IoT Middleware Architecture over Information-Centric Network

Sugang Li*, Yanyong Zhang*, Dipankar Raychaudhuri*, Ravishankar Ravindran†, Qingji Zheng†, Lijun Dong†, Guoqiang Wang†
*WINLAB, Rutgers University, North Brunswick, NJ, USA
†Huawei Research Center, Santa Clara, CA, USA

Abstract—**Many approaches have been proposed to build a unified IoT platform where physical and digital objects are accessible by applications crossing different organization and domains, and are based on IP-overlay architecture. These solutions inherit the constraints of the current internet, especially in terms of naming, heterogeneity, mobility and security. In this paper, we propose a new Information-Centric Network (ICN) based IoT middleware to address these challenges by leveraging various promising features of ICN, such as naming. We elaborate the functions of ICN-based IoT middleware by integrating with the two future internet architectures, namely Named-Data Networking and MobilityFirst. Moreover, we evaluate the efficiency of service discovery (one of the functions in the proposed ICN-based IoT middleware) and demonstrate the feasibility of the proposed ICN-based IoT middleware .**

I. Introduction

Many standalone Internet of Things (IoT) platforms have been developed and deployed in different domains in the past. The recent trend, however, is to evolve towards a globally unified IoT platform, in which billions of objects can connect to the Internet, interact with each other and inter-operate with many different applications across the boundaries of organization and domains.

Building a unified IoT platform, however, poses a set of unique challenges and requirements on the underlying network and systems.

- **Identity**: To realize a unified IoT platform, it needs to assign names (or IDs) that are unique and unified within the scope and lifetime of each device, data items generated by these devices, or a group of devices towards a common objective. Currently, IoT systems have multiple vertical stacks with their own identity mechanisms that do not inter-operate with one another.
- **Heterogeneity**: IoT devices will have heterogeneous means of connecting to its application environment, and often have resource constraints, e.g., constrained resources in power, computing, storage and bandwidth.
- **Mobility**: In some scenarios, the data producer is mobile and unable to provide reliable connection with data consumers. Thus, in presence of mobility, a unified IoT platform requires the system to deliver IoT data within delays that are acceptable to applications.
- **Security**: The heterogeneity and openness of the IoT environment imposes great attack surface towards IoT devices and services. Without carefully considering security concerns such as identity authentication, data integrity and privacy, any IoT platform is hard to be adopted by either end-users or enterprises.

A. ICN-IoT Middleware

Current approaches towards a unified IoT platform are mostly based upon Internet overlays, whose inherent inefficiencies hinder the platform from satisfying the challenges outlined earlier. In recent years, in order to address the inefficiencies of today's Internet, Information-Centric Network (ICN) has been proposed. ICN identifies a network object (including a mobile device, content, or service) by an application-centric name instead of its IP address, and adopts the name-based routing In this paper, we propose a unified IoT middleware, referred to as *ICN-IoT* middleware, which can support several specific ICN architectures. As shown in Figure 1, the ICN-IoT middleware mainly consists of the following modules: publish/subscribe management, device/service discovery, context processing, and naming service. Among these modules, the publish/subscribe management is a centralized service while others are implemented in a distributed manner (discussed in Section III-A). Finally, the security component is implemented across all services.

It is worth noting that functions in the proposed middleware are required to comply with the IoT deployment requirements and protocol agnostic, and therefore the instantiation of the ICN-IoT middleware may differ depending on the underlying ICN protocol. To be specific, this paper discusses the IoT functions with respect to the Named-Data Networking and MobilityFirst.

B. Name Data Networking (NDN)

NDN [13] provides a receiver-driven architecture by transmitting two types of packets : Interest and Data. Consumer issues Interest of the hierarchical content name and forwards it to Data Producer. When the Interest arrives at some node (either intermediate router or Data producer) that owns the requested content, the node will reply the requested content along the reverse path to the Data Consumer. NDN forwarding is supported by three data structures: Forwarding Information Base (FIB) maintains the forwarding information at each NDN

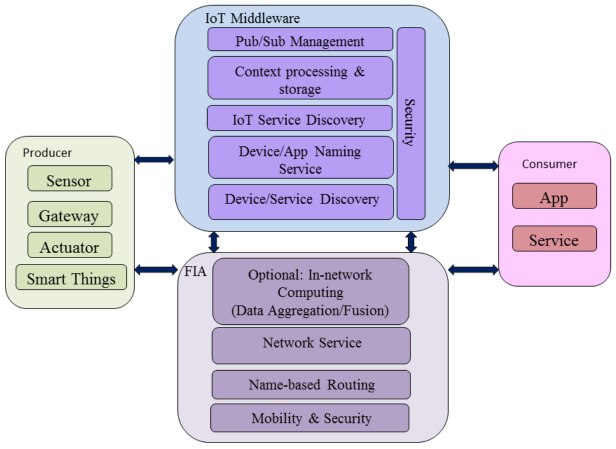

Fig. 1: ICN-IoT middleware functionality break down.

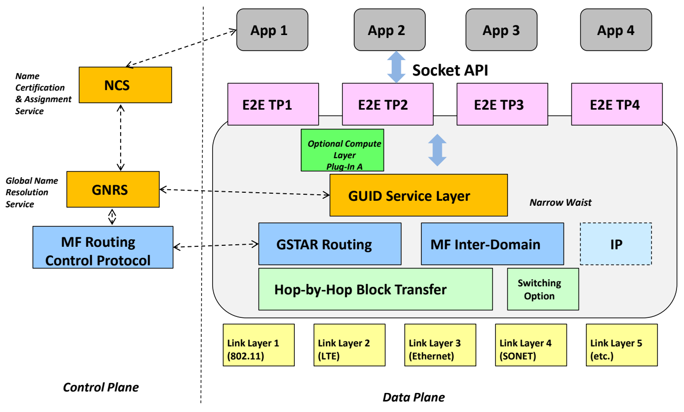

Fig. 2: Mobilityfirst architecture.

node. The content prefix and out-face mapping is recorded in the FIB. Once a certain face receives Data packet for a pending Interest, it could be added into the FIB to indicate a healthy data retrieving path. Pending Interest Table (PIT) is a data structure that maintains the set of pending Interest routed by the node expecting Data packet in return. PIT entry is created by receiving Interest, Due to the limit size of PIT, every PIT entry is associated with a timeout which is defined or estimated by the consumer based on the round trip time. Once a matching Data packet is received, the NDN node forwards it through 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. In addition, due to the limit size of local CS, Data producer can define the freshness of the Data packet in order to timeout it in the CS. With such mechanism, NDN decouples the content from its original location without location binding.

C. *MobilityFirst (MF)*

MF [10] utilizes GUID to name every network object, while separating this GUID from its actual network address. The separation of identifier(GUID) and its locator(network address) allows MF to support dynamic address binding, multiple addressing binding and late binding. As shown as Figure 2, MF core network architecture includes the following network components.

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 the latest binding network address. Recent work shows that this translation performance is much better (50-100 ms delay) than DNS resolution [12].

Hybrid GUID/NA address routing: Each MF router can make routing decision based on NA or GUID in the header of data packet, since routing decision are made by a hop-by-hop manner [6].

Delay-Tolerant Network(DTN): Each MF router is able to cache data packets in its storage. By doing this, data can be stored or forwarded depending on different routing policies, such as quality of the link which is useful over wireless interface.

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 "Longest Common Path(LCP)" 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 the results is more than one, the router reassembles the header and forwards the copied packet to corresponding interfaces. If the number of multicast group members becomes significant, another approach can be adopted to resolve this issue–instead of assembling all GUIDs into the header after one query, each router on the path queries GNRS server for the group members and performs LCP to decide next hop.

II. Related Work

ICN is a clean-slate architecture whose objectives align with the networking and application requirements of Internet of Things. Previous studies by Zhang Y. et al [14] and Li S. et al [5] have pointed out that the ICN network is able to bring great benefits for IoT applications, including efficient data retrieval, and good support of mobility, naming, scalability and security. Driven by the same vision, other studies also have demonstrated the feasibility of ICN on supporting specific IoT applications. For example, the study by Amadeo et al. [1]

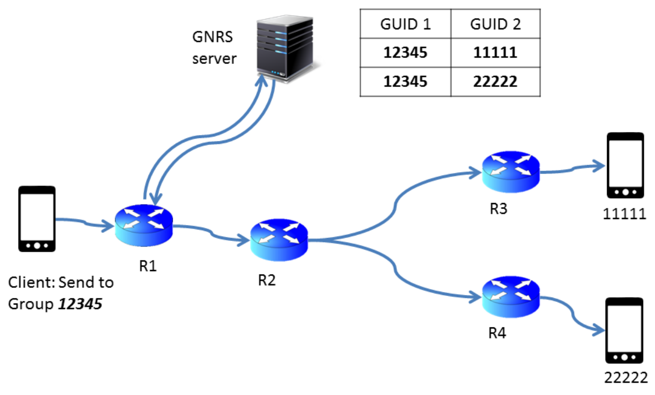

Fig. 3: MF multicast example: A client sends a message to Group 12345

provides a solution for efficient multi-source data retrieval, by using a single Interest packet to retrieve multiple data with different suffixes from different locations, which greatly enhances the traditional NDN protocol to support "prefix-based" Interest. Suppose a subscriber is interested in the resource named "/office/temperature", which is to gather information from multiple sensors deployed in his office.Meanwhile, temperature sensors can advertise their resource with this prefix and their own suffix (e.g sensor in the conference room should be named "/office/temperature/conferenceroom"). To this end, the subscriber can issue a single Interest to retrieve multiple resources simultaneously.

George et al. [8] proposes a Publish-Subscribe Internetworking ICN architecture where identifiers can be used to represent a thing, an application, a group of similar things, or any contextual specific entity in the network. A Rendezvous Node functions as a middle box where advertisements from publishers and subscription requests from subscribers can meet to form a membership.

There are also some studies that implement IoT systems over the ICN architecture. The study by Shang et al. [11], shows a practical use case of integrating NDN with a Building Management System (BMS). It shows that human-readable hierarchical naming scheme can bring great convenience in configuring and managing a large number of BACnet devices. Baccelli et al. [2] reports a CCN experiment over a deployed IoT system, in which they ported light-weight CCN-lite code over RIOT operating system. Based on this platform, the authors measured the performance of various routing protocols and the usefullness of caching.

III. ICN-IoT Middleware Design

In this section, we present the overall ICN-IoT middleware design. It consists of three system components, as shown in Figure 4 – Embedded Device, Aggregator, Local Service Gateway, and IoT server, which offer five principle middleware functionalities, including device discovery, device naming service, IoT service discovery and Pub/Sub management. In contrast to centralized or overlay-based implementation in the legacy IP-based IoT platform, our ICN-IoT architecture

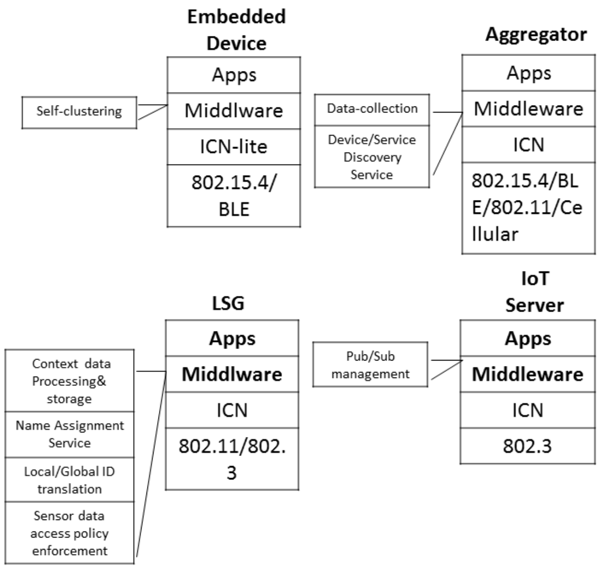

Fig. 4: System Architecture of ICN-IoT middleware

pushes these middelware functionalities down to the distributed components to enable self-configuring subsystem to provide not only local services but can also scale to large IoT service.

A. System Architecture

The proposed architecture has the following four main system components:

- *Embedded Device*, which enables seamless ICN protocol in resource-constraint network. An ICN embedded device supports at least one of following communication mode: transmitting and routing over the data to the Aggregator, or responding the request based on name or ID.
- *Aggregator*, which interconnects various IoT services and content in a local network. An Aggregator usually plays two roles: one is to act as the gateway to bridge the communication between resource-constrained wireless sensors and the rest of the nodes in the local network, and the other one is to integrate sensing/actuating services in the local network.
- *Local Service Gateway (LSG)*, which connects the local IoT system to the rest of the global IoT system, and handles local name assignment and enforces data access policies for local IoT devices. In addition, it can run context processing services to publish only the contextual information (instead of raw data) to the IoT server.
- *IoT Server*, which is a centralized server that maintains subscription memberships and provides the lookup service for subscribers. Unlike legacy IoT servers that are involved in the data path from publishers to subscribers – raising the concern of its interfaces being a bottleneck – the IoT server in our architecture is only involved in the control path where publishers and subscribers exchange their names and certificates.

In the following subsections, we will discuss the functionality for each system component (as shown in Figure 4) and present detailed protocol design considering NDN and MF.

B. Device and Network Service Discovery

Device discovery is a key component of any IoT system. The objective of device discovery is to expose new devices to the rest of the IoT system – every entity should be exposed to its direct upstream device and possibly other devices. Specifically, it includes the following three aspects: (1) a newly-added sensor should be exposed to its aggregator, and possibly to its LSG and the IoT server; (2) a newly-added aggregator is exposed to its LSG, and possibly to its neighbor aggregators; and (3) a newly-added LSG should be exposed to the IoT server. We note that device discovery could be used in other contexts, such as neighboring sensors discovering each other to form routing paths, but in this draft, we use the term to specifically mean discovering new devices for IoT middleware purpose.

During device discovery for newly-added sensors, the sensor passes its device-level information (such as manufacture ID and model number) and application-level information (such as service type and data type) to the upstream devices. If the sensor is to have an ICN name, the name is assigned by the naming service (described in Section III-E), and recorded by both the LSG and the aggregator (and possibly the IoT server).

ICN enables flexible and context-centric device discovery which is important in IoT ecosystem where heterogeneous IoT systems belonging to different IoT services may co-exist. Contextualization is a result of name-based networking where different IoT services can agree on unique multicast names that can be pre-provisioned in end devices and the network infrastructure using the routing control plane. This also has an advantage of localizing device discovery to regions of network relevant to an ICN service, also enabling certain level of IoT asset security. In contrast IP offers no such natural IoT service mapping; any forced mapping of this manner will entail high configuration cost both in terms of device configuration, and network control and forwarding overhead.

In the device discovery phase, sensors expose their information, such as its manufacture secure ID and model name, to the upstream aggregators pulling information from sensors. There are two ways of achieving this aim: (1) sensors pushing the information towards the aggregators, and (2) aggregators pulling the information from sensors. In both NDN and MF, the pulling method is used. 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 group-GUID as the destination address, and the configuration service issues a request via multicasting. When receiving such request, the new device replies with the manufacture secure ID, and the configuration service registers the device and generates a local ICN name for it.

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 group-GUID as the destination address, and the configuration service issues a request via multicasting. When receiving such request, the new device replies with the manufacture secure ID, and the configuration service registers the device and generates a local ICN name for it.

The network service discovery for IoT infrastructure service like naming, gateway, or context-processing service is hosted on LSGs or Aggregators. These devices periodically broadcast their services, which will be responded to by sensors that need these services. Please note that only those sensors that need naming service or context processing service will respond. The detailed process is very similar to that involved in the device discovery (which is described above).

C. Service Discovery

Service discovery intends to learn IoT services that are hosted by one aggregator by its neighbor aggregators. The requirements include low protocol overhead (including low latency and low control message count), and discovery accuracy.

In today's IoT platforms, sensors, aggregators and LSGs are connected via IP multicast, which involves complicated group management and multicast name to IP translation service. Multicast, however, is greatly simplified in ICN as most ICN architectures have natural support for multicast.

Below, we explain how service discovery is implemented. The key to service discovery is to expose aggregator's services to its neighbor aggregators. How this is implemented differs in NDN and MF. In NDN, the source aggregator broadcasts an interest using the well-known name $/area/servicename/certificate$, which will eventually reach the destination aggregator. NDN's Interest/Data mechanisms allows only one response for each Interest send while discovery requires to learn multiple entities, hence efficient discovery is realized using exclusion via Selectors in the protocol or as an overlay protocol [9].

In MF, after establishing the multicast group, the source aggregator sends a request containing the service name and certificate to the multicast group. The destination aggregator that hosts the service checks the certificate and registers the source Aggregator if there is a matched service. It replies with an acknowledgment containing certificate to the source aggregator.

For secure service discovery, a secured name needs to assigned to the service host. Especially in MF IoT, secured group GUID is utilized to realize service request multicast, which may be owned by multiple hosts, hence conventional public/private key scheme may not be suitable for this case. As an alternative, group key management protocol (GKMP) [4] can be adopted to resolved the issue above – A naming service residing at LSG or IoT server (depending on application scope) generates a group public key that used as group GUID for a service, then this group public/private keys pair is assigned

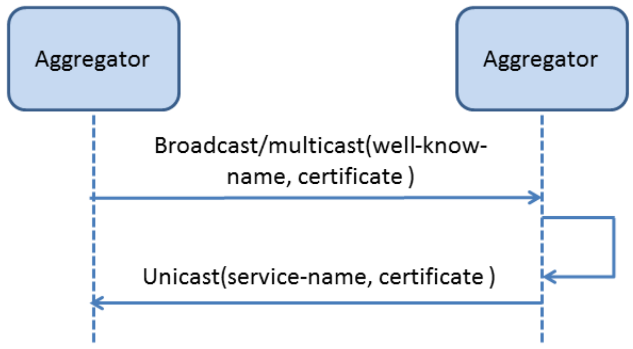
Aggregator
Aggregator
Broadcast/multicast(well-know-name, certificate)
Unicast(service-name, certificate)

Fig. 5: General service discovery message flow for NDN and MF

to each Aggregator that host this service. The service host Aggregator in the group then listen on this group GUID, and use the group private key to decrypt the incoming discovery message. Finally, we note that this form of secure service discovery is difficult for NDN.

As 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 hosting this services replies with acknowledgment if all conditions match.

D. Data Aggregation and Context Data Processing

In order to facilitate context-aware communication and data retrieval, we need to support context processing in the IoT system. The objective of context processing is to expose the sensor's low- level context information to upstream aggregators and LSGs, as well as to resolve the application's high-level context requirements using lower-level sensor contexts. The context processing service usually runs on both aggregators and LSGs.

Context processing requires the underlying network to be able to support in-network computing at both application and network levels. ICN inherently supports in-networking computing and caching, which thus offers unique advantages compared to traditional IP network where the support for in-network computing and caching is poor.

Application level contexts differ from application to application, and therefore, we need to provide a set of basic mechanisms to support efficient context processing. Firstly, the network needs to define a basic set of contextual attributes for devices (including sensors, aggregators, and LSGs), including device-level attributes (such as location, data type, battery level, etc), network-level attributes (such as ICN names), and service-level attributes (such as max, min, average, etc).

Secondly, we need to have means to expose sensor/aggregator/LSG contextual attributes to the rest of the system, through centralized services such as naming resolution service.

Thirdly, the IoT server needs to allow applications (either producers or consumers) to specify their contextual requirements. Fourthly, the unconstrained part of ICN-IoT needs to be able to map the higher- level application-specific contextual requirements to lower-level device-level and network-level contextual information.

E. Device/App Naming service

The objective of the naming service is to assure that either device or service itself is authenticated, attempting to prevent sybil (or spoofing) attack [7] and that the assigned name closely binds to the device (or service). Naming service is specific to MobilityFirst, and it is hard to achieve in the context of NDN. Naming service assigns and authenticates sensor and device names. An effective naming service should be secure, persistent, and able to support a large number of application agnostic names.

Traditional IoT systems use IP addresses as names, which are insecure and non-persistant. IP addresses also have relatively poor scalability, due to its fixed structure. Instead, ICN separates names from locators, and assigns unique and persistent names to each sensor/device, which satisfies the above requirements.

In what follows, we discuss an approach for secure naming service in ICN-IoT. We first consider the case where the embedded system is programmable so that before deployment, the owner can preload identity information (such as secure ID, a pair of public/private key and a certificate) , or has some manufacture ID and a pair of public private key (which is certified by the manufacturer). That is, the device is associated with information including device identity, public/private keys (PK_{device}, SK_{device}) and a certificate either from the owner or the manufacturer which certifies the device identity and public/private keys. When such a device is discovered, the aggregator will first verify the device identity (e.g., the device can generate a signature with the private key SK_{device} and present the signature and the certificate to the aggregator so that the aggregator can verify it), and then assign a name to the device as follows: the aggregator will issue a request to LSG together with its device identity and PK_{device}, so that LSG can assign an NDN name and generate a certificate (certifying the binding of NDN name, PK_{device}). To this end, the ICN name and the certificate will be sent back to the aggregator and will be stored locally if the device is resource-restricted. Otherwise, the ICN name and the certificate will be passed to the device.

For the MF-IoT, assigning a GUID for a device is rather straightforward: after verifying the device identity, the Aggregator inserts the public key PK_{device} and device information to the upper layer component to verify if there is a conflict in the corresponding scope. Specifically, LSG is in charge of local scope and IoT server guarantees the global uniqueness. Finally, the unique public key is used a GUID for the new device. Analogously, service discovery can be secured in a similar way.

In the case where devices are only associated with the secure manufacture ID while without being pre-loaded public/private keys and the certificate, it is critical to assure that devices are authenticated by using other trust model. For example the system can take advantage of the web-of-trust model or the contextually semantic information so that the devices manufactured by the same vendor can authenticate each other. Moreover, in order to comply with the capability of resource-restricted devices, light-weight cryptographic primitive (such as symmetric cryptography) may be used instead of public key cryptography.

Finally, we note that the same naming mechanism can be used to name higher-level IoT devices such as aggregators and LSGs.

F. *Publish/Subscribe Management*

Data Publish/Subscribe (Pub/Sub) is an important function for ICN-IoT, and is responsible for IoT information resource sharing and management. The objective of pub/sub system is to provide centralized membership management service. Efficient pub/sub management poses two main requirements to the underlying system: high data availability and low network bandwidth consumption.

In conventional IP network, most of the IoT platforms provide a centralized server to aggregate all IoT service and data. While this centralized architecture ensures high availability, it scales poorly and has high bandwidth consumption due to high volume of control/data exchange, and poor support of multicast.

Next we consider two decentralized pub/sub models. The first one is the Rendezvous mode that is commonly used for today's pub/sub servers, and the second one involves Data-Control separation that is unique to ICN networks where the control messages are handled by the centralized IoT server and the data messages are handled by the underlying ICN network. Compared to the popular Rendezvous mode where both control and data messages both meet at the centralized server, separating data and control messages can greatly improve the scalability of the entire system, which is enabled by the ICN network.

In today's IP network, Rendezvous mode is the classic pub/sub scheme in which data and requests meet at an intermediate node. In this case the role of the IoT server is only required to authenticate the consumers and providing it Rendezvous service ID.

While NDN is a Pull-based architecture without supporting the Pub/Sub mode naturally, COPSS [3] proposes a solution to fix this problem. It integrates a Push-based multicast feature with the pull based NDN architecture at the network layer by introducing Rendezvous Node (RN). RN is a logical entity that resides in NDN forwarder. 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 subscribers. The data publisher just sends the content using Publish packet by looking up FIB instead of PIT. If the same content prefix is requested by multiple subscribers, RN will deliver one copy of content downstream, which reduces the bandwidth consumption substantially.

Compared with the Rendezvous mode in which data plane and control plane both reside on the same ICN network layer,

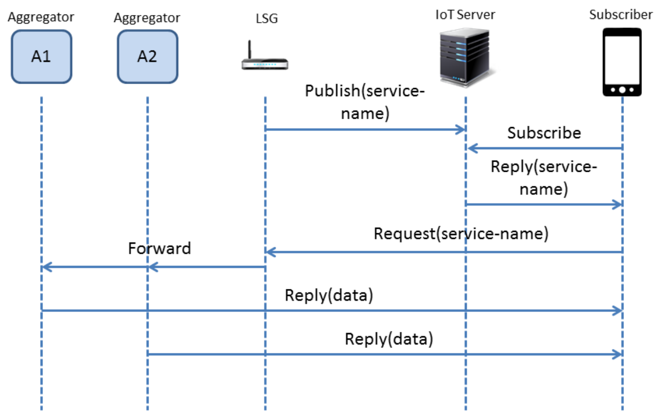

Fig. 6: Publish/subscriber management message flow

we consider an architecture where the control message is handled by the centralized server 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 requests from subscribers. Since the subscribers has non knowledge about the number of resource providers and their identities in a dynamic scenario, IoT server has to take responsibility of grouping and assigning group name for the resource. The generic message flow for publish/subscribe function is shown in Figure 6, and we will illustrate the detail for NDN and MF. In NDN, the grouping resource is intuitive: all resources have the same schematic prefix will be grouped together, and the prefix can be used to identify the service. The traditional NDN supports using a common partial name to retrieve multiple resources via Selector, but M. Amadeo et al. [1] provides a more efficient solution to improve this mechanism by introducing long-life multi-source Interest in PIT. MF takes advantage of $Group\text{-}GUID$ to identify a service provided by multiple resources. This $Group\text{-}GUID$ will be distributed to the subscriber as well as the publisher. In an example of NDN, it uses the common prefix$/home/monitoring/$ to identify a group of resource that provides multiple monitoring services such as $/home/monitoring/temperature$ and $/home/monitoring/light$. The subscriber retrieves the prefix from the IoT server, and sends Interest toward the resource. In a MF example, $GUID\text{-}x$ identifies the "home monitoring" service that combines with "light status" and "temperature". The resource producers, i.e. the host of "temperature" and the host of "light status" are notified that their services belong to $GUID\text{-}x$, then listen on $GUID\text{-}x$. The subscriber sends the request containing $GUID\text{-}x$ through multicasting which ultimately reaches the producers at last common node. Once receiving the request, the resource producer unicasts the data to the subscriber. In addition, if multiple resource consumers subscribe to the same resource, the idea of $Group\text{-}GUID$ can be reused here to group the consumers to further save bandwidth using multicast.

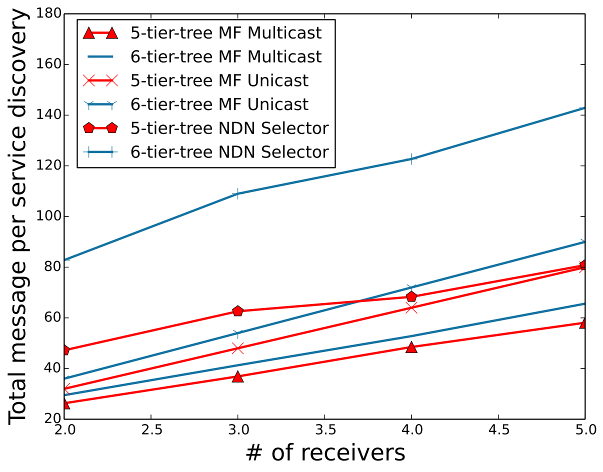

Fig. 7: Overhead of service discovery in tree topology

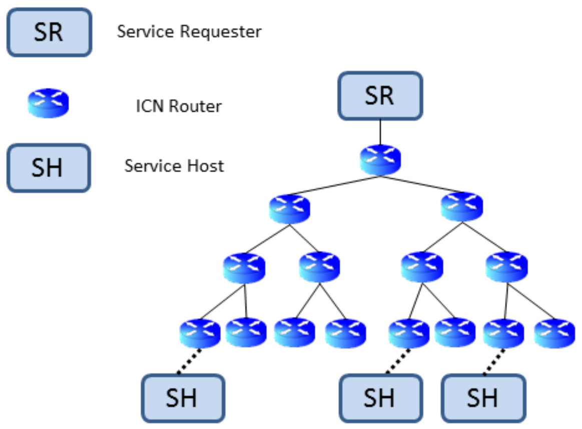

Fig. 8: Topology example for a 6-tier tree

IV. Evaluation

In this section, we investigate the efficiency of service discovery based on MF multicast, MF unicast and NDN interest selector. We have conducted detailed experiments in NS3-based MF simulator and ndnSIM. MF network protocol is implemented over standard NS3 P2P module and the experiments are running in a tree-based topology. In MF multicast case, $GroupGUID$ is used to identify a service at unknown location, i.e., service host (SH) can be located at any leaves of the tree while service requester (SR) attaches to the root of the tree. SR issues a request and the first router performs GNRS lookup for the $Group\text{-}GUID$. On receiving the service request message, the matched receivers will reply with an acknowledgement to the service requester via unicast. While in MF unicast case, we configure the SR to issue requests periodically and each matched SH will reply the request sequentially. In NDN selector experiment, SR expresses Interest with name "/service", and corresponding full name is "/service/$\{1,2,..., n\}$". On receiving the response from one of the SHs, the SR appends the response's name to the exclusion list of a Interest and re-expresses a Interest with the same name. This operation will repeat until no response comes back. Figure 7 shows the overhead in terms of total message per service discovery in tree topology that is shown in Figure 8. In the 5-tier tree case, we observe that MF multicast introduces the least overhead among the three schemes, as it triggers only one GNRS lookup per request, and the routers only forwards request duplicates based on the next common hop(s). When the number of SH increases to 5, NDN interest selector and MF unicast achieve similar performance. This is due to selector prevents both routers and SHs forwarding the discovered responses, which decreases the total response message. In MF unicast, total number of messages grows linearly as the number of SH increases. The result of another similar experiment but with a 6-tier tree demonstrates that NDN selector scheme encounters the highest overhead due to the impact of network scale increment on Interest broadcast.

V. Conclusion

In this paper, we present an ICN-based IoT middleware architecture to support various IoT applications. We discuss the detailed protocol design to support the middleware functionalities including device discovery, naming service, service discovery and pub/sub management. The simulation result shows that service discovery over MF multicast brings less overhead than MF unicast and NDN Selector approaches if multiple service hosts belong to the same group. For the future work, we will refine the middelware design details and prototype an ICN-based IoT middleware platform, and our ultimate target is to realize a full-fledged ICN-IoT middle that fits for real IoT environment.

References

[1] 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.

[2] 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*. ACM, 2014, pp. 77–86.

[3] 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*.

[4] H. Harney and C. Muckenhirn, "Group key management protocol (gkmp) architecture," 1997.

[5] S. Li, Y. Zhang, D. Raychaudhuri, and R. Ravindran, "A comparative study of mobilityfirst and ndn based icn-iot architectures," in *Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine), 2014 10th International Conference on*. IEEE, 2014, pp. 158–163.

[6] 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.

[7] J. Newsome, E. Shi, D. X. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proceedings of the Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, Berkeley, California, USA, April 26-27, 2004*, 2004, pp. 259–268.

[8] G. C. Polyzos and N. Fotiou, "Building a reliable internet of things using information-centric networking," *Journal of Reliable Intelligent Environments*, pp. 1–12, 2015.

[9] R. Ravindran, T. Biswas, X. Zhang, A. Chakraborti, and G. Wang, "Information-centric networking based homenet," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 1102–1108.

[10] 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*, 2012.

[11] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using named data networking," *Network, IEEE*, vol. 28, no. 3, pp. 50–56, 2014.

[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*.

[13] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[14] Y. Zhang, D. Raychadhuri, R. Ravindran, and G. Wang, "Icn based architecture for iot - requirements and challenges," IETF Internet Draft draft-zhang-iot-icn-architecture-00. IRTF, Tech. Rep., 2013.