2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing

A Mobile Phone based WSN Infrastructure for IoT over Future Internet Architecture[1]

Jun Li, Yanyong Zhang, Yih-Farn Chen[§], Kiran Nagaraja, Sugang Li and Dipankar Raychaudhuri
WINLAB, Rutgers University, New Brunswick, NJ, USA, [§]AT&T Labs - Research, Florham Park, NJ, USA

***Abstract* — Large-scale wireless sensor network (WSN) deployment is a major challenge for Internet of Things (IoT) to connect physical world through sensors or tags at the scale of billions of objects. Today's WSNs normally use dedicated gateways to bridge sensors and IP networks. The installation and maintenance of such a WSN infrastructure are expensive and non-scalable. As the number of smart phone users grows exponentially every year, a powerful mobile computing platform could be used as an alternative WSN infrastructure, which is ubiquitous and scalable. This paper studies the major challenges of using mobile phones as spontaneous gateways of WSNs in IoT systems. The challenges include (1) matching the throughput of mobile phone gateways and sensor data rate at hotspot locations, (2) securing sensor data access and (3) providing accounting records of gateway the service offered by mobile phones. In this paper, we show that using a name-based future Internet architecture (FIA), such as MobilityFirst, these challenges can be overcome effectively though its native networking layer functions. A proof-of-concept prototype system demonstrates the feasibility of proposed solution. The system delivers a temperature sensor data from an Android phone directly to multiple applications via in-network multicast over an MobilityFirst network testbed.**

Keywords- M2M communications, wireless sensor networks, Internet of things, future Internet architecture

I. INTRODUCTION

The vision of Internet of Things (IoT) is to connect billions of physical objects (Things) and make them available to Internet applications[1]. For many years, machine-to-machine (M2M) communication technology [2,3] plays an important role to connect real world objects to Internet applications. The difference between IoT and M2M is that IoT aims an open access platform independent of applications while M2M offers a closed system designed for vertical market applications. The goal of IoT is to offer ubiquitous accesses of the status of Things, represented by sensor data, for Internet applications.

The first challenge of building an IoT platform with open standards is to establish a large-scale wireless sensor network (WSN) infrastructure independent of specific applications. The low-power requirement of sensors prohibits them from directly connecting to IP networks. Therefore, a gateway that can support both sensor and IP network protocols is required to bridge a WSN and the Internet. In traditional M2M system, every WSN must have at least one dedicated gateway. The gateways are usually owned by the application owners and are not shared across applications.

Deploying a WSN gateway is particularly difficult in public areas, similar to deploying WiFi APs. It requires accesses of a physical location, a power source and an Internet backhaul connection. Although the traffic load of a WSN can be much lower than a WiFi network, the density of gateway deployment cannot be lower because of the low transmission power of sensors. The overall cost of WSN infrastructure is a major hindrance for IoT system deployment, especially in an outdoor environment.

A good example of a large WSN deployment can be seen in the following use case:

As a part of a smart city initiative, NYC municipal government starts an environmental monitoring project. The project objective is to monitor temperature ubiquitously throughout the city. The measurement must be able to identify every heat and cool hotspots on streets, in parks and other public areas. This requires a city-wide, high density wireless sensors deployment. If every sensor uses cellular (GPRS/3G/4G), the cost of sensors and life on battery will prohibit a large volume deployment. A large number WSN gateways are needed to build a WSN infrastructure.

The ever-growing mobile phones now are forming a ubiquitous computing platform, penetrating to every corner of our society. Wherever there is a need for connecting physical world objects (Things), it is most likely covered by some mobile users' daily activities. According to studies in [4], mobile users contribute 15% of their time to places other than their home, school or work, which implies a huge amount of mobile user time to cover the majority of public places where installing and maintaining WSN infrastructure could be expensive.

This paper studies major issues of using mobile phones as WSN gateways for IoT systems. The first issue is how to keep the uniqueness of sensor identities when their data are collected by random mobile phones. The second issue is to determine how many mobile phones and how much of their time are required at a given location. The third issue is how to keep the secure access of sensor data, which includes data integrity and access control. A secure access is also essential for IoT operators to establish an economic model for this WSN infrastructure, which includes a proper accounting mechanism for the service offered by spontaneous gateways and sensor data received by IoT applications.

[1] Research Supported by NSF Future Internet Architecture (FIA) program under grant #CNS- 1040735

This paper is organized as follows. Section II presents the M2M/IoT system architecture with different types of WSN infrastructures. Section III proposes our mobile phone-based WSN solution. The solution relies on a name-based future Internet architecture (FIA) and MobilityFirst is used as an example. Section IV discusses the capacity issue of mobile phone-based WSNs. Section V discusses the security issues including data integrity and access control; it also talks about a possible incentive model for mobile gateways. Finally, we present a prototype system that implements a mobile WSN gateway on an Android phone, which has interfaces to temperature sensors and MobilityFirst FIA. The prototype demonstrates sensor data is collected by the mobile gateway, and sent to multiple applications via in-network multicasting of a MobilityFirst network testbed.

II. M2M/IoT System Architecture

An abstract M2M/IoT system architecture can be shown in Figure 1. Normally an M2M system consists of

- a wireless sensor network (WSN) that connects physical Things with sensors, tags or actuators
- an M2M server that hosts sensor data and
- an end-user application accessing sensor data over Internet

In this paper, without loss of generality, we use "sensor" to represent all kinds of sensors, tags or actuators; and use "sensor data" for data from sensors, tags and data to actuators. For applications using actuators, the data flow is from application to M2M server and to actuators through WSN.

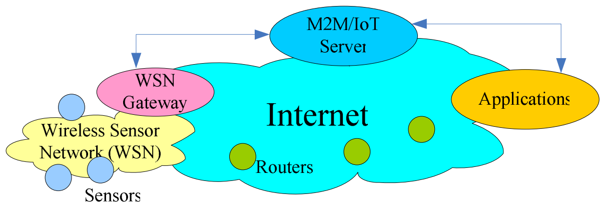

Figure 1. M2M/IoT System Architecture

The vision of Internet of Things (IoT) is to bring billions of physical world objects to the Internet and make their data available to Internet applications. The difference of an IoT application from a conventional Internet application is the need of wireless sensor networks (WSNs) to connect physical world objects.

There are three types of WSN infrastructure. First is to build a dedicated wireless architecture exclusively for M2M/IoT applications, On-Ramp [5] and SigFox [6] are typical examples. On-Ramp is a San Diego startup who deploys a wide area wireless sensor network for smartGrid. Sensor data on smart meters are directly collected by base stations on cell tower. They claim one base station can cover the whole San Diego city. SigFox is a French company who deploys a wide area cellular network dedicated for M2M, however, it uses a gateway between sensors and cellular network. They claim low cost cellular deployment because the cell size can be very large. SigFox claims 1:1000 ratio to conventional cell size and has deployed their network, which already covered 80% of France. Both companies use a proprietary RF technology with features of low data rate and high receiver sensitivity. This type of WSN architectures has a high entry barrier. Until they are widely adopted in a large scale, the service cost per sensor could be very high. It could be a good platform for well-defined vertical markets, for examples, oil field and industrial complex where normal mobile users don't reach frequently. But it may not be a good choice for consumer markets in highly populated public areas or in private homes, because Internet backhaul coverage is already pervasive, additional wide area network coverage is redundant.

Second type of WSN is a two-tier WSN architecture, consisting of a local WSN and an existing Internet backhaul. The WSN gateway can bridge the low power PHY/Link layers designed for sensors and WiFi or Ethernet connected to Internet backhaul. This is the most practical and economic way to implement a WSN infrastructure. Most of M2M systems as of today use this type of WSN architecture. The oneM2M [7], a standard partnership including ETSI, TIA, ATIS, CCSA etc. as member organizations, promotes this WSN architecture to use existing Internet infrastructure.

As a variation to the second type, the third type of WSN architecture uses mobile terminals as the gateways for WSNs. Mobile gateways reduce not only the infrastructure cost but also sensor cost due to the lower average energy used on sensors [8]. In the past, the applications are using enterprise built special mobile terminals as gateways that are trusted and operated within a closed system for the enterprise. A handheld wireless RFID reader is a typical example of mobile gateway of a wireless sensor network.

III. Mobile Phone based WSN Infrastructure over Future Internet Architecture

Using mobile terminals as WSN gateways is not a new concept. There are many enterprise M2M applications using mobile gateways. However, the gateways are proprietary devices with hardware and/or software specific to a given application, where, sensor data collected by mobile gateways is directly feed to the application or a dedicated server in the enterprise network.

An application independent mobile phone-based WSN infrastructure can be implemented by wireless mobile network operators (MNOs). They can offer an M2M/IoT service platform as shown in Figure 1, which includes wireless sensor networks and an M2M/IoT server. The M2M/IoT server can host sensor data collected by WSN gateways and make them ready for subscribing applications to access. Instead of building an expensive WSN infrastructure by only fixed gateways to cover potential sensor fields, a mobile network operator can use the mobile phones of their customers as spontaneous WSN gateways to collect sensor data.

In an enterprise M2M solution, a mobile gateway is responsible to deliver the data to a dedicated server over Internet. An end-to-end connection between mobile phone and the server must be established for every data upload. This can impose a large load to mobile phones for very short sensor data due to connection setup latency. The end-to-end connection is also required for the accounting records on sensor data uploading for each mobile phone. Therefore, the main cost to mobile gateways is not sensor data collection but the delivery of the data, because later uses more energy and time of the gateways. Even if the bandwidth is subsidized by the wireless operator, the battery drain may become a major hindrance for mobile users to participate in this program.

In this paper, we present a new way to build WSN infrastructure assuming Future Internet Architecture (FIA) is used.

A major trend of FIA development is Information Centric Networking (ICN) [9] that assumes a name-based networking layer. The name of a networked object (also called an information object) is separated from its network address / locator. The data packets from / to a networked object can be identified by the name of the object, regardless of where it is located.

The naming scheme is a key differentiator of different ICN proposals. Named Data Networks (NDN) [10], one NSF FIA proposal, uses URIs as the names of networked objects. The networking layer of NDN can route any data packet named by URI. It proposes a mechanism to build routing table entries by memorizing the request traces for the URI. A request to a URI named data can either be flood in a local area network or routed based on URI hierarchy in a wide area network. Using URI as names of networked objects is intuitive. The advantages are embedded hierarchy and semantics. For examples, from a URI, router is able to aggregate URI with longest prefix matching; and applications are able to apply semantic based policies for searching, filtering and aggregating purposes. The biggest disadvantage of using URI is insecure because URI is clear text and can be spoofed. A separate self-verifiable identity, like a public key and a certificate bundle the key to the URI name, must be used to avoid spoofing. Another disadvantage is the variable length of names based on URIs can slow down the routing performance significantly compared to fixed length names.

Security and router performance are major considerations for us to choose another naming scheme. In our FIA proposal, MobilityFirst [11], self-verifiable, fixed length names are used for networked objects. The object name itself contains a public key of the object owner. The public key based name is a globally unique identifier (GUID). Only owner can claim the integrity of the data from a network object identified by such a GUID. In other words, the name cannot be spoofed by non-owners. Since GUID is fixed length, it is easier to build router hardware to route by GUID.

In this paper, we assume MobilityFirst is used as the future Internet architecture (FIA), although other ICN based FIA can also be used. We will show MobilityFirst has the key enablers to solve two most important issues– sensor data security and accounting for mobile gateways.

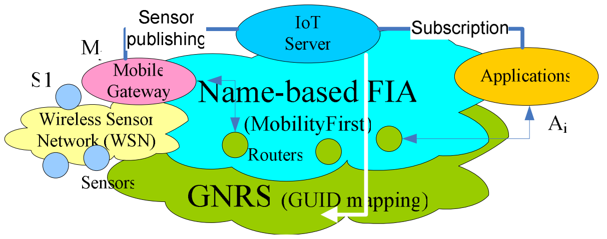

Figure 2 Mobile WSN Gateway over Future Internet

An M2M/IoT system architecture over MobilityFirst FIA with a mobile phone based WSN infrastructure is shown in Figure 2. We have same system components as in Figure 1, but a WSN gateway is a mobile phone and the IP-based Internet is replaced with MobilityFirst FIA.

The system works as follows. The owner of a sensor publishes the sensor's GUID (*S1*) and its description at IoT server. An application with GUID (*A1*) subscribes the sensor data via IoT server. While IoT server maintain a service subscription mapping from *S1* to *A1,* it will update the global name resolution service (GNRS) with an entry of *S1 -> A1.* At the hotspot of sensor *S1*, a mobile gateway collects the sensor data and send it to MobilityFirst network with a data chunk named as *S1.* At the ingress router, the GNRS is looked up with a mapping *S1-> A1* returned. Then the ingress router will lookup GNRS again to find the network location (*NA1*) of the application named as *A1* and deliver the sensor data to *NA1*.

Sensor may contain its GUID in every sensor message, however it maybe undesirable because a public key based GUID could be a big overhead for a short sensor message. IoT server may be able to manage a mapping between short IDs of sensors to their GUIDs for each hotspot. A task of a spontaneous gateway is to obtain this mapping for their frequently-visited hotspots, such as a college campus or a shopping center. They may need to check with IoT server regularly for an update. Since this mapping can be relatively static, it won't impose too much cost on mobile gateway.

As shown in Figure 3, the major cost of a mobile gateway is to deliver sensor data to MobilityFirst routers, which needs $\tau 1$ time. A wireless connection must be maintained for at least $\tau 1$ for each sensor data delivery. Since sensor data is short, it is usually only one packet per sensor at a time. A pipeline of a sequence of sensor data is unlikely happening between a mobile gateway and a router. Comparing with traditional overlay solutions that deliver data chunks to the IoT server or to the application directly, using $\tau 2$ and $\tau 3$, respectively. Apparently $\tau 2, \tau 3$ could be much larger than $\tau 1$. [13] shows the average roundtrip delay (RTT) over TCP is more than 100ms at 50 percentiles, while there are 3% RTT is less than 10ms. Assuming the ingress router is within this 3%, $\tau 1 < \tau 2/10$ is a reasonable assumption. With a shorter delivery time, the capacity of a mobile gateway can be increased accordingly. At the same energy cost, a mobile gateway can deliver ten folds of sensor data over MobilityFirst FIA than over current Internet.

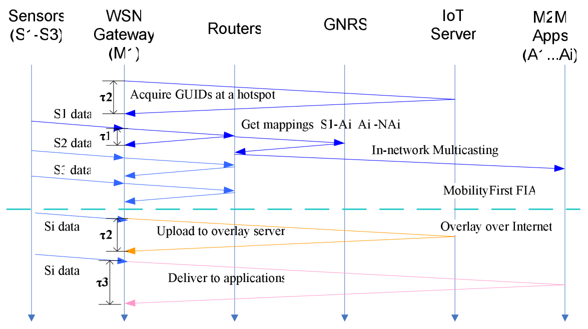

Figure 3 Data Flow through Mobile Gateways

MobilityFirst FIA has many other useful features that can benefit mobile gateways. First, it supports in-network multicasting. If there are multiple subscribers $\{Ai\}$ for a sensor *S1,* the mobile gateway needs only transmit once of *S1's* data to its access router. It can avoid the bottleneck at IoT server without increasing the load on the mobile gateways. Second, it supports late binding. A mobile gateway needs not to have a firm location of the subscribing application in order to deliver the sensor data. MobilityFirst network has store-and-forward capability that can hold the data in a cache until the destination location is found (GUID to network address is bound). In case of a destination is mobile, a rebind can be performed at a router when sensor data fails to be moved forward.

IV. WSN Coverage by Mobile Gateways

The first question to this mobile phone based WSN is the throughput of gateways, that is, will it provide enough coverage for sensors at hotspot locations? To simplify the problem, we can measure the data rate requirement from sensors as the number of sensor messages to be delivered over time, as expressed in the following definition:

$$R_T(k) = \sum_{n=1..N} \int_{kT,}^{(k+1)T} S_n(t)\, dt \quad k = 0, 1,.., K$$

where $S_n(t)$ represents the number of sensor messages to be delivered for sensor *n* at time *t*. There are total number of sensors of *N*. *T* is the time scale/unit used to measure the average number of sensor message over a period of *KT*. For examples, *T* could be one minute, one hour or one day over a *KT* of one hour, one day or one month, respectively. $R_T(k)$ is then representing the average number of sensor messages to be delivered at the time period $T_k = [kT, (k+1)T]$.

On the other hands, we can measure the capacity that mobile gateways can offer. The capacity of each mobile gateway can be defined as the number of $\tau 1$ length time periods the mobile can offer as a gateway. The total capacity at a hotspot can be expressed as:

$$C_T(k) = 1/\tau 1 * \int_{kT,}^{(k+1)T} \int_s p(s,t)*s\, ds\, dt \quad k = 0,1...K$$

where $p(s,t)$ is a random process of the number of mobile gateways available at time *t*. $C_T(k)$ then depicts the number of accumulated time period with length of $\tau 1$ from all possible gateways. This is the total number of sensor messages all mobile gateways can deliver between $[kT, (k+1)T]$. That is the capacity of mobile based WSN infrastructure at a hotspot.

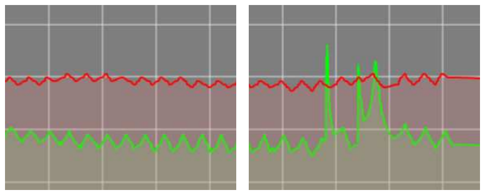

Figure 4 Aggregated capacity (red) vs. data rate (green)

We compare the capacity and requirement relationship by put them in same time scale. Figure 4. is for illustration only, which is not based on any real data. 4.a shows the daily average of WSN capacity (upper, red line) and sensor data rate (lower, green line) for a month. We can see the capacity meets the data rate requirement. 4.b shows the hourly average over a day. We can see there are few peak hours that the capacity cannot meet the rate requirement. It is common that the requirement at a smaller time scale can be harder to meet than what at a larger time scale. An IoT operator should exam the capacity and data rate requirements for all locations at different time scales. Based on the examination, the operator can make following control actions:

1) Adjust sensor transmission schedule to reduce the peak aggregated data rate, i.e., to smooth $S_n(t)$ can reduce the peak of $R_T(k)$.

2) Adjust mobile gateway data collecting schedule to reduce or increase the aggregated capacity over time. If $C_T(k)$ is much higher than $R_T(k)$ at all time scales, there is an over supply. We can add a utility function to control the percentage of mobile phones or their time to participate as gateways.

In case the peak data rate exceeds the capacity at a time interval T_k, i.e., $R_T(k) > C_T(k)$, it implies the QoS for some sensors cannot be met. Since the available capacity is a random process, there is a possibility that the QoS cannot be met even if the average numbers are shown OK. However, even if an application has a very high QoS requirement, i.e., very small delay tolerance, the mobile based WSN can still help. For example, in the smart city use case we gave in introduction, if mobile gateways can pickup 90% of sensor data, the city government only needs to send a mobile team to collect the rest 10% of sensor data in the area with less mobile phone traffic. It will still be big a saving for the city government.

It is reasonable to assume that majority of IoT applications have very low data rate and require QoS at very large time scale. With the ever-growing smart phone users today, the coverage potential of mobile phone based WSN is huge. The problem is then not only to find enough mobile users to participate the program but also to utilize mobile phones at the lowest cost. This is a far more challenging issue to study. For example, an operator may setup an incentive program based on the game theory and let mobile phones in a hotspot to bid the reward. On the other hand, the gateway function on the mobile phone must be smart enough to value the incentive based on its status. We leave this issue to future work.

V. Secure Delivery of Sensor Data

The success of this WSN infrastructure needs a sound business model, which relies on a secure sensor data delivery through spontaneous mobile phones. This section shows secured delivery protocols based on the security functions run on sensors, gateways and/or MobilityFirst routers.

Any pay service must guarantee a quality of service (QoS). For IoT services, a key QoS metric is the security assurance. Sensor data security includes the data integrity that ensures the data is from the right source and the data access control that only allows subscribers to access the data. Since the IoT service operator needs spontaneous mobile users to be the contributors of the system and incentives are paid to them, a secured accounting record must be maintained for each contributor.

As shown in Figure 5., the IoT operator wants to ensure that sensor data are from trusted sensors via trusted mobile gateways; and applications can receive the correct information, which is denoted on path ①. If sensor data goes through an untrusted mobile gateway, the data is either altered (③) or not altered (②). If a fake sensor is trying to mimic a real one, wrong information could be received by a subscribing application (③④). The design goal of the WSN is to avoid the security holes (③④) in Figure 5.

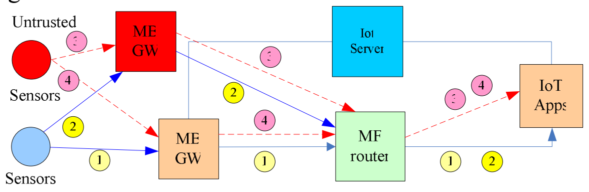

Figure 5 Security holes of using mobile WSN GW

A trusted sensor is a sensor registered on IoT server by the owner of sensor and runs only the software given by the owner.

A trusted gateway is a mobile phone runs a trusted binary module downloaded directly from the IoT service operator. We call this module the *gateway function (GWF)*. At its broadest scope the GWF is capable of providing the following functions:

1. **Sensor identity management:** provide translation to full self-certifying GUIDs for constrained sensor devices with shorter non-self-certifying local IDs
2. **Data integrity:** provide digital signing of data and identity for sensors with limited resources and no native crypto capabilities
3. **Data encryption:** encrypt sensor data with key securely obtained from M2M server on behalf of constrained sensors
4. **Operation auditing:** secure reporting of functions rendered for remuneration and logging

All of the above functions require execution in a trusted environment. Furthermore, credentials are to be securely stored and protected from external unauthorized activities. These guarantees can only be met with appropriate hardware support, such as provided by a Trusted Platform Module (TPM)[14]. Several of today's mobile devices are equipped with a TPM, making pervasive availability of such environments a realistic assumption in the future Internet.

The security assurance can be achieved via security functions run on either sensors, mobile gateways or MobilityFirst routers. We will discuss each option separately.

A. Sensor-based Delivery Protocol

We assume sensors can run both signature and encryption functions onboard. These sensors must store a private key *E1* that pairs to the public key in its GUID *S1*; and they also carry a symmetric key *H1*. Both *E1* and *H1* can be an individually assigned to each sensor or group assigned to a set of sensors. The simplest message format from such a sensor will contain its GUID, a signature of data and the encrypted data.

Message = S1 + E1[hash(Data)] + H1[Data]

Mobile gateways can just pass this message to MobilityFirst network without doing anything. The network will deliver the message to subscribing applications who have obtained the encryption key *H1* from IoT server. The applications can verify the data integrity by verifying the signature with the public key in *S1* and then decrypt the data with *H1*.

The main advantage of sensor-based protocol is the simplicity; Mobile gateway do nothing but transport of data. Even an untrusted gateway cannot alter the sensor data.

The disadvantage is the heavy load on sensors. For a short message, a signature and a long GUID add a relatively overhead. The signature operation could be costly compared to simple encryption with a symmetric key. And since both *E1* and *H1* may be used for a set of sensors in order to reduce the management cost, if one sensor got compromised and the keys are stolen, the rest of sensors are in danger. Subscribing applications need to know *H1* to decrypt the data, this also put *H1* at a risk to be stolen or re-used without authorization.

A bigger issue for pure sensor-based protocol is hard to record legitimate gateway service activities that mobile gateways provide, then difficult to reward them.

B. Security Delegation to Mobile Gateways

Now let us look at the trusted mobile gateway that runs a GW function (GWF). IoT service operator delegates GWF to perform signature, decryption and re-encryption of sensor data. In addition, GWF keeps an accounting record for the service gateway provides.

Now, a sensor can use a shorter local ID $S1_{loc}$ and a symmetric key $H1$, it sends the data message as:

$$Message = S1_{loc} + H1[Data]$$

When a mobile gateway enters a hotspot, the GWF will download a list of registered sensors from the database on the IoT server. Each entry contains local ID $S1_{loc}$, GUID $S1$, symmetric keys $H2$ and $H1$. The GWF on mobile gateways has a private key $E2$ paired with the public key in GUID of GWF $G2$. The message sent from GWF contains the GUID of the sensor, signature of data and encrypted data:

$$Message = S1 + E2[hash(Data)] + H2[Data]$$

A subscribing application receiving the message will be able to verify the data integrity by using the public key of GWF, which is obtained from IoT server at the time of subscription. Subscribing applications also obtain the symmetric key $H2$ to decrypt the data. $H2$ can be made for each sensor so that there is less risk for applications to misuse it without authorization.

Theoretically, $H2$ should be a dynamic group key for a multicasting group that contains different decryption key for each application. Since $H2$ is easier to renew between IoT server and GWF on mobile gateways, IoT operator may choose to use a simple symmetric key $H2$ instead of paying the overhead to manage a dynamic group key.

The gateway delegation protocol imposes much less load on sensors. Message is shorter with a shorter local ID and no signature. In addition, encryption key $H1$ is not exposed to applications, which reduces the risk of $H1$ being stolen.

With the GWF, it is possible to maintain an accounting record for mobile's gateway services, which is explained as following:

C. Accounting on Mobile Gateway Service

We assume an IoT operator may only reward trusted mobile gateways that run the GWF downloaded from its sever. Inside the GWF, an accounting module can be executed to record all services of GWF, including sensor data collection, actuator control, security process and key renewal etc. The records must be made available to the operating system and signed by the private key of GWF. The signature may also include the public key (GUID) of the mobile phone to authenticate the records' ownership. A simple accounting mechanism can be the total CPU time used by the GWF, normalized by the CPU power. In case the air time is charged to the gateway phone data program, the cost is separately accounted. The accounting records must be periodically uploaded to IoT server.

There is a risk that GWF is compromised even though it runs on a TWP. The IoT operator needs to renew GWF including the private key periodically. In order to minimize the lost due to the gateway function compromise, the IoT server may need to randomly assign a private key Ei for a different set of GWF at different hotspot locations and/or for different set of sensors. Only IoT server knows the mappings of the key to sensors, which are synchronized to subscribing applications for data integrity check.

GWF is owned by the IoT operator and it takes resource from the mobile phones owned by mobile users. In general, we can use GUID based AAA to perform authentication, authorization and accounting processes between a GUID identified service request and a GUID identified service host, in this case, a GWF and a mobile phone, respectively.

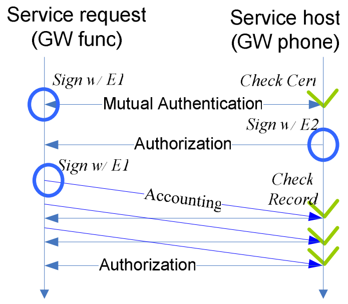

Figure 6. GUID based AAA between request & host

As shown in Figure 6. The first step is mutual authentication of GUIDs by verifying the certificates of public keys of both. The IoT service operator's public key must be certified by a public authority because it is considered as the paying party. The second step is that mobile phone authorizes GWF to use its resource. The permission is signed by the public key of the phone to avoid later dispute. The third step is the GWF accounting on the mobile phone. Theoretically, each service transaction must be signed by both service and host. In order to reduce the burden, the accounting can be made on batch of services or simply based on time period. That is, for example, each 5 minutes of service needs an accounting transaction between service and host.

Even though GWF is trusted, the sensor data can still be dropped before it is sent out to the network. In gateway delegation protocol, there is still no network side confirmation to GWF that the gateway service is really performed. On the application side, there is no way to differentiate the sensor data is from which mobile phone. And involving applications for internal system accounting is by itself not the right way to do.

D. Security Delegation to MobilityFirst FIA

MobilityFirst router is designed to be capable of selective routing based on packet GUID. One scenario is to check the integrity of a data packet: if the packet fails the signature verification by the public key in its source GUID, the packet is dropped.

We assume an ingress filtering function (IFF) is requested by IoT service operator to be hosted at ingress routers. The IFF can perform following functions:

1. **Filter fake data:** a data integrity check, if a message is sent without proper signature, it is dropped at the entry.
2. **Re-encryption of data:** a decryption of data from GWF and re-encrypt it before forwarding

3. **Accounting confirmation:** an accounting record based on checking the signature of the mobile gateway, so that the accounting record can be confirmed.

When an ingress router has the IFF, the message flow becomes:

Sensor message is unchanged

$$Message = S1_{loc} + H1(Data)$$

GWF of Mobile gateways has no need to have *E2, H1, H2,* it uses mobile phone's private key *E3* to sign the encrypted data. The message is:

$$Message = S1 + E3(hash(H1(Data)) + H1(Data)$$

On the ingress router, when mobile phone is connected, the router obtains its GUID *G3* that contains the public key paired to *E3,* then it can verify the signature from either sensor or GWF. IFF replaces GWF's tasks of signature, decryption by *H1* and re-encrypt data by *H2*. Then the message is:

$$Message = S1 + E2(hash(Data)) + H2(Data)$$

H2 is still used for access control purpose. However, MobilityFirst provides another way for access control, that is, GNRS mapping. GNRS has a mapping entry *S1->Ai,* if and only if *Ai* is a valid subscriber through IoT server. Therefore, if applications can trust MobilityFirst core network as a trusted platform, the message from IFF can be simply:

$$Message = S1 + Data$$

The MobilityFirst FIA delegation protocol makes mobile gateway's GWF lightweight. And more importantly, it confirms the accounting records on the network side. Although IFF on ingress routers is another cost for IoT operators, as a complete system solution, it is necessary and much cheaper than confirming everything with the IoT server.

VI. PROTOTYPE SYSTEM IMPLEMENTATION

Shown in Figure 7 is the proof-of-concept prototype system with sensors and subscribing applications at the edges, Android-based mobile gateways at the access, MobilityFirst routers at the access and core of the network, and the GNRS and IoT servers providing name resolution and IoT resource management. Except for the IoT server, we utilize nodes within ORBIT wireless testbed to instantiate the various services.

Sensors used in this first version are transmit only, measure temperature and humidity, and communicate over the 2.4GHz band using a protocol similar to IEEE 802.15.4 LR-WPAN. In subsequent deployments, Bluetooth will be the choice protocol for sensor data access, and matches the projection of ubiquitous availability of Bluetooth v4.0 with low-energy consideration on all smart phones. All sensors use a low duty cycle of one minute, which extend their lifetime to as long as 20 years with a coin battery.

On the Android-based mobile gateway, we use an USB reader to capture the sensor data and pass it onto the gateway function module. The GWF is built as a Java application that runs atop the Dalvik VM. While software integrity assessment techniques on a trusted platform enable attesting the kernel and modules (or applications) directly loaded by the kernel, applications or classes loaded by the VM can break the chain of trust. This can be addressed by adding integrity assessment hooks within the VM, for instance in the findClass() function of the PathClassLoader, that can measure, log and report integrity of the classes of the GWF to ensure trusted execution. The Samsung Galaxy Nexus phones (w/ WiFi a/b/g/n and 3G) we use here do not yet support a TPM.

The phones access the MobilityFirst network over the WiFi interface and run a GUID-based protocol stack in addition to the IPv4 stack. The IoT server, presently hosted outside of the ORBIT testbed, is accessed over the IP network. As shown in Figure 8, GWF first obtains a list of sensor IDs for the current location. For each sensor data message acquired, a set of validations and security operations (described in previous sections) are performed, and the data is sent over the MobilityFirst stack to the access router. Since data in MobilityFirst is propagated reliably in a hop-by-hop manner, the data connection to the access router need only be active for the duration of an one-hop transfer. The forwarding operation is logged locally and reported to the IoT server. Such accounting reports can be batched for efficiency, which also addresses any spotty connectivity limitations.

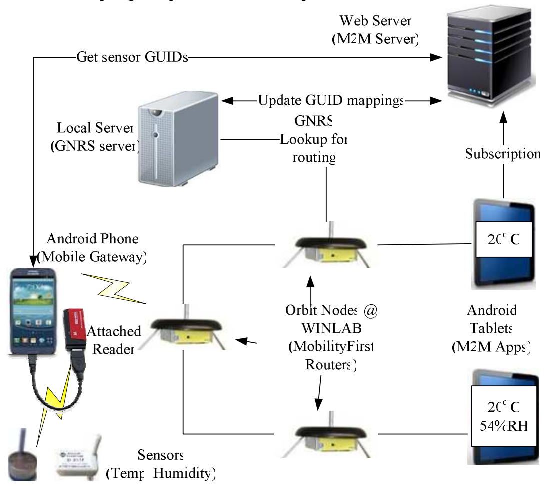

Figure 7 Prototype system components

The IoT server runs on cloud-hosted platform, running a PHP-based front-end, and an SQL back-end for data persistence. We should note that the M2M services could be easily implemented in a distributed manner to achieve scalability and low access latency, with some acceptable overheads for consistent operation. The server maintains two main tables. First is a resource table with details of all registered sensors including owner, location, ID (unique to a location), type, unit and GUID (globally unique derived from owner's public key). M2M applications are also registered as resources with details such as GUID and owner. Second is the subscription table, which contains the GUID mappings from sensor resources to application resources. When an application with GUID A1 is granted subscription to sensor with GUID S1, a mapping from S1 to A1 is inserted into the subscription table. A corresponding entry is also added to the GNRS that maps a GUID (U_{S1}) (subscription to S1's data) to A1, i.e. {U_{S1} A1}. Additional subscriptions to S1's data results in appending to the set of GUIDs mapped to in the GNRS, for example: {U_{S1} A1, A2, A3 ….}. In addition to sensor and subscription management, the IoT server manages the set of cryptographic credentials, the GWF binaries, and the accounting and billing for the gateway services.

The MobilityFirst routers are implemented as Click modular router extensions, with support for hop-by-hop data transfer, GUID-based forwarding, dynamic GNRS resolution, multicast/anycast, and temporary data storage for handling poor wireless conditions. For data packets received from the mobile gateway, the router looks up the GNRS to determine forwarding addresses. For instance, in the sample mappings inserted above, the packet originating from sensor S1 and destined to U_{S1}, would be forwarded to A1, A2, A3 and so on. For multipoint delivery, routers use multicast by adapting a longest common path heuristic. The difference with IP multicast is that routers rely on GNRS for group membership and use lightweight forwarding structures that are already part of intra- or inter-domain routing.

VII. Conclusion and Future Work

This paper presents a mobile phone-based WSN architecture for IoT service platform. It shows that a name-based FIA provides key features to implement lightweight mobile gateways for the platform. The system capacity of such an platform is analyzed compared to the sensor data rate at a given hotspot. Secure sensor data delivery protocols are proposed in order to ensure data integrity, avoid unauthorized data access and apply accounting on the service of sensor data delivery. A prototyping system architecture is presented of which the data delivery implementation is completed. Future work includes incentive model design to reward mobile contributors to the system, WSN capacity analysis based on real user mobility data and the implementation of the security delivery protocols for the prototyping system.

References

[1] Luigi Atzori, Antonio Iera, Giacomo Morabito, "The Internet of Things: A survey", Computer Networks, May 2010.

[2] "Initial M2M API analysis", EU FP7 project IoT-A (Internet of Things architecture". 2011

[3] Geng Wu et al. "M2M: From mobile to embedded internet", Communications Magazine, IEEE, April 2011; Volume: 49 Issue:4

[4] BAYIR, M. A., DEMIRBASA, M. & EAGLE, N. (2010), Mobility profiler: A framework for discovering mobility profiles of cell phone users. Pervasive and Mobile Computing, 6 (2010), 435-454

[5] On-Ramp Wireless Inc. http://www.onrampwireless.com/

[6] SigFox, http://www.sigfox.com/en/

[7] OneM2M, http://www.onem2m.org/

[8] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in IEEE Global Telecommunications Conf., vol. 1, Dec. 2003, pp. 377–381.

[9] ICNRG, IRTF Information centric networking research group, http://trac.tools.ietf.org/group/irtf/trac/wiki/icnrg

[10] V. Jacobson et al., "Networking Named Content," CoNEXT '09, New York, NY, 2009, pp. 1–12.

[11] MobilityFirst: NSF FIA project. http://www.nets-fia.net/

[12] T. Vu, A. Baid, Y. Zhang, T.D. Nguyen, J. Fukuyama, R.P. Martin, D. Raychaudhuri, "DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet", The 32nd Int'l Conference on Distributed Computing Systems, ICDCS 2012.

[13] Jay Aikat , Jasleen Kaur , F. Donelson Smith , Kevin Jeffay, Variability in TCP round-trip times, Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, October 27-29, 2003, Miami Beach, FL, USA

[14] Nauman, Mohammad, et al. "Beyond kernel-level integrity measurement: enabling remote attestation for the android platform." Trust and Trustworthy Computing, 2010.