MF-IoT: A MobilityFirst-Based Internet of Things Architecture with Global Reach-ability and Communication Diversity

Sugang Li*, Jiachen Chen*, Haoyang Yu*, Yanyong Zhang*, Dipankar Raychaudhuri*,
Ravishankar Ravindran†, Hongju Gao‡, Lijun Dong†, Guoqiang Wang†, and Hang Liu§
*WINLAB, Rutgers University, NJ, U.S.A. Email: {sugangli, jiachen, hy214, yyzhang, ray}@winlab.rutgers.edu
†Huawei Research Center, CA, U.S.A. Email: {ravi.ravindran, lijun.dong, gq.wang}@huawei.com
‡China Agricultural University, Beijing, China. Email: hjgao@cau.edu.cn
§The Catholic University of America, U.S.A. Email: liuh@cua.edu

***Abstract*—The rapid growth in IoT deployment has posed unprecedented challenges to the underlying network design. We envision tomorrow's global-scale IoT systems should support global device reach-ability, mobility, diverse communication patterns, and resource efficiency. Existing solutions either rely on IP protocols that do not have efficient mobility support, or seek application-layer optimizations that incur high computing and deployment overhead. To fill this void, we propose to adopt clean-slate network architecture in the core network that decouples locators from network addresses, and towards this end, we propose MF-IoT, which is centered around the MobilityFirst architecture that focuses on mobility handling.**

MF-IoT enables efficient communication between devices from different local domains, as well as communication between devices and the infrastructure network. The MF-IoT network layer smoothly handles mobility during communication without interrupting the applications. This is achieved through a transparent translation mechanism at the gateway node that bridges an IoT domain and the core network. In the core network, we leverage MobilityFirst functionalities in providing efficient mobility support for billions of mobile devices through long, persistent IDs, while in the local IoT domain, we use short, local IDs for energy efficiency. By seamlessly translating between the two types of IDs, the gateway organically stitches these two parts of the network.

Through large scale simulation studies, we show that MF-IoT is able to achieve the four features we envisioned for an IoT system, with performance optimizations in reducing network traffic load, avoiding congestion, and ensuring fast and timely packet delivery.

***Keywords*-IoT, Network architecture**

I. Introduction

With the advent of new generation embedded devices, including wearables, robots and drones, Internet of Things (IoT) has received a great deal of attention in the past few years. The applications of IoT range from individual-level applications such as smart health care and smart homes, to large-scale ones such as smart cities. According to [1], the total number of connected "devices[1]" can reach 50 billion by the year of 2020. Compared to traditional sensors, these new devices are able to cope with more complex logic — sensors can carry out more local computation and actuators can be easily controlled via the network. Additionally, many devices now have higher mobility than before. These changes in the IoT devices have introduced new opportunities as well as posed new challenges to the underlying network design.

In the last few years, several solutions have been proposed to design new IoT systems, which can be generally classified into two categories based on the underlying network design. Solutions in the first category (*e.g.*, those discussed in [2], [3]) try to support the IoT system through traditional Internet Protocol (IP). However, IP has its intrinsic problems in dealing with device mobility since it couples a node's identity with its location. Also, the wide deployment of Network Address Translation (NAT) hinders global device reach-ability in scenarios such as sending invasion alarms to the user's mobile devices. To better cope with mobility and realize global reach-ability, therefore, solutions in the second category (*e.g.*, those discussed in [4], [5]) try to alleviate the above problem in the application layer. In these solutions, a variety of Wireless Sensor Network (WSN) protocols are used inside the constrained part of the networks (that mainly consist of resource constrained embedded devices), while the interactions between infrastructure nodes and sensor devices are achieved by a server (or a proxy) that runs application-layer logic. Hence, to allow embedded nodes to initiate communication with infrastructure nodes, these solutions either have to maintain long-lived connections between the client and the proxy, which will likely result in scalability issues, or rely on polling, which will likely cause traffic overhead and long latency. To make matters worse, sensor nodes deployed by different organizations are usually not compatible with each other, and therefore users will end up installing multiple servers (or server applications) to support all the sensor nodes.

In this paper, we argue that we should design a generic and efficient network architecture to support sensor nodes and infrastructure nodes across domains and organizations. We present the IoT system we envision in Fig. 1. Much more than a simple combination of WSN and core network, the proposed network architecture should have the following critical features: *1*) *global reach-ability* for all the embedded and

[1] In this paper, we refer to these devices as embedded devices, sensor devices, or IoT devices interchangeably.

978-1-4673-9948-7/16 $31.00 © 2016 IEEE
DOI 10.1109/IoTDI.2015.17

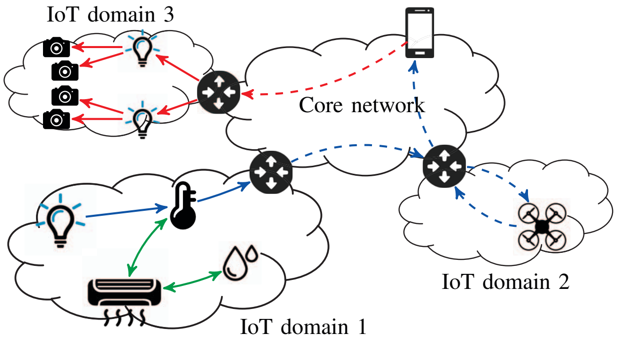


Fig. 1: The envisioned IoT system architecture consists of the core network and many local area IoT networks (referred to as IoT domains). We illustrate three applications that motivate our study in this paper. The green arrows illustrate the data exchange between an air conditioner and a temperature sensor and a humidity sensor; the blue arrows illustrate the application in which a light sensor triggers a drone in another IoT domain to take pictures and send them to a cell phone (in core network); the red arrows illustrate the application in which a cell phone synchronizes multiple cameras so that they take pictures at the same time.

infrastructure nodes, in that they can be identified and located via persistent, globally accessible, identities, *2*) *mobility support* for seamless connection in the presence of node mobility, *3*) *richer communication patterns* including direct device-to-(multiple/any) device, device-to-(multiple/any) infrastructure, infrastructure-to-(multiple/any) device communication without the necessity of going through the application layer (where the devices may belong to different local IoT domains as shown in Fig. 1), and *4*) *resource efficiency* which supports a large number of embedded devices that are severely constrained in energy, computation, storage, and/or network capacity. To build an IoT system with these features, we believe the main challenge lies in the network layer design, especially in the data plane. In this paper, we focus on this particular aspect, while adopting existing algorithms in neighbor/service discovery and routing in the proposed system.

At the center of the proposed IoT network architecture is MobilityFirst [6], a next-generation network that focuses on handling device mobility in the Internet. We choose to build our architecture on MobilityFirst because it is designed to address the inefficiencies of IP when accommodating rapidly increasing mobile and sensor devices in the Internet, which shares a common set of challenges as building a new IoT system out of these devices. In MobilityFirst, each device (or application, or even a piece of content) has a 20-byte flat *Globally Unique IDentifier (GUID)*. MobilityFirst decouples the node identity (GUID) from its location (Network Address, *NA* in short). The translation from GUID to NA is performed by a logically centralized *Global Name Resolution Service (GNRS)*. Different from DNS, GNRS is a network component transparent to the applications, which means the routers rather than the senders can perform late-binding (GNRS re-lookup) on a delivery failure when an end host moves. This design allows quick and local fail recovery to increase the delivery rate, and more importantly, the applications focus only on the GUID's rather than the changing NAs. MobilityFirst is well-suited to support the aforementioned features of the IoT architecture.

However, we need to address several challenges before we can extend MobilityFirst into the IoT networks:

1) *Long GUID's:* the usage of 20-byte GUID's in IoT is too heavy-weight, if at all possible, for layer 2 protocols like 802.15.4 [7] (which has a mere 127-byte Maximum Transmission Unit, MTU);
2) *Costly GNRS lookup:* MobilityFirst routers perform a GNRS lookup when the destination NA is unknown or the client is no longer at the NA specified in the packet (due to mobility). This operation is not feasible for storage-constrained embedded nodes;
3) *Link-state routing:* MobilityFirst adopts link-state routing [8], similar to OSPF [9], which poses high computation and storage burdens on embedded nodes.

To address these challenges, in this paper, we propose *MF-IoT*, a generic network architecture that extends MobilityFirst into the IoT world. We create a resource efficient "dialect" of MobilityFirst to allow sensor nodes to communicate within a local area, referred to as a local IoT/sensor domain (see Fig. 1). Gateways are used to translate between MobilityFirst and the dialect but this process is transparent to the applications. Unlike the application layer solutions, the dialect only exists in the network layer. Unlike NAT either, each device in MF-IoT has a GUID, like in MobilityFirst, and this GUID can be used to realize global reach-ability. MF-IoT also takes communication pattern diversity, GUID size, and computation awareness into consideration to provide rich yet light-weight support to IoT applications.

The contributions of this paper are as follows:

- We identify a list of requirements for a generic IoT architecture;
- We extend the GUID-based communication into the IoT domains to allow global reach-ability and seamless mobility handling, while using Local Unique IDentifiers (LUID) in local IoT domains for efficiency;
- We support a rich set of communication patterns, including unicast, multicast and anycast between sensor nodes and infrastructure nodes *and* among sensor nodes (that may or may not belong to the same domain);
- We adopt *service-based* GUID assignment which facilitates communication with a specific service provided by a node instead of with the node itself, and easier support for functions like caching, service mobility (anycast), *etc.*;
- Through large scale simulations, we show that MF-IoT can significantly reduce the network traffic load and effectively avoid congestion in the network, leading to packet delivery latencies that are orders of magnitude shorter. Indeed, MF-IoT can deliver packets within tens of milliseconds, while IP-based solutions encounter severe congestion and the resulting latencies are more than a few seconds.

The remainder of the paper is organized as follows: §II summarizes the envisioned requirements of a generic IoT network architecture and describes our main design rationales. The design detail is presented in §III and the architecture is evaluated in §IV. §V discusses existing IoT network architectures, and §VI concludes the paper.

II. Design Rationale

In this section, we first discuss the requirements we envision an IoT architecture should satisfy to efficiently connect billions of devices online and enable diverse interactions among them. We then give a brief description of MobilityFirst and explain why we base our design on MobilityFirst. Finally, we present our design rationales one by one.

A. Requirements of a Generic IoT Architecture

As the number of embedded devices rapidly increases, they pose a set of new challenges/requirements on the underling network design.

*1) **Global reach-ability**:* One of the salient features offered by IP is its global reach-ability – applications can reach each other by simply specifying the destination IP address in the packet header, without worrying about details such as hardware type, or the application on the destination.

This feature is particularly important for scenarios like emergency notification. In such scenarios, it is desirable that the sensors are able to notify the predefined clients without the need of going through a server or proxy, requiring each client to be associated with a unique and persistent identifier. Global reach-ability is also important for remote actuation applications, where users may need to use their smartphones to *directly* control devices such as air conditioners, rather than going through 3rd party protocols [10].

Most of the existing IoT architectures (*e.g.*, [2], [4]) focus either on the communication within a local sensor network or on the adaptation over the application layer, but we take the viewpoint that an IoT architecture should also enable transparent interactions between sensors and infrastructure nodes at little overhead.

*2) **Mobility support**:* With the rapid deployment of mobile sensors such as robots and drones, an IoT architecture should provide seamless mobility support such that applications can communicate with each other without worrying about the consequence of a node's location change or any network change (*e.g.*, new identity, new route to the target). At the same time, client devices tend to move frequently — the user's smartphone may move to a different network while in the middle of controlling a rice cooker, or listening to the security alarm at home. An IoT architecture should be able to handle mobility of different parts of the system.

*3) **Communication diversity**:* A significant departure from traditional WSNs whose main function is data collection, the new IoT paradigm aims to facilitate a larger variety of use cases and a much richer set of communication patterns.

An important communication pattern in IoT is device-to-device communication; a sensor should be able to directly communicate with an actuator rather than being connected by a server. This communication pattern can cut down response time, traffic volume and potential failures caused by the server, all of which are critical to real-time IoT systems. Additionally, direct communication between a device and multiple devices should also be supported.

Another communication pattern needed by IoT is direct communication between a device and infrastructure nodes. In fact, the *observe* mode described in [11] follows this communication pattern — a client registers a certain event, and when that event is detected by a device, it would send notifications to the client directly.

Finally, anycast should be supported, delivering messages to any node from a group.

*4) **Resource constraints and heterogeneity**:* Even though today's IoT devices are becoming increasingly more powerful, their resource constraints remain a big issue. Many of the embedded devices still have very limited computation, memory, and communication capability. Moreover, embedded devices vary greatly in their capability. As a result, an IoT architecture should take into consideration these factors.

B. Background on MobilityFirst

MobilityFirst [6] is proposed as a future Internet architecture with mobility and global accessibility as core design concerns. To achieve these features, MobilityFirst introduced several components into the network:

1) Globally Unique IDentifier (GUID): MobilityFirst utilizes persistent GUID to name every network object. The separation between the identifier (GUID) and the locator (network address, *NA*) provides support for mobility and global accessibility. Meanwhile, GUID can be a public key derived from the properties of the object or a human-readable name, hence it allows the objects to be self-certifiable.

2) Global Name Resolution Service (GNRS): GNRS is a logically centralized service that maintains the mapping from the GUID of an object to its current NA(s). MobilityFirst routers can perform late binding — querying the GNRS whenever a destination NA could not be resolved in the local scope. This is a network-layer solution which is different from DNS, and it provides better support for mobility since the network has the potential to recover a delivery failure locally. Works in [12], [13] proposed distributed solutions for GNRS implementation which can have scalability and acceptable lookup performance in the core network.

3) Routing: MobilityFirst routes packets based on the NA(s). Work in [8] proposed a basic routing solution in MobilityFirst similar to Open Shortest Path First (OSPF). In this solution, each router maintains the global topology and calculates the shortest path to the destination in a distributed manner.

4) Service ID: To support multiple network services such as unicast, multicast, and in-network computing, service ID is included in the packet header so that each router is capable of making decision based on its policy.

Based on the aforementioned components, MobilityFirst has the potential to be a network architecture for IoT. However, some challenges remain for the deployment in IoT systems in which many resource-constraint devices might exist. First of all, MobilityFirst uses a 20-byte flat GUID. If the network operator tries to run the low-rate network (*e.g.*, IEEE

802.15.4), it will be inefficient in data transmission. Secondly, GNRS operations remains unrealistic for the low-end devices since they may not have direct link to the GNRS server, nor does it have enough storage to support store and forward in late binding. Routing scheme also needs to be optimized to preserve energy consumption if we want to use MobilityFirst in IoT. Therefore, in this work, we propose MF-IoT, a generic network architecture that extends MobilityFirst into the IoT world, providing rich (yet light-weight) support for different applications and communication patterns.

C. MF-IoT Design Rationales

Based on the requirements, we propose MF-IoT, an architecture that extends MobilityFirst to allow seamless communication among IoT nodes and between IoT and infrastructure nodes. We build our architecture over MobilityFirst because of its inherent support for reach-ability (via 20-byte persistent GUID) and mobility (via late-binding in the network layer [6]). Accordingly, we create a much lighter-weight protocol embedded devices can use within a local IoT domain to meet their resource constraints. In order to achieve global reach-ability, we use network-layer translators (gateways) to provide transparent translation between the light-weight protocol and MobilityFirst.

*1) **GUID vs. LUID**:* The 20-byte GUID is a key feature in MobilityFirst to provide mobility support. Each device would have a persistent and unique GUID no matter when and where it moves. It is also important for MF-IoT to keep this feature in achieving global reach-ability and mobility handling.

However, always carrying the 20-byte GUID's (40 bytes for a source-destination pair) in the packet header may not be always feasible over a low-rate layer-2 protocol such as 802.15.4. To solve this issue, we first introduce a lighter-weight packet header (total length of 10 bytes, see §III-B)) and a 2-byte *locally unique ID* (LUID in short). In this way, we map a device's 20-byte GUID to its 2-byte LUID when we reach the local area IoT domain. To cope with collisions that may occur in this mapping process, we let each domain have its own GUID to LUID mapping which is managed by a gateway deployed at the edge of the domain.

Different from NAT and other existing domain-based solutions, MF-IoT does not change the identity the application uses. The applications, either on constrained IoT devices or on the infrastructure nodes, still use the 20-byte GUID to identify each other, while the network performs translation which is transparent to these applications (see §III-C for detailed explanation). An IoT node carries its GUID no matter where it moves, even when it is relocated to another local IoT domain and is assigned with a new LUID. This ensures the global reach-ability and mobility handling yet still considers resource constraints of embedded devices.

*2) **Service-based GUID**:* In MobilityFirst, a GUID can be used to identify a network node, an application, or even a piece of content. In MF-IoT, we associate a GUID with a specific service, hence *service-based* GUID's. Here, service has a finer granularity than a network node since in IoT, a node often carries multiple services — *e.g.* a robot might carry a light sensor, a temperature sensor and several actuators, each of which provides a service. Its granularity is similar to that of a "port" in the TCP/UDP definition – each application can have its GUID('s) that are exposed to the network and other applications. In MF-IoT, we name each individual service instead of the node GUID + port approach like in TCP/UDP.

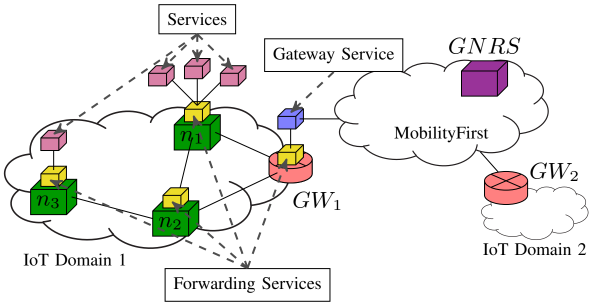


Fig. 2: MF-IoT architecture overview.

With service-based GUID's, applications on an IoT node can simply listen to one or more GUID's for different services, *e.g.* sensor data reporting, actuation, caching, *etc.* With this approach, we can easily support transparent and energy efficient service migration, without affecting the functionality of the services (see §III-D for detail).

MF-IoT also treats message forwarding and gateway as services, allowing simpler topology management and logic separation in IoT especially when multiple services co-locate on a single IoT node (see §III-E).

*3) **GUID-centric communication diversity**:* MF-IoT is well suited to support direct device-to-device communication, no matter if these two devices are in the same domain or not. The applications on the devices can identify each other with corresponding service GUID's while the underlying network takes care of the translation between GUID and LUID. IoT applications can also reach infrastructure nodes easily, through their GUID's.

MF-IoT does not distinguish unicast and multicast services. Whenever there are multiple services listening for the same GUID, the network would forward a message to *all* of these services. However, MF-IoT distinguishes *to-all* services from *to-any* services (anycast). This is achieved by a "Service ID" (SID) field in the packet header similar to MobilityFirst.

III. MF-IoT Architecture Design

In this section, we describe the detailed design of MF-IoT. We first present the components in MF-IoT, the data packet format, and then explain how the components work together to provide the features discussed in §II-A.

A. Components in MF-IoT

MF-IoT consists of the following components (Fig. 2):

1) IoT/Sensor domain: We refer to a local area IoT/sensor network as an IoT/sensor domain. A large portion of the nodes in the IoT domain are resource-constrained, where energy-efficient link and physical protocols such as 802.15.4 or Bluetooth Low Energy (BLE) are primarily used.

TABLE I: MF-IoT packet format (4 octets per row).

0 1 2 3	4 5 6 7	0 1 2 3	4 5 6 7	0 1 2 3	4 5 6 7	0 1 2 3	4 5 6 7
VER	PKT_TYP	SVC_TYP	PROT	TTL	PKT_LENGTH		
SRC_LUID				DST_LUID			
NONCE				PAYLOAD			
PAYLOAD							

2) MobilityFirst domain: MobilityFirst domain refers to the infrastructure network consisting of MobilityFirst routers.

3) Gateway: A gateway (*e.g.*, GW_1 and GW_2 in Fig. 2) serves as the bridge between local IoT domains and the MobilityFirst domain, translating between MF-IoT packets and MobilityFirst packets for each local domain. In order to be compatible with both ends, multiple physical interfaces should be adopted by a gateway node.

4) Network nodes: In our MF-IoT architecture, network nodes can be categorized in three classes: constrained nodes within an IoT domain, resource-rich nodes within an IoT domain, and infrastructure nodes. Note that a resource-rich node within an IoT domain usually has rich computing and storage resources (*e.g.*, camera), but the network interface is compatible with that of a resource-constrained node.

5) Service: We treat any resource that might be of interest to users (such as a sensor or an actuator) as a service which is the unit of GUID assignment.

To provide basic network functions while separating them from the application logic, we treat forwarding and gateway also as services (namely, *forwarding service* and *gateway service*). They provide functions like neighbor discovery, routing, packet forwarding and translation between MF-IoT and MobilityFirst packets.

Application services refer to IoT resources such as sensing and actuating. While multiple such services can be provided by a single IoT node (or even within a single application for embedded devices), a single service can also be supported among multiple sensor nodes (details in §III-D).

B. Packet Format

In MF-IoT, we use fixed-length headers instead of Type-Length-Value (TLV), so that less space and computing is needed. As shown in Table I, we define the following fields:

VER: the version number of MF-IoT packets;

PKT_TYP: type of a packet, can be data, control, *etc.*;

SVC_TYP: the network service type, *e.g.*, multicast, anycast;

PROT: the upper layer (*e.g.*, transport layer or application layer) protocols, which helps map the upper layer to the corresponding logic;

TTL: time-to-live, used to prevent routing loops;

PKT_LENGTH: length of a packet, can allow packet size up to 4kB (2^{12} bytes);

SRC_LUID & DST_LUID: source and destination;

NONCE: a random number generated by the sender. In MF-IoT multicast, the link layer of the branching node broadcasts the packet instead of unicasting to every next hop. The previous hop will receive the same packet, and drop it if packet with same nonce is seen before.

Algorithm 1 *Send* function implementation in MF-IoT

```
 1: procedure SEND($G, d$)
        parameters:
        $G$: destination GUID
        $d$: data to be sent
 2:     $tmp \leftarrow t_i[G]$                          ▷ Lookup local translation cache
 3:     if $tmp = \emptyset$ then                        ▷ Initiating communication
 4:         $L \leftarrow$ REQUEST($G, \emptyset$)
 5:     else if $tmp.State = Stale$ then                 ▷ After move
 6:         $L \leftarrow$ REQUEST($GL, tmp.LUID$)
 7:     else                                             ▷ Continue communication
 8:         $L \leftarrow tmp.LUID$
 9:     end if
                            ▷ Forward MF-IoT packet based on routing
10:     FORWARD($L_i, L, d$)
11: end procedure
```

C. Transparent GUID/LUID Translation

To enable global reach-ability, MF-IoT uses gateways as the bridge between the MF-IoT and MobilityFirst domains. It maintains the mapping between GUID and LUID via a translation table. The translation table contains 3 columns (as shown in Table II) — the GUID, its LUID in the local domain and the mapping type. The mapping type can be *Local*, meaning the GUID is in this local domain, *Remote*, meaning the GUID is outside of the local domain, or *Local+Remote*, which is usually a multicast GUID and means that there receivers are both inside and outside the domain (see §III-F). The LUID in the local domain can be recycled based on Least Recent Used (LRU) or Time To Live (TTL) policies. This ensures the uniqueness of LUID in a local domain during a period of time even with a 2-byte length (which allows 65,536 concurrent GUID mappings).

When an embedded device joins a domain, it registers its GUID's (each service has a GUID) at the gateway. The gateway would give each GUID a LUID and mark them as *Local*.

When an application tries to send a message to a certain GUID (G), it would call the *send* function provided by the host node's forwarding service (see Algorithm 1). Note that in this process, the LUID is transparent to the application. The forwarding service requests G's LUID from the gateway (lines 3–4), and the gateway looks up G in its translation table. If there is already a mapping, the gateway simply replies with the LUID; otherwise it creates an entry $\{$GUID=G, LUID =L, Type=Remote$\}$ in the translation table, where L is randomly generated (different algorithms could be adopted here, which is orthogonal to this study). Note that in this stage, the gateway does not have to perform a GNRS lookup and it can respond to the request immediately. After getting G's LUID, L, the forwarding service checks its own routing and neighbor table to forward the packet using L as the destination LUID.

The forwarding service can also have a local translation cache (t_i in Algorithm 1) for frequently communicated parties. Before requesting G's LUID from the gateway, the forwarding

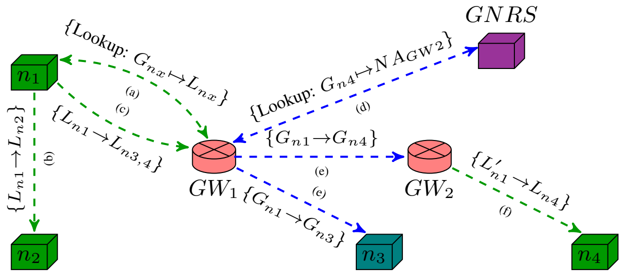

$GNRS$
$\{$Lookup: $G_{nx} \mapsto L_{nx}\}$
(a)
(c)
$\{L_{n1} \rightarrow L_{n3,4}\}$
n_1
$\{L_{n1} \rightarrow L_{n2}\}$
(b)
n_2
$\{$Lookup: $G_{n4} \mapsto NA_{GW2}\}$
(d)
$\{G_{n1} \rightarrow G_{n4}\}$
(e)
(e)
GW_1
$\{G_{n1} \rightarrow G_{n3}\}$
GW_2
$\{L'_{n1} \rightarrow L_{n4}\}$
(f)
n_3
n_4

Fig. 3: Illustrations for global reach-ability (blue lines: MobilityFirst traffic, green lines: MF-IoT traffic).

TABLE II: Translation table on GW_1 in Fig. 3.

GUID	LUID	Type
G_{n1}	L_{n1}	Local
G_{n2}	L_{n2}	Local
G_{n3}	L_{n3}	Remote
G_{n4}	L_{n4}	Remote
…	…	…

service can first check its own cache (lines 2 and 8). The cache could have stale information when a node moves to a new domain, but we try to keep the original LUID information to reduce changes in the routing tables. Therefore, when requesting the LUID of a stale entry in the cache, the forwarding service would carry the original LUID as its preference (lines 5–6). If there is no collision, the gateway would register this original LUID in its translation table.

Upon arrival of a MF-IoT packet, a gateway looks up its translation table and obtain the GUID's for both source and destination forward the packet using MobilityFirst logic. At this point, it might need to look up GNRS for the destination node's NA if it is unknown. On the other hand, when the gateway receives a MobilityFirst packet whose destination GUID (G_d) is in its domain (the translation table has a matching entry whose type is *Local*), the gateway would create a LUID (L_s) for the source GUID (G_s) and mark the type as *Remote* if the source is not in the translation table, and then send a MF-IoT packet consisting of L_s and L_d. This entry is created such that the destination device can send a message to the sender.

In MF-IoT, the gateway does not differentiate if a MobilityFirst packet is coming from an infrastructure node or an embedded node in another domain. This feature enables global reach-ability and seamless mobility handling. Below we explain how these two objectives are achieved.

*1) **Global reach-ability**:* Fig. 3 depicts three scenarios where an embedded node n_1 wants to send a message to a node in the same domain (n_2), an infrastructure node (n_3), and an embedded node in another domain (n_4), respectively. To simplify the description, we assume that each node has only one forwarding service and one application service, represented by a box. In Fig. 3, MF-IoT traffic is represented by green lines and MobilityFirst traffic is represented by blue lines. Note that we use dotted lines here to denote that the traffic is not direct traffic between the two nodes, but there might be relay nodes between them. We next describe the protocol exchange according to the figure.

(a) To initiate the communication with n_2, n_3, and n_4, n_1's forwarding service needs to first get their LUID from the gateway. For n_2, GW_1 can respond directly since it has a *Local* entry in the translation table. For the other two, GW_1 creates new entries and mark them as *Remote*. Here, GW_1's translation table is shown in Table II.
(b) The routing algorithm in the local IoT domain forwards the packet based on the destination LUID. Since n_2 is in the same domain, the local routing algorithm would forward the packet to n_2 eventually.
(c) If the destination LUID (L_{n3} or L_{n4}) is not in the same domain, the local routing algorithm forwards the packet to GW_1, which translates the packet to MobilityFirst packets $\{G_{n1} \rightarrow G_{n3}\}$ or $\{G_{n1} \rightarrow G_{n4}\}$.
(d) Now GW_1 sends the packets with traditional MobilityFirst logic. In MobilityFirst, the first step is a GNRS lookup for the NA of the destination. If the destination is an embedded node in another domain (n_4), GNRS would reply with the NA of the corresponding gateway (GW_2). For an infrastructure node (n_3), GNRS would respond directly with its NA (not shown in the figure).
(e) After getting the NA, the packet will be forwarded in the MobilityFirst network and eventually reach n_3 or GW_2. Note that thanks to late-binding technique in MobilityFirst, the packet would reach the destination even if n_3 or GW_2 has moved and has a new NA. This provides seamless mobility support when an infrastructure node or an entire local IoT domain moves.
(f) When GW_2 receives the packet destined to G_{n4}, it checks the translation table and finds that n_4 belongs to the local domain. It then creates a LUID mapping for G_{n1} (L'_{n1}) and forwards an MF-IoT packet $\{L'_{n1} \rightarrow L_{n4}\}$. Note that the LUID of G_{n1} in this domain does not have to be the same as L_{n1} given by GW_1. However, this new LUID does not affect the communication between n_1 and n_4 since they are communicating with the GUID while LUID is kept transparent from them.

*2) **Handling node mobility**:* Next, we show how MF-IoT handles the situation when embedded nodes move from one domain to another. There are two cases we need to consider, the first involving one of the communication parties moving to a different domain (*e.g.*, n_2 moves to GW_2), and the second involving one of the communication parties moving into the same domain (n_4 moves to GW_1). In both cases, the node that does not move (n_1) will not observe any change in the communication.

In the first case, let us consider the following situation: when n_1 sends a message to n_2, n_2 has moved from GW_1 to GW_2. We assume that n_1 already initiated the communication before n_2 moved, and therefore it already has $G_{n2} to L_{n2}$ mapping in its local cache. When the packet ($\{L_{n1} \rightarrow L_{n2}\}$) reaches GW_1, either via proactive routing (which detects the node departure and updates the routing) or reactive routing (which cannot find n_2 during message forwarding and then redirects the packet to GW_1), GW_1 contains a *Remote* entry for n_2

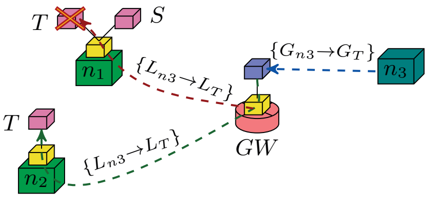


Fig. 4: Service migration. When the original temperature sensor fails, the service is migrated to a back up sensor without interrupting the application because of service-based GUID assignment.

and it will forward the packet similar to the steps (d, e) in the previous example. Note that during this process, n_2's LUID has changed, but the application uses only its GUID and is unaware of this change. If GW_1's translation table has not been updated when n_1's packet arrives, GW_1 can store the message and forward it later when n_2 reconnects from the new domain.

In the second case, let us consider the following situation: when n_1 sends a message to n_4, n_4 has moved from a different domain to GW_1 and has registered with GW_1. In this case, GW_1 has assigned a LUID to n_4, L_{n4}. When $n1$ sends a packet $\{L_{n1} \to L_{n4}\}$, it would reach n_4 without going to GW_1, without n_1's active involvement.

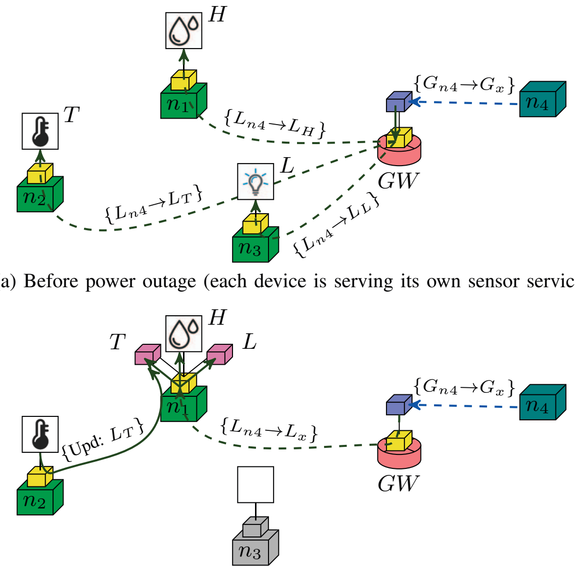
Having considered infrastructure node mobility, IoT domain (as a whole) mobility (described in the previous example), and embedded node mobility, we believe that MF-IoT can provide seamless mobility support for an IoT system.

D. Service-based GUID

With the wide deployment of IoT devices where each device can have more than one sensor services (*e.g.*, a robot may have a temperature sensor, a humidity sensor, and several actuators), there is a need to communicate with a specific sensor service rather than lumping all the sensor services together. Thus, MF-IoT gives each of these services a GUID, which enables seamless service migration and service caching. Such features would be particularly useful in extreme cases like disaster management. Below, we will discuss 2 typical use cases to illustrate the benefits of service-based GUID's compared to the traditional ID (IP) + port solution.

1) Service migration: In this case, we have an embedded node n_1 which has a temperature sensor (T) and a smoke sensor (S), and a backup node n_2 with a temperature sensor that is not in use in the beginning of the example (see Fig. 4). In this example, an infrastructure node n_3 queries T from time to time to get the current temperature. When the temperature sensor on n_1 fails, n_2 will serve the same functionality. In the traditional IP + port solution, the new T would have n_2's IP address with a specific port and accordingly, GW and/or n_3 would need to know the change. Note here that it is often not feasible to migrate the whole node and let the new node (n_2) use the original node's (n_1's) IP address because n_2 only provides a subset of services that n_1 supports. Thus, the traditional solution is inconvenient for users and application designers.

In MF-IoT, we can have the temperature sensor on n_2 take over the service T by inheriting T's GUID G_T and LUID L_T. In this way, the routing algorithm would find T's new location without any extra overhead from n_3 and GW.



(a) Before power outage (each device is serving its own sensor service)

(b) During power outage (n_2 and n_3 can be turned off and update the caching service on n_1 periodically)

Fig. 5: Service caching during power outage.

2) Service caching: There are also cases that the more powerful devices can help lower-end devices cache the sensor readings, or low-end devices collaborate and cache for each other to save energy, which we refer to as *service caching* in MF-IoT. The caching node will listen for the specific service GUID and the source sensor can update the caching node with the same GUID.

Fig. 5 shows a local IoT domain containing 3 services (a humidity sensor service H from node n_1, a temperature sensor service T from node n_2, and a light sensor service L from node n_3). During normal operations (Fig. 5a), each of three nodes has its own power source and can serve data requests from other parts of the system (*e.g.*, an infrastructure node n_4). When power outage happens (Fig. 5b), to extend the lifetime of the entire domain, they can elect a representative (n_1 in the example) and cache the latest readings from all three sensor services on n_1. n_2 and n_3 can then go to the sleep mode and wake up periodically to get the sensor reading and update the cache. In Fig. 5b, n_2 is updating the cache and n_3 is in the sleep mode. Since n_1's cache is listening for the LUID for T and L, the corresponding requests would be forwarded to n_1 and the caching service can respond with the cached value. When the battery on n_1 drops lower than a threshold, n_1 may also offload the caching service to other nodes (*e.g.*, n_3) and go to the sleep mode. Of course, the caching service can also be placed on the gateway (GW).

E. Forwarding Service and Gateway Service

MF-IoT treats the basic network functionalities within an IoT domain, such as forwarding and gateway, as services. This leads to easier topology management and better separation between application functions and network functions.

Fig. 6 illustrates a local IoT domain with 4 nodes (n_1–n_4), each of which has some or no services, and a gateway (GW).

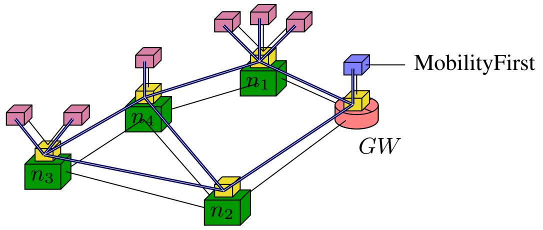


Fig. 6: Virtual topology with forwarding as a service (virtual topology marked in blue).

TABLE III: Neighbor table on n_{4F}.

Neighbor	Identity
L_{n1F}	MAC(n_1)
L_{n2F}	MAC(n_2)
L_{n3F}	MAC(n_3)
L_{n4S1}	PID($n4S1$)

TABLE IV: FIB on n_{4F}.

Destination	Next Hop
L_{n3S2}	L_{n3F}
L_{n4S1}	L_{n4S1}
L_{MF1}	L_{n1F}
$\ldots$	$\ldots$

Unlike the traditional solutions in which the services have to take care of neighbor discovery and routing, in MF-IoT, each node's forwarding service collectively performs these tasks. To send a packet, an application simply sends the data to its forwarding service and the network functions reside only in forwarding services. This clear separation would help developers focus on a specific part of the system.

When the embedded nodes move, this design can also help simplify topology management. For example, when n_4 moves away from n_1 and they cannot reach each other, this solution only has one link change (between forwarding service of n_1 and n_4) while the tradition solution would have 3 link changes.

On the gateway, we also separate the forwarding service (that relays packets for IoT nodes) from the gateway service (that translates between MF-IoT packets and MobilityFirst packets). Therefore, only the packet that will be forwarded out will go through the gateway service. This reduces the response time and energy consumption on the gateway and leads to simpler modification of routing algorithms on a gateway node.

Similar to popular WSN designs, MF-IoT separates the neighbor discovery from routing. The forwarding service on the embedded nodes maintains two basic data structures — a neighbor table (see Table III) and a FIB (see Table IV). We denote the forwarding service on node n_i as n_{iF} and the application service on node n_i as n_{iSj} in the tables. The neighbor table maintains the direct neighbors in the virtual topology. In the example, the forwarding service on n_4 has 3 neighbor forwarding services with LUID $L_{n1F} - L_{n3F}$ (the first 3 rows in the table). Since the application service on n_4 (L_{n4S1}) is also a neighbor to the forwarding service, we require the neighbor table on n_4 to maintain an extra entry that maps L_{n4S1} to the Process/Thread ID (PID) of the service (the last row). The FIB maintains the next hop(s) for each active LUID.

To forward a message, the forwarding service would first get the next hop(s) for the destination LUID according to the FIB, and then forward the packet either through wireless media or through Inter-Process Communication (IPC) according to the neighbor table. According to Tables III and IV, if the destination of the packet is a service on n_3 (L_{n3S2}) the forwarding service on n_4 would forward it to the forwarding service on n_3 (1st row in FIB) through layer 2 with MAC($n3$) (3rd row in neighbor table). If the destination is the service on its own device, the forwarding service would know $n4S1$ is a direct neighbor (2nd row in FIB) and forward it though IPC (4th row in neighbor table). If the destination is a node not in the domain (*e.g.*, $MF1$), the FIB would have an entry pointing towards the gateway (to $n1F$, 3rd row in FIB) and it will be forwarded to the MAC address of n_1 (first row in neighbor table).

Here, the forwarding service takes care of neighbor discovery and forwarding. The update of the FIB is performed by the routing module (§III-G). An extra TTL field can be added to the table to allow soft-state neighbor and routing management.

With the forwarding and application services separated, one can better focus on either component without interfering with the other.

F. Additional Communication Patterns

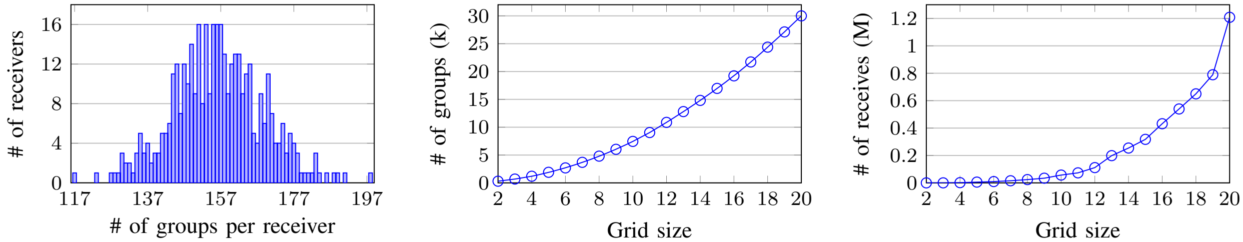
As described in §III-C, MF-IoT can support direct device to device communication (both intra- and inter-domain) and the communication between devices and infrastructure nodes. In this subsection, we describe additional communication patterns that are supported in MF-IoT.

1) Multicast: Since we use service-based GUID's which are independent of any specific node, everyone in the same local IoT domain can listen to the same service GUID. Therefore, multicast can be supported naturally in MF-IoT and we further lump unicast and multicast together and refer to them as a *to-all* service. The forwarding service on the branching point would have more than one entry in the FIB for a GUID if there are more than one receiver. It then replicates the packet and sends a copy to each next hop (either it is on another node or an application in the same node). MF-IoT also takes advantage of the broadcast media all the wireless nodes are using. When the number of next hop nodes is larger than a threshold, a node can broadcast the packet instead of replicating and sending the packet multiple times. The next hop nodes will look up their FIB and discard the packet if no matching entry is found.

2) Anycast: In addition to unicast and multicast, MF-IoT also supports anycast. The listeners in anycast work in the same way as in multicast — they would listen to the same GUID and a tree would be formed by the routing protocol either proactively (*e.g.*, OSPF-like) or reactively (*e.g.*, AODV-like). When sending an anycast packet, the sender would place a different `SVC_TYP` value in the packet header and the intermediate nodes would only forward it to one of the next hop nodes based on its policy (*e.g.*, shortest path, to a node with highest energy level, *etc.*).

3) "Observe" mode: According to [11], the observe mode is important for WSN and IoT applications. In this mode, the observer registers a specific event at a sensor and when the event is detected, the sensor notifies the observer. Usually one registration in the observe mode can get multiple notifications from a single sensor.

The observe mode can also be supported in MF-IoT, and furthermore, we can provide additional mobility handling and



(a) Histogram for a user's group number

(b) The number of groups *vs.* grid size

(c) The number of groups *vs.* grid size

Fig. 7: Intra-domain simulation setup.

multicast support. The observers (either in the same local IoT domain, in the core network or even in different local IoT domains) can listen to the same specific GUID. When an event is triggered, the subject can send the notification to all the receivers through multicast. With the mobility support and an inherent push model, we allow the notifications to be sent in a timely and efficient manner.

G. *Routing*

MF-IoT does not restrict the routing in the network. The application designers can feel free to use any existing routing mechanism or design their own according to the communication pattern they envision. Here, we suggest several mechanisms that we have in mind:

RPL [14] is widely used in the existing IoT systems for home and building automation [15]. The solution builds a tree among the nodes and usually the gateway is seen as the root. It can provide easier management with lower memory and energy consumption thanks to the tree topology. For applications which mostly depend on sensor to gateway and sensor to infrastructure communication, the solution has its benefits since all the traffic has to go through the gateway. MF-IoT can also adopt such kind of routing — RPL algorithm can run as a separate module and modify the FIB of on the forwarding engines. The data plane does not need to be modified.

AODV [16] is used by Zigbee [17] as the default routing. It provides on-demand distance vector routing to accelerate the direct sensor-to-sensor communication (they do not need to go to the root of the tree as RPL). However, to find a path on demand, a request has to be flooded in the whole network which made the solution less efficient when the network is large. AODV also can be used in MF-IoT in small domains for direct communication.

With the advent of Software Defined Networking (SDN), the concept of a central controller eases the traffic engineering and policy enforcement in the network. At the same time, it allows the forwarding engines to be simpler and more efficient. This concept can also be used in IoT world since the gateway usually has more resources, no power constraints and possibly larger radio coverage. The sensors can report the link changes to the gateway and after calculation, the gateway will send the forwarding rules back to the sensor nodes either proactively or on demand. This solution can reduce the amount of flooding in AODV, and support efficient sensor-to-sensor communication compared to RPL. It also has the flexibility to support communication based on policies and resource constraints on the sensors. At the same time, the sensors do not have to calculate the routing and it can save energy in resource constraint nodes. We will use this kind of routing as the default routing for MF-IoT in the evaluation.

IV. Evaluation

To evaluate the performance of MF-IoT, we modified our event-driven simulator that was used in [18], [19], to represent typical IoT usecases. In the evaluation, we compare MF-IoT with IP and another state-of-the-art IoT architecture that is based upon a clean-slate Named Data Network (NDN, see §V) architecture. We will show that compared to IP and NDN, MobilityFirst provides a better support for IoT.

Specifically, We consider three scenarios in the evaluation: intra-IoT domain communication, IoT device mobility, and communication between IoT devices and infrastructure nodes.

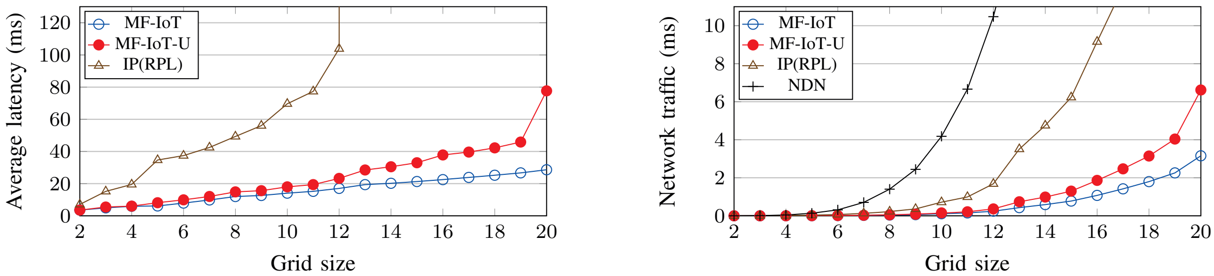
A. *Intra-IoT Domain Device-to-Device Communication*

*1) **Simulation setup**:* We first report the performance of MF-IoT within a local IoT domain. We simulate a domain that has 400 sensor nodes, forming a 20×20 grid, and a gateway (or proxy in the IP case). Each sensor node can communicate with 8 direct neighbors through 802.15.4, with bandwidth of $250kbps$. To perform stress tests, we first generate 30,000 communication groups each containing a sender and 1–50 receivers (therefore the traffic containing both unicast and multicast traffic). The number of receivers per group follows a Zipf distribution with α=0.35. As a result, each node in the network belongs to 117–198 such kind of groups (see Fig. 7a). We also generate a set of messages for each group. The number of messages per group also follows a Zipf distribution but with α=0.81 [20]. The trace then has 138,662 messages and 1,208,203 receive events. The size of the messages varies between 1 and 100 bytes, and follows a normal distribution with $\mathbb{E}$=50. The arrival of the messages follows a Poisson distribution with a mean inter-arrival time of $1/160$ seconds.

We compare the following three networking solutions:

IP: We use UDP over 6Lowpan [2] to represent the state of the art IP-based IoT solutions. According to [21], we use RPL [14] as the routing mechanism. To send a message to multiple receivers, the sender would send multiple unicast messages. If a message cannot fit into a 802.15.4 MTU (127 bytes), it needs to be segmented and re-assembled later.

NDN: NDN [22] uses a query/response model and therefore the receivers of a group have to poll the sender for updates in the group. We choose a polling period of 2 seconds to get a



(a) Average packet delivery latency *vs.* grid size

(b) Aggregate network traffic load *vs.* grid size

Fig. 8: Simulation results for intra-IoT domain device-to-device communication.

1-second average latency, which might be acceptable for many event notification cases but still large for emergent real-time cases. To reduce the average latency, NDN has to poll more frequently and the network traffic would increase rapidly. This is an inevitable trade off in NDN. Unlike IP, it can form a temporal multicast tree if the requests have temporal locality. We place a 10kB Content Store on each sensor node. NDN packet uses TLV format as described in [23] and we do not place signature and signed info in the Data packet to make them feasible to be transmitted in 802.15.4. Similar to IP, if an NDN packet cannot fit into a MAC packet, segmentation and reassembling would happen on the end hosts.

MF-IoT: We consider two variations of MF-IoT in the evaluation — MF-IoT w/o multicast (MF-IoT-U) and full fledged MF-IoT. In MF-IoT-U case, we only use unicast feature in the network and to send a message to multiple receivers, the sender has to send multiple unicasts. We use centralized routing described in §III-G in both variations.

We use the end-to-end latency as well as the aggregate network traffic transmitted by all the nodes as the performance metrics of our evaluation.

2) ***Simulation Results****:* To show each solution's performance trend, we use different grid sizes ranging from 2×2 to 20×20, and plot the number of groups and receive events for each grid size in Fig. 7b and Fig. 7c. The performance results are reported in Fig. 8.

From Fig. 8a, we observe that with larger grids, the average latency for each solution becomes larger since the sender-receiver pairs are farther apart. Among the four solutions, MF-IoT outperforms the other three. Specifically, MF-IoT-U caused minor congestion in the 20×20 grid and the average latency grows by around 15ms. However, the average latency in the IP solution grows even faster since the traffic has to go through the proxy. When the grid size reaches 13×13, the traffic load reaches the capacity limit on the gateway, causing congestion. The average latency goes up to 18.32 seconds eventually. Here, we assume the proxy has an infinite queue so that IP would not drop packets. NDN does not cause serious congestion in the network thanks to its intrinsic flow balance design. However, due to the polling frequency, the average latency remains around 1 second, and when the network grows larger, the average latency goes up by around 100ms partially caused by some minor congestion.

Fig. 8b shows the aggregate network traffic generated by each solution. We observe that MF-IoT and MF-IoT-U generate much less traffic compared to IP and the difference becomes more pronounced when the network size becomes larger. Finally, NDN causes a lot of wasteful traffic due to the polling mechanism.

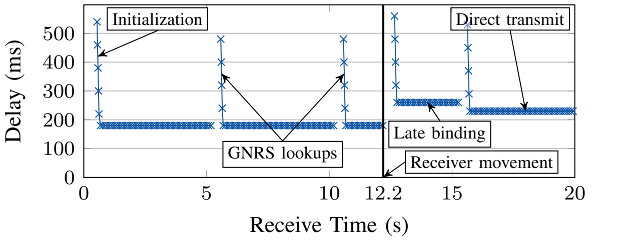


Fig. 9: The observed receiving latency at different simulation times when an embedded node communicates with an infrastructure node (I). I moved to to a different network address at time 12.2s.

To summarize, MF-IoT has achieved lower traffic overhead and average latency compared to other solutions. As a result, we believe that IoT systems that adopt MF-IoT will accomadate higher traffic load and larger network size than the other start-of-the-art solutions.

B. Communication between IoT and Infrastructure Domains

Next, we demonstrate that MF-IoT supports efficient communication between an embedded device and infrastructure, even when the infrastructure node is mobile. We consider the use case that involves a sensor node n trying to send data to an infrastructure node I once every 100ms. We report the latencies observed at I at different times in Fig. 9.

In the start phase of the simulation run, marked as "Initialization", n first requests I's LUID from the gateway, and sends the packet to the gateway. The gateway then conducts a remote GNRS lookup to find the network address for I. The overall latency for this initialization phase is around 550ms.

To deal with node mobility, each GNRS entry cached on the MobilityFirst router would expire after some time. Here, we set the expiration time as 5 seconds. We thus observe the reciever-side latency has spikes each time the cached GUID-to-NA mapping expires and the gateway has to perform remote GNRS lookups (marked as "GNRS lookups").

In the simulation setup, I moves to a different network at 12.2 second and the gateway is not aware of this. Therefore, the gateway would send the packet to the first-hop router towards I's original NA (denoted as R) where R performs

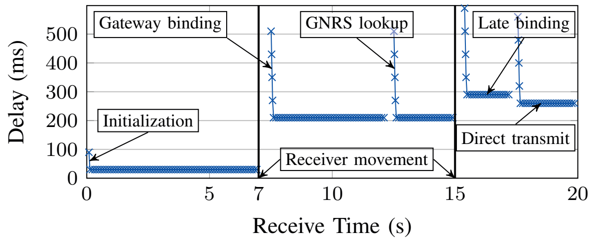


Fig. 10: The observed receiving latency at different simulation times when an embedded node communicates with another embedded node. The receiver moved to to a different IoT domain at time 7s, and moved to a third domain at time 15s.

late-binding. We assume that the movement happens instantly and R can obtain I's new NA through another GNRS lookup. With this additional remote GNRS lookup, the average latency at these times (marked as "Late binding") also increases.

Finally, we note that this entire process is transparent to the sensor node n.

C. Inter-IoT Domain Device-to-Device Communication

Finally, we demonstrate that MF-IoT can efficiently support communication between two embedded devices, even when one of them moves to a different domain. Specifically, we consider two embedded nodes n_1 and n_2, and n_1 sends a packet to n_2 every 100 ms. The receiver node n_2 is within the same local IoT domain as n_1, and it moves to another IoT domain at 7s, and to a third IoT domain at 15s. We plot the latency observed at n_2 at different times in Fig. 10.

In the beginning of the simulation (marked as "Initialization"), n_1 requests n_2's LUID from the gateway, and its latency is rather low since they are in the same domain.

n_2 moves to another domain at time 7. Here, n_1 still sends the MF-IoT packet with n_2's original LUID, but the packet will be forwarded to the gateway as n_2 has left the domain. The gateway then performs a remote GNRS lookup for the NA of n_2's new gateway (marked as "Gateway binding"). From this point on, the gateway has to perform a GNRS lookup every 5 seconds as cached GNRS entries expire every 5 seconds. As a result, the latency increases around here.

When n_2 moves to a third domain at time 15, the gateway in the second domain would perform a late binding (like in the case of the infrastructure node movement), and the first gateway obtains the NA for n_2's latest gateway at time 17 (with a spike in the latency numbers).

This process is entirely transparent to both nodes. Finally, we note that in the two mobility usecases we have considered, it is the receiver that has moved during the communication. MobilityFirst-IoT also works seamlessly if the sender moves to a different domain, wherein the sender simply needs to register the reciever with its new gateway (see §III-C).

V. Related Work

In this section, we discuss related work in state-of-the-art IoT system and architecture design.

A. State of the Art IoT Architectures

Existing work on IoT systems can be broadly classified into two categories: network adaptation for constrained devices and application-layer approaches for resource accessibility and manageability. Network adaptation solutions like 6LoWPAN [2] and ZigBee IP [3] compress the packet header to allow IPv6 (with MTU $\geq$ 1280 bytes) to operate on resource-limited networks like IEEE 802.15.4 (with only 127-byte MTU). However, the tight coupling between identifier and locator in IP makes it difficult for these solutions to provide efficient mobility support.

To deal with resource mobility, studies in the second category seek solution in the application layer. Constrained Application Protocol (CoAP) [4] proposes a specialized web transfer protocol to cope with constrained nodes and networks. It provides a query/response interaction model between endpoints, where resources could be accessed via URIs. State of the art IoT platforms such as IoTivity [5] usually involve generic interfaces to accommodate different lower layer protocols and a centralized server to facilitate efficient resource retrieval. The downside of these overlay approaches is that they usually rely on a server, which is an additional deployment overhead. Also, they are not well suited to support event notification or pushing type of communication pattern.

As a result, we believe that in order to support next-generation IoT systems, we need to consider a network architecture that can naturally support IoT's inherent demands, one that is fundamentally different from IP.

B. Information-Centric Networking and Its Use in IoT

Information-Centric Networking (ICN) is a clean-slate Internet architecture, which is proposed to evolve the network from host-centric to content-centric model where data are identified and accessed by names. Named Data Networking (NDN) [22] (or Content-Centric Network (CCN) [24]) is one of the popular ICN proposals. It uses human-readable, hierarchically-structured Content Names as the identity in the network. NDN provides a query/response communication pattern by using two types of packets in the network: Interest (query) and Data (response). It also introduced a new forwarding engine model having three data structures — Forwarding Information Base (FIB) which maintains the outgoing (inter)faces for each name prefix; Pending Interest Table (PIT) which keeps the unsatisfied Interests and there incoming (inter)faces; and Content Store working as an in-network cache. Data consumers issue Interests with Content Names. The intermediate forwarding engines forward the Interest towards Data provider by according to FIB. Bread crumbs are left on the path via PIT. On receiving an Interest, a node that provides the requested Data (either an intermediate caching router or a Data provider) can reply the Interest. The Data packet travels through the reverse path according to PIT and it consumes the entries in the PITs.

Work by Zhang *et al.* [25] defines several architectural requirement for the IoT including global accessible name and mobility, which indicates ICN has the potential to be used as the underlying network for IoT since it integrates named-based routing, compute, and caching/storage as part of the network. To adapt to the resource-constraint devices, CCNLite [23] is proposed as a lightweight implementation of NDN which simplifies the original code base and data structure. In order to support multi-source data retrieval, work in [26] proposes a framework that multiple data producer can answer to the same Interest packet. However, the similar to NDN, these solutions only focus on data retrieval, and it is difficult for them to achieve functions like event notification and multicast in IoT. Moreover, NDN requires the Data packet to be transmitted on the reverse path as the Interest, it causes difficulties in IoT where links might be asymmetric. The need for PIT and Content Store also puts burden on storage-constraint devices.

Work in [27] proposes a generic IoT middleware architecture based on NDN and MobilityFirst to support basic IoT function such as service discovery and naming service. These functions can also be used in MF-IoT in the application layer and is orthogonal to the design in this paper.

VI. Conclusion

In this paper, we propose MF-IoT, a generic network architecture that satisfies the requirements placed by the emerging IoT systems, namely, global reach-ability, mobility, communication diversity, and resource efficiency. We achieve this goal by creating a network-layer dialect (Local Unique IDentifier, LUID) in a local IoT domain and adopt a gateway to efficently translate between GUID's that are used in the core network and the corresponding LUID's. The translation is transparent to the applications so that MF-IoT can have efficient global reach-ability and mobility support. With service-based GUID assignment, we further enable seameless service migration, service caching and the separation between application logic and network functions. Our simulation results show that MF-IoT can greatly improve the performance of IP-based and NDN-based solutions.

References

[1] "The Internet of Things — How the Next Evolution of the Internet Is Changing Everything," https://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, Apr. 2011.

[2] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), Internet Engineering Task Force, Sep. 2007, updated by RFCs 6282, 6775. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[3] Z. Alliance, "ZigBee IP Specification," *ZigBee 095023r10, Work in Progress, July*, 2010.

[4] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252 (Proposed Standard), Internet Engineering Task Force, Jun. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7252.txt

[5] "Iotivity," https://www.iotivity.org/.

[6] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: A Robust and Trustworthy Mobility-Centric Architecture for The Future Internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[7] "Local and Metropolitan Area Networks Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE Std 802.15.4-2003*, pp. 0_1–670, 2003.

[8] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "GSTAR: Generalized Storage-Aware Routing for Mobilityfirst in the Future Mobile Internet," in *Proceedings of the sixth international workshop on MobiArch*. ACM, 2011, pp. 19–24.

[9] J. Moy, "OSPF Version 2," RFC 2328 (INTERNET STANDARD), Internet Engineering Task Force, Apr. 1998, updated by RFCs 5709, 6549, 6845, 6860, 7474. [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt

[10] "Alljoyn framework," https://allseenalliance.org/framework.

[11] K. Hartke, "Observing Resources in the Constrained Application Protocol (CoAP)," RFC 7641 (Proposed Standard), Internet Engineering Task Force, Sep. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7641.txt

[12] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "DMap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet," in *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, 2012, pp. 698–707.

[13] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A Global Name Service for A Highly Mobile Internetwork," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 247–258.

[14] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[15] J. Martocci, P. D. Mil, N. Riou, and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks," RFC 5867 (Informational), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5867.txt

[16] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561 (Experimental), Internet Engineering Task Force, Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[17] Z. Alliance, "ZigBee 2007 Specification," *Online: http://www. zigbee. org/Specifications/ZigBee/Overview. aspx*, vol. 45, p. 120, 2007.

[18] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "COPSS: An Efficient Content Oriented Publish/Subscribe System," in *Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2011, pp. 99–110.

[19] M. Arumaithurai, J. Chen, E. Monticelli, X. Fu, and K. Ramakrishnan, "Exploiting ICN for Flexible Management of Software-Defined Networks," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 107–116.

[20] L. A. Adamic and B. A. Huberman, "Zipfs law and the Internet," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[21] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A Survey on the IETF Protocol Suite for the Internet of Things: Standards, Challenges, and Opportunities," *Wireless Communications*, vol. 20, no. 6, pp. 91–98, 2013.

[22] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named Data Networking (NDN) Project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[23] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information Centric Networking in the IoT: Experiments with NDN in the Wild," in *Proceedings of the 1st International Conference on Information-centric Networking*, ser. ICN '14. ACM, 2014, pp. 77–86.

[24] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking Named Content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNext)*. ACM, 2009, pp. 1–12.

[25] Y. Zhang, D. Raychadhuri, R. Ravindran, and G. Wang, "ICN based Architecture for IoT," IETF Internet Draft draft-zhang-iot-icn-architecture-00. IRTF, Tech. Rep., 2013.

[26] M. Amadeo, C. Campolo, and A. Molinaro, "Multi-source Data Retrieval in IoT via Named Data Networking," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 67–76.

[27] S. Li, Y. Zhang, D. Raychaudhuri, R. Ravindran, Q. Zheng, and L. Dong, "IoT Middleware Architecture over Information-Centric Network," in *Globecom Workshop on Information Centric Networking Solutions for Real World Applications (ICNSRA)*. IEEE, 2015, pp. 19–24.