IoT middleware over Information-Centric Network

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

ABSTRACT

hello

1. INTRODUCTION

requirement in the draft,

1.1 ICN IoT Middleware

1.2 NDN Background

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 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 groups multiple network object 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 router queries GNRS server for a group GUID mapping, GNRS server returns a list of member GUIDs.

2. RELATED WORK

NDN BMS, NDN Multisource Retrival PSI Architecture

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.

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
- *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 service expose resource funtionality hosted on devices that provide some forms of physical access to the entity cite***IoT service paper**. In general, the IoT service can be separated into three classes, which are sensing service, actuating service and control service. In current IoT platform, they are connected via a server or require more development effort such as maintaining a local name to IP mapping to be interconnected. With ICN architecture support, IoT service discovery and configuration become less complex and more efficient. Depending on sthe application scenario, there are two mode: $peer - to - peer\ mode$ and $master - slave\ mode$.

Peer-to-peer Mode: A source aggregator tries to discover a IoT service provided by another aggregator. In NDN, the source aggregator expresses an interest using a well-known name "$/area/servicename/certificate$" via a broadcast channel to reach the target aggregator. In MF, this can be handled by multicast (mentioned in section ???) without flooding the entire local network. All device provide related service can join a multicast group which is identified by a $Group - GUID$. The source send 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 condi-

tion 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

Service discovery Multisource retrieval

5. CONCLUSION

Conclusion