the approach that Revolv, SmartThings, and The Thing System took, requiring their own gateway or hub to provide the point of orchestration.

Apple and Google are starting to build out their own ecosystems targeted at the home. Apple's HomeKit is taking a two-pronged approach to integrating devices: certified devices that have been tested (and licensed) and bridged devices that are limited to non-actuation roles. In this way, Apple retains considerable control over the HomeKit ecosystem and the device vendors who can participate in it. By most accounts today, it appears that unincorporated organizations or academic researchers will face difficulty is fully participating in the Apple HomeKit ecosystem. While it may be difficult for researchers to integrate their own devices into HomeKit, it appears likely that the research community will be able to write and deploy apps that target the HomeKit model, where devices are given names and natural language-based automation will allow orchestration of these devices.

Google/Nest is taking a slightly more open route, working on standardizing a low-level set of networking protocols to connect devices, using Thread, but planning likely open but still proprietary protocols (Weave from Nest) and programming frameworks (Weave from Google) to offer localized orchestration between devices without the need for an Internet connection. Nest is exposing a rather limited subset of the functions of their devices to other devices using Weave, but it remains to be seen how much of the internal device functionality will be eventually exposed to third party devices and services. By keeping some key functions closed, Nest ensures that their devices cannot inadvertently or intentionally misused to take control of a home's actuation points nor can the devices be subsumed into third party ecosystems, at least not as fully functioning devices.

APP STORE FOR THE GATEWAY

As a growing number of devices, gateways, and frameworks gain market traction, it seems plausible that the gateway will support various programmable functions. One way of supporting such functionality could be an app store for the gateway – a sandboxed execution environment that can download and host apps which use data from sensors, control actuators, and orchestrate a variety of interactions between the devices, cloud, and users. This would allow users to download and deploy a variety of different applications.

One can imagine linking smartphones and the immobile gateways into one "virtual" gateway that spans multiple gateways and executes applications across all of them. The gateways would then serve the dual roles of network layer gateways, providing network access to the devices, and application layer gateways, placing the application logic at the nearest or most appropriate gateway.

CLOSING THOUGHTS

According to a recent Gartner report [3], IoT is currently near the peak of inflated expectations. To get from disillusionment to enlightenment, many problems will need to be solved. How do we secure these systems and maintain privacy, while still allowing openness and interoperability? There are also many open questions with respect to how consumers will actually interact with IoT systems. Will we live in a world where expert developers will create general-purpose applications (the predominant model for smartphone applications), will end users become programmers or will the smart home automatically learn user preferences and desires. Recent advances in speech recognition and natural language processing present great opportunity for influencing how IoT will develop and ultimately be used at home and in the workplace. Will traditional web-scale infrastructure be sufficient at IoT scale, or will new a paradigm shift occur in the backend to tackle efficient data processing in aggregates, soft real-time interactivity and user mobility? The possibilities afforded by the IoT vision are very exciting, and we are convinced that this will be an important and growing area of research in the years to come. ■

Disclaimer: All views and opinions expressed in this article are solely those of the authors and not of their employers.

REFERENCES

[1] http://thethingsystem.com/dev
[2] https://developer.apple.com/ibeacon/
[3] http://www.gartner.com/newsroom/id/3114217