# IoT Middleware over Information-Centric Network

Sugang Li, Yanyong Zhang, Dipankar Raychadhuri
Ravi Ravindran, Qingji Zheng, Guoqiang Wang
WINLAB, Rutgers University, North Brunswick, NJ, USA
Huawei Technologies, Santa Clara, CA,USA

## ABSTRACT

hello


## 1. INTRODUCTION

For many years, many standalone Internet of Things (IoT) platforms have been developed and deployed in different domains. The recent trend, however, is to evolve towards a globally unified IoT platform, in which billions of objects connect to the Internet, available for interactions among themselves, as well as interactions with many different applications across boundaries of administration and domains.

Building a unified IoT platform, however, poses a set of unique challenge on the underlying network and systems.Firstly, it needs to support a large number of networked objects - Cisco predicts there will be around 50 Billion IoT devices (sensors, RFID tags, actuators, etc) on the Internet by 2020 [**?**]-and many of these object are mobile, for smoothly with respect to metrics like response time, throughput, resolution and routing scalability. Secondly, IoT devices will have heterogeneous means of connecting to the Internet, and often have severe resource constraints, e.g., constrained resources in power, computing, storage, bandwidth. Thirdly, interactions between the applications and objects are often private, contextual, real-time and dynamic, requiring strong security and privacy protections.

### 1.1 ICN IoT Middleware

Current approaches towards a unified IoT platform are mostly based upon Internet overlays, whose inherent inefficiencies hinders the platform from satisfying the challenge outlined earlier. In recent years, in order to address the inefficiencies of today's Internet, Information-Centric Network has been proposed. ICN identifies a network object (including a mobile device, content, or service) by application centric name instead of its IP address, and adopts a hybrid name/address routing, lending itself to supporting the unified IoT platform. In this paper, we propose to build a IoT middleware for various ICN architecture.Shown as Figure 1, only publishing/subscribing management is deployed in a centralized server, while device/service discovery, context processing and naming service are pushed down to the distributed middleware component(discussed in Section **??**). While most of the funtionalities are in common, we leave the interface to the network being flexible so that the middleware can adopt different ICN protocols. Specially, we will discuss these funtionalities in the hierarchical-name-based NDN and flat-name-based Mobilityfirst.

### 1.2 NDN Background

NDN provides a receiver-driven architecture by transmitting with two types of packets : Data and Interest. Data Consumer issues Interest of the hierarchical content name and forward to Data Producer. When the interest arrives at the node that own this piece of content, Data Producer will reply with a Data Packet along the reverse path to the

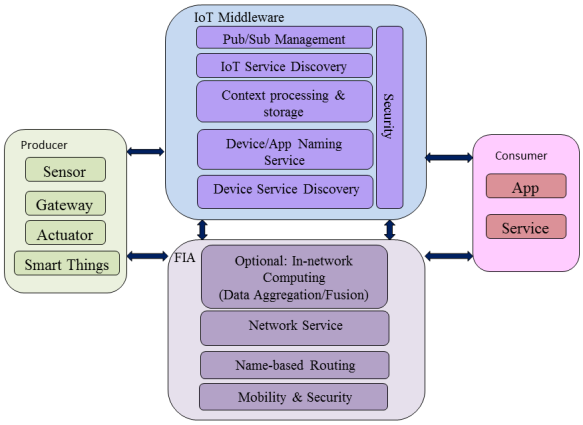


**Figure 1: ICN-IoT middleware functionality break down.**

Data Consumer. NDN provides three types of key component to support such functionality. Forwarding Information Base (FIB) collects the forwarding information at each NDN node. The content prefix and out-face mapping is recorded in the FIB. Once a certain face receive Data packet for a Pending Interest Table , it could be added into the FIB to indicate a healthy data retrieving path.Pending Interest Table (PIT) is a data structure that maintain the set of pending Interest routed by the node that expecting Data packet in return. PIT entry is created by receiving Interest, and filters the redundant Interest that carried the same content name. Due to the limitation of PIT size, every PIT entry comes with a timeout which is estimated based on a round trip time. Once a matching Data packet being received by the node, it is forwarded via the face recorded in the PIT before this entry being removed. Content Store(CS) is a data cache for storing Data packet for matching Interest along the reverse path. Also, due to the constraint of local CS size, Data Producer can define the freshness of the Data packet in order to timeout it in the cache. With such transmission mechanism, NDN eliminates the notion of source and destination as it is in IP network, and handle routing only for Interest Packet.

### 1.3 MobilityFirst Background

MF utilizes GUID to name every network object, while separating this GUID from its actual network address. This identifier(GUID)/locator(NA) split design allows MF supporting dynamic address binding, multiple addressing binding and late binding. Shown as Figure **??**, MF core network architecture includes the following network component.

**Global Name Resolution Service(GNRS)**: GNRS is a centralized service that maintains mappings between GUID and network address. MF routers create the entries by performing an Insert for the GUIDs of attached network devices and the associated network address, and query GNRS for a translation from GUID to latest binding network address. Recent

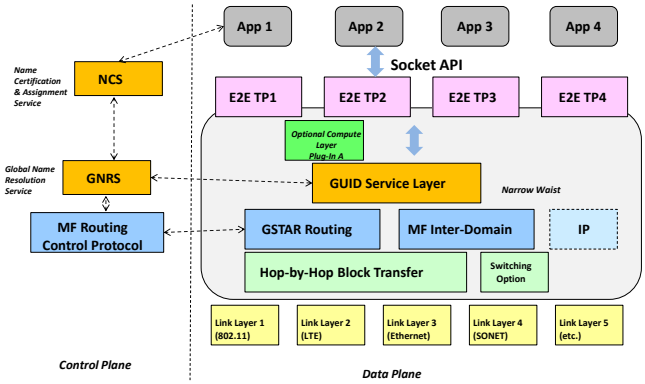


**Figure 2: Mobilityfirst architecture.**

works shows that this translation performance is much(50-100 ms) than DNS resolution [**?**].

**Hybrid GUID/NA address routing**: Each of the router in MF can make routing decision based on NA or GUID in the header of data packet, since routing decision are made on a hop-by-hop manor [**?**].

**Delay-Tolerant Network(DTN)**: The storage in each MF router provides the capability of caching the data packe. Hence, data can be hold or forwarded based on different routing decision.

## 1.4 MF Multicast

MF multicast is based on the idea of group GUID that gathers multiple network objects into one entity. Group GUID to object GUID mapping is a one-to-many mapping that being maintained at GNRS server. Network object can claim to join the multicast group, and either edge router or a centralized management service will perform insertion of the object GUID to group GUID mapping to into GNRS server. When the first router queries GNRS server for a group GUID mapping, GNRS server returns a list of member GUIDs. The router assembles these GUIDs into the header and performs $LongestCommonPath$ algorithm to determine the next hop address.Shown as Figure 3, routers look up the routing table for these GUIDs, and check if there are multiple next hops. When the number of result is more than one, the router reassembles the header and forwards the copied packet to corresponding interfaces.

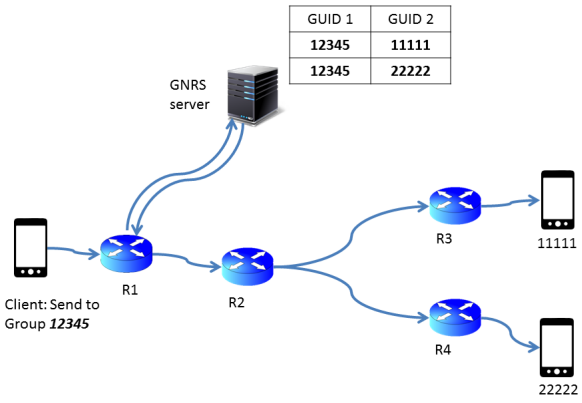


**Figure 3: MF multicast example: Client**

## 2. RELATED WORK

ICN is a promising architecture for a general purpose future Internet, including Internet of Things. Previous works by Zhang Y. et al [**?**] and Li S. et al [**?**] states the benefits that ICN network is capable to bring to IoT, including efficient data retrieval, well support of mobility, naming, scalability and security.

Some early work have demonstrated the feasibility of ICN on particular IoT system for both public space and home automation.

Another study by Amadeo et al. [**?**] provides a solution on efficient multi-source data retrieval. It proposes a method to use one Interest to retrieve multiple data with different suffixes on different location by adjusting the traditional NDN protocol to support "prefix-based" interest. For example, a subscriber is interested in the resource named "/office/temperature", which is the information from multiple sensor deployed in his office. At the same time, temperature sensors should advertise their resource with this prefix and their own suffix(e.g sensor in the conference room should be named "/office/temperature/conferenceroom"). Hence, the subscriber can issue single interest to retrieve multiple resource.

George et al. [**?**] propose a Publish-Subscribe Internetworking ICN architecture that that identifier can be used to represent a thing, an application, a group of similar, or any contextual specific entity in the network. A Rendezvous Node functions as a middle box where advertisement from publisher and subscription request from subscriber can meet up to form a membership.

Many study take one step further to implement IoT system over ICN architecture. In the study by Shang et al. [**?**], a practical use case that integrates NDN with Building Management System (BMS) is represented. It shows that human-readable hierarchical naming scheme brings convenience in configuring and managing large number of BACnet devices. Baccelli et al. [**?**] report a CCN experiment over a life-size IoT system. They port light-weight CCN-lite code with the RIOT operating system. Based on this platform, performance of various routing protocol and impact of caching have been measured.

## 3. ICN IOT MIDDLEWARE SYSTEM DESIGN

In this section, we will propose a overall system design formed by three distributed physical components. They cover four principle functionality to develop IoT middleware over ICN network. In legacy IP-based IoT platform, these functionality are almost implemented on a centralized server, or over-lay application. In our ICN-IoT middleware architecture, they are pushed down to distributed components which can form a subsystem to provide local service as well as a complete system that enables global communication.

### 3.1 Physical Components

- *Aggregator*: Aggregator is the base-line component in our middlware architecture that interconnects the various IoT service in the local network. Aggregator usually acts as gateway for resource-constraint wireless sensors, or it can be a device that integrates sensing/actuating service.
- *Local Service Gateway*:
- *IoT Server*:

### 3.2 Device Discovery

The objective of device discovery is to ***. Device discovery is a key component of any IoT system, which can be considerably simplified by the ICN network. In today's sensor networks and IoT systems, the device discovery module mainly focuses on the reach-ability of a device, instead of device type and service type [**?**]. This method becomes inconvenient when a large number of sensors are deployed in the field and the developer needs to configure each device manually. Furthermore, a translation service is required to maintain the mapping from network addresses to names. In contrast, ICN uses names to discover new devices, without involving manual configuration or name translation. Below, we explain the ICN-IoT device discovery process in detail, including both devices that are able to run full-stack protocols (referred to as *resource-rich sensors*) and devices that are unable to do so (referred to as *resource-constrained sensors*).

**Resource-rich sensors**: Many a resource-rich sensor comes with manufacture secure ID and model name, which need to be exposed to the aggregator and LSG to facilitate device discovery. This objective is achieved in different ways by NDN and MF. In NDN, this process is initiated by the configuration service running on LSG, which periodically broadcasts discovery Interests (using the name $/iot/model$). The new sensor replies to the discovery interest with its information, and the configuration service then registers the sensor and generates a local ICN name for the sensor. In MF, we can set the model number, $group - GUID$, as the destination address. ***YZ: next? ***On receiving of the request, the new device replies with the manufacture secure ID, and the configuration service register and generates a local ICN name for the new device.

**Resource-constrained sensors**: Many resource-constrained sensors connect to the Internet using the IEEE 802.15.4 technology via an aggregator that acts as a border router [**?**]. These sensors voluntarily discover each other and establish forwarding paths to the aggregator, self-organizing into a mesh network. Neighbor discovery is achieved differently in NDN and MF. In NDN, when a new sensor arrives, it broadcasts a neighbor discovery message to connect with neighbors. After establishing connectivity with neighboring sensors, it broadcasts an Interest named $/ndn/aggregatorservice$ to discover the Aggregator. In MF, neighbor discovery is naturally supported, and the new sensor simply needs to send a discovery request to a well-known broadcast GUID to discover the Aggregator. The above process is shown as Figure **??**.

### 3.3 IoT Service Discovery

IoT services can be generally categorized into three classes: sensing, actuating, and control. ***YZ: what is the diff between actuating and control? *** In today's IoT platforms, sensors are connected via a server, which requires considerable development effort such as maintaining a local name to IP mapping. In ICN-IoT, IoT service discovery and configuration becomes more efficient. Specifically, we consider two service discovery modes: $peer - to - peer$ and $master - slave$.

**Peer-to-peer Service Discovery**: In this mode, we consider the scenario in which an aggregator (referred to as the source aggregator) tries to discover the IoT services provided by another peer aggregator (referred to as destination aggregator). In NDN, the source aggregator broadcasts an interest using the well-known name $/area/servicename/certificate$, which will eventually reach the destination aggregator. In MF, this is handled by multicast (discussed in Section **??**) instead of flooding the entire local network. All sensors that provide relevant services can join the multicast group that is identified by a $Group - GUID$. After establishing the multicast group, the source aggregator sends a request containing the service name and certificate to the multicast group.The target aggregator that hosting this service checks the certificate and registers the source smart thermostat if there is a matched service. It replies with a acknowledgement containing its certificate to the source aggregator. The message flow is shown as Figure **??**

In an example of NDN smart home, a thermostat expresses a request to discover a AC service using well-known name $/home/ac/certificate$ via broadcast channel. In MF case, a multicast group GUID $1234$ can be assigned to all home appliance IoT service. The thermostat sends request containing the service name and certificate to $1234$. In both cases, the AC that hosting this services replies with acknowledgement if all condition match.

**Master-slave Mode**:In some IoT applications, there are more

than one sensing service or actuating service connecting to one control service. A source aggregator hosting control service expresses a request with servic name and its certificate to discover a service from all the available aggregators. The target aggregators verify the certificate and register the source aggregator if there is a matched service.If two condition are satisfied, they answer with acknowledgement containing their certificates.

In the AC control NDN example shown in Figure **??**, the number of AC service provider increases to three, which can be named as $/office/ac/1$, $/office/ac/2$ and $/office/ac/3$ . The thermostat expresses a partial name $/office/ac/certificate$ via a broadcast channel. In the MF case shown in Figure **??**, a multicast group GUID $1234$ can be assigned to identify AC service. The thermostat sends request with service name and certificate to $1234$. The target ACs reply with their certificate if the request meets all criteria.

## 3.4 Device/Application Naming Service

name based on location/service type local naming, aggregator expose short name 802.15.4 secure name MF secure GUID

## 3.5 Publish/Subscribe Management

Data Publish/Subscribe (Pub/Sub) is an important function for ICN-IoT, response for resource sharing and management. In conventional IP network, most of the IoT platforms provide a centralized server to aggregate all IoT service and data. While resource is being published as service, centralized architecture ensures the availability, but it poorly supports scalability and high bandwidth consumption due to high volume of control and data exchange. Thus, we consider a decentralized pub/sub model in ICN-IoT middleware. Specifically, we will discuss rendezvous mode and a data-control separated mode.

**Rendezvous Mode**:Rendezvous mode is classical pub/sub scheme in which data and request meet at a intermediate node. NDN is a Pull-based architecture, where the Pub/Sub is not naturally supported , but COPSS [**?**] proposes a solution for such requirement. It integrates a push based multicast feature with the pull based NDN architecture at the network layer by introducing Rendezvous Node(RN). RN is logical entity that reside in NDN node. The publisher first forwards a Content Descriptor (CD) as a snapshot to the RN. RN maintains a subscription table, and receives the Subscription message from subscriber. The data publisher just sends the content using Publish packet by looking up FIB instead of PIT. If the same content root is required by multiple subscribers. RN will deliver one copy of content downstream, hence reduced bandwidth consumption.

**Data-control separated Mode**:Compared with Rendezvous mode in which data plane and control plane both reside on the same ICN network layer, we consider a architecture that centralized server takes over the control message while data is handled by ICN network layer. Following the naming process mentioned above, the LSG has the ICN name for the local resource which is available for publishing on IoT server. IoT server maintains the subscription membership, and receives subscription request from subscriber. Due to the precise number and identify of the resource provider is unknown for the subscriber in a dynamic scenario, IoT server also takes responsibility of grouping and assigning group name for the resource. In NDN, the grouping resource is rather straight forward : all resource have the same schematic prefix will be grouped together, and the prefix can be used to identify the service. The traditional NDN dose not support using a a common partial name to retrieve multiple resource, but M. Amadeo et al. [**?**] provide solution that resolves this issue by introducing long-life multi-source Interest in PIT. MF takes advantage of $group - GUID$ to identify a service formed by multiple resource. This $group - GUID$ will be distributed to the subscriber as well as the publisher. Figure **??** shows the example that NDN uses the common prefix$/home/monitoring/$ to identify a group of resource that provides multiple monitoring service such as $/home/monitoring/temperature$ and $/home/monitoring/light$. The subscriber retrieves the prefix from IoT server, and sends Interest toward the resource. In Figure **??**, $GUID-1$ identifies the "home temperature monitoring" service with $GUID-2$ and "home light status" service with $GUID-3$. The resource producers, i.e. the temperature is notified that its service belongs to $GUID-1$, then listens on $GUID-1$. The subscriber sends request toward $GUID-1$, and request will be multicasted to the producers at the last common node. On receiving the request, the resource producer unicasts the data to the subscriber. Moreover, if multiple resource consumers subscribe to the same resource, the idea of $group - GUID$ can be reused here to group the consumer to further save bandwidth using multicast.

# 4. EVALUATION

We have conducted a detailed experiment of service discovery in NS-3 simulator.

## 4.1 Service Discovery

## 4.2 Multi-source Retrieval

## 5. CONCLUSION

Conclusion