IRTF Contribution

ICN based Architecture for IoT

Standard contributions

Author(s)	Prof. Yanyong Zhang, Prof. Dipankar Raychadhuri (Winlab, Rutgers University) Ravi Ravindran and Guo-Qiang Wang (Futurewei)
Date	October, 2013
Status	Draft
Version	1.0
Template Version	

TABLE OF CONTENTS

1 Executive Summary

Internet of Things (IoT) promises to connect billions of objects to Internet. After deploying many stand-alone IoT systems in different domains, the current trend is to develop a unified IoT platform so that objects can be made accessible to applications across organizations and domains. Towards this goal, quite a few proposals have been made to build a unified IoT platform as an overlay on today's Internet. Such an overlay solution, however, is inadequate to address the important challenges posed by a unified IoT system, especially in terms of mobility, scalability, and communication reliability, due to the inherent inefficiencies of the current Internet. To address this problem, we propose to build a unified IoT platform based on the Information Centric Network (ICN) architecture, which we call ICN-IoT. ICN-IoT leverages the salient features of ICN, and thus provides seamless mobility support, scalability, and efficient content and service delivery.

In this proposal, we first present a few popular IoT scenarios in smart homes, smart grid, smart transportation, and smart healthcare. Then we identify a list of important requirements with the unified IoT architecture that promises to support tens of billions of objects. After discussing the weaknesses of the current overlay-based IoT solution, we propose an ICN-based solution, ICN-IoT, which can sufficiently satisfy these requirements. We present an example ICN-IoT architecture and discuss how it supports efficient data discovery, data processing and data distribution. Finally, we show that ICN-IoT efficiently supports context-based scenarios, which are very common for many IoT applications.

2 IoT Motivation and Challenges

During the past decade, many standalone Internet of Things (IoT) systems 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 great challenges on the underlying network and systems. To name a few, it needs to support 50-100 Billion networked objects [1], many of which are mobile. The objects will have extremely heterogeneous means of connecting to the Internet, often with severe resource constraints. Interactions between the applications and objects are often real-time and dynamic, requiring strong security and privacy protections

Before we present our solution to such a unified platform, we first describe a few popular IoT scenarios to motivate this proposal.

2.1 Popular scenarios

Several types of IoT applications exists, where the goal is efficient and secure management and communication among objects in the system and with the physical world through sensors, RFIDs and other devices. Below we list a few popular IoT applications.

2.1.1 Smart Homes

Home is a complex ecosystem for smart systems such as climate control, home security monitoring, smoke detection, or smart meters. In a unified IoT platform, we would inter-connect these systems through the Internet, such that they can interact with each other and make decisions at an aggregated level. Also, the systems can be accessed and manipulated remotely. The challenges for smart home include topology independent service discovery, common protocol for heterogeneous device/application/service interaction, policy based routing/forwarding, service mobility as well as privacy protection.

2.1.2 Smart Grid

Central to smart grid is data flow and information management, achieved by using sensors and actuators, which enables important capabilities such as substation and distribution automation. In a unified IoT platform, data collected from different smart grids can be integrated to reach more significant optimizations. The challenges for smart grid include reliability, real-time control, secure communications, and data privacy.

2.1.3 Smart Transportation

We are currently witnessing the increasing integration of sensors into cars. Current production cars already carry many sensors ranging from rain gauges and accelerometers over wheel rotation/traction sensors, to cameras. While intended for internal vehicle functions, these could also be networked and leveraged for applications such as monitoring external traffic/road conditions. Further, we can build vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications that enable many more applications for safety, convenience, entertainment, etc. The challenges for smart transportation include fast data/device/service discovery and association, efficient communications with mobility, trustworthy data collection and exchange.

2.1.4 Smart Healthcare

As more embedded medical devices, or devices that can monitor human health become increasingly deployed, smart healthcare is becoming a viable alternative to traditional healthcare solutions. For smart health applications, a unified IoT platform is critical for sharing data and enabling timely actuations. The challenges in smart healthcare include real-time interactions, high reliability, short communication latencies, trustworthy, security and privacy.

3 IoT Architectural Requirements

A unified IoT platform has to support interactions among a large number of mobile devices across the boundaries of organizations and domains. As a result, it naturally poses stringent requirements in every aspect of the system design. Below, we outline a few important requirements that a unified IoT platform has to address.

3.1 Naming

The first step towards realizing a unified IoT platform is the ability to assign names that are unique within the scope/lifetime of each device, data items generated by these devices, or a group of devices. Naming has the following requirements. First, names need to be application-centric, i.e., solely serving the purpose of an application or service. Second, names need to be persistent against dynamic attributes that are common in IoT systems, such as mobility or migration. Third, names need to be secure based on application requirement.

3.2 Scalability

Cisco predicts there will be around 50 Billion IoT devices such as sensors, RFID tags, and actuators, on the Internet by 2020 [1]. As mentioned above, a unified IoT platform needs to name every informational entity such as data, devices, etc. To deal with scalability, the name-locator separation is the basic requirement, implying that it is necessary to have a name resolution service. The requirement on such a resolution service is thus clear: the system should be able to insert/update/look up a name within a short latency. To satisfy this requirement, de-centralization is the key.

3.3 Resource Constraints

IoT devices can be broadly classified into two groups: resource-sufficient and resource-constrained. In general, there are the following types of resources: power, computing, storage, and bandwidth.

Power constraints of the IoT devices limit how much data these devices can communicate, as it has been shown that communications consume more power than other activities for embedded devices. Flexible techniques to collect the relevant information are required, and uploading every single produced data to a central server is undesirable. Computing constraints limit the type and amount of processing these devices can perform, also information needs to be available where it is more likely to be consumed. As a result, more complex processing needs to be conducted elsewhere, example at the network edge, which makes it important to balance local computation versus communication. Storage constraints of the IoT devices limit the amount of data that can be stored on the devices. This constraint means that unused sensor data may need to be discarded from time to time. Bandwidth constraints of the IoT devices limit the amount of communication these devices can have, which will have the same implication on the system architecture as the power constraints; namely, we cannot afford to communicate with every single sensor data generated by the device.

3.4 Traffic Characteristics

IoT traffic can be broadly classified into local area traffic and wide area traffic. Local area traffic is between nearby devices. For example, neighboring cars may work together to detect potential hazards on the highway, sensors deployed in the same room may collaborate to determine how to adjust the heating level in the room. These local area communications often involve data aggregation and filtering, have real time constraints, and require fast device/data/service discovery and association. At the same time, the IoT platform has to also support wide area communications. For example, commuters can find out real-time traffic and road information and then decide which commuting route to take. Wide area communications require efficient data/service discovery and resolution services.

While traffic characteristics for different IoT systems are expected to different, certain IoT systems have been analyzed and shown to have comparable uplink and downlink traffic volume in some applications such as [2], which means that we have to optimize the bandwidth/energy consumption in both directions. Further, IoT traffic demonstrates certain periodicity and burstiness [2]. As a result, when provisioning the system, peak traffic volume has to be considered.

3.5 Contextual Communication

Many IoT applications shall rely on contextual information such as social, grouping, location, type of ecosystem (home, grid, transport etc.) of devices and data (which are referred to as contexts in this document) to initiate dynamic relationship and communication. For example, cars traveling on the highway may form a “cluster” based upon their temporal physical proximity as well as the detection of the same event. These temporary groups are referred to as contexts. IoT applications need to support interactions among the members of a context, as well as interactions across contexts.

Temporal context can be broadly categorized into two classes, long-term contexts such as those that are based upon social contacts as well as stationary physical locations (e.g., sensors in a car/building), and short-term contexts such as those that are based upon temporary proximity (e.g., all taxicabs within half a mile of the Time Square at noon on Oct 1, 2013). Between these two classes, short-term contexts are more challenging to support, requiring fast formation, update, lookup and association of these contexts.

3. 6 Handling Mobility

Mobility in the IoT platform can mean 1) the data producer mobility (i.e., location change), 2) the data consumer mobility, 3) IoT Network mobility; and 3) disconnection between the data source and destination pair (e.g., due to unreliable wireless links). The requirement on mobility support is to be able to deliver IoT data below an application acceptable delay constraint in all of the above three cases.

There are varying degrees of mobility in a unified IoT platform, ranging from static as in fixed assets to highly dynamic in vehicle-to-vehicle environments.

3.7 Storage and Caching

Storage and caching plays a very significant role depending on the type of IoT ecosystem with the fact that data generated is also subjected to privacy and security guidelines. In a unified IoT platform, depending on content caching requirements can be cached at will or at service authorized points, and thus we don’t need to always forward a content request to its original creator. Rather, locating and receiving a cached copy is sufficient for IoT applications. This optimization can greatly reduce the content access latencies.

In network storage and caching, however, has the following requirements on the IoT platform. First, the platform needs to support the efficient resolution of cached copies. Second, certain content should not be cached anywhere, and the service platform should be able to enforce this. Third, the platform should strive for the balance between caching, content security/privacy, and regulations.

3.8 Security and Privacy

The unified IoT platform makes physical objects accessible to applications across organizations and domains, and security and privacy thus become a serious concern. Security includes data integrity, authentication, trust management, and access control at different layers of the IoT platform. Privacy means that both the content and the context around IoT data need to be protected. These requirements will be driven by various stake holders such as industry, government, consumers etc.

3.9 Communication Reliability

IoT applications can be broadly categorized into mission critical and non-mission critical. For mission critical applications, reliable communication is one of the most important features as these applications have strong QoS requirements. Reliable communication requires the following capabilities for the underlying system: (1) seamless mobility support to support for extreme disruptions (DTN), (2) efficient routing in the presence of intermittent disconnection, and (3) QoS aware routing.

3.10 Self-Organization

The unified IoT platform should be able to self-organize to meet various application requirements, especially the capability to quickly discover heterogeneous and relevant devices/data/services based on the context. This discovery can be achieved through an efficient platform-wide publish-subscribe service, or through private community grouping/clustering based upon trust and other security requirements. In the former case, the publish-subscribe service must be efficiently implemented, able to support seamless mobility, in-network caching, name-based routing, etc. In the latter case, the IoT platform needs to discover the private community groups/clusters efficiently.

3.11 Ad hoc and Infrastructure Mode

Depending upon whether there is communication infrastructure, an IoT system can be classified as either ad-hoc or infrastructure mode. For example, a vehicle may determine to report its location and status information to a server periodically through cellular connection, or, a group of vehicles may form an ad-hoc network that collectively detect road conditions around them. In the cases where infrastructure is unavailable, one of the participating nodes may choose to become the temporary gateway. The open IoT platform needs to be able to handle both situations.

The unified IoT platform needs to design a common protocol that serves both modes. Such a protocol should be able to provide: (1) energy-efficient topology discovery and data forwarding in the ad-hoc mode, and (2) scalable name resolution in the infrastructure mode.

4 State of the Art

Over the years, many stand-alone IoT systems have been deployed in various domains. These systems usually adopt a vertical silo architecture and support a small set of pre-designated applications. A recent trend, however, is to move away from this approach, towards a unified IoT platform in which the existing silo IoT systems, as well as new systems that are rapidly deployed, will make their data and services accessible to general Internet applications. In such a unified platform, physical world resources can be accessed over Internet and shared across the boundaries of physical, enterprise and application. However, current approaches to achieving this objective are based upon Internet overlays, whose inherent inefficiencies hinders the platform from satisfying the IoT requirements outlined earlier (particularly in terms of scalability, security, mobility, and self-organization)

4.1 Silo IoT Architecture

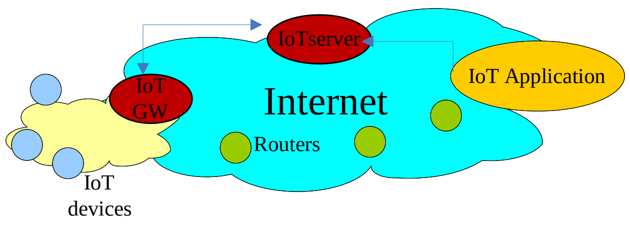


Figure 1: Silo architecture of standalone IoT systems

A typical standalone IoT system is illustrated in Figure 1, which includes devices, a gateway, a server and applications. Many IoT devices have limited power and computing resources, unable to directly run normal IP access network (Ethernet, WIFI, 3G/LTE etc.) protocols. Therefore, we use the IoT gateway to connect these devices to the server. Through the IoT server, applications can subscribe to the data collected by the devices, or interact with the devices.

There have been quite a few popular protocols for standalone IoT systems, such as DF-1, MelsecNet, Honeywell SDS, BACnet, etc. However, these protocols are host-driven, instead of information driven, leading to a highly fragmented protocol space with limited interoperability.

4.2 Overlay Based Unified IoT Solutions

The current approach to a unified IoT platform is to make IoT gateways and servers adopt standard APIs. IoT devices connect to the Internet through the standard APIs and IoT applications subscribe and receive data through standard control and data APIs. Building on top of today's Internet as an overlay, this is the most practical approach towards a unified IoT platform. There are ongoing standardization efforts including ETSI[3], and CORE[4]. Network operators can use standard API to build common IoT gateways and servers for their customers. Figure 2 shows the architecture adopted in this approach.

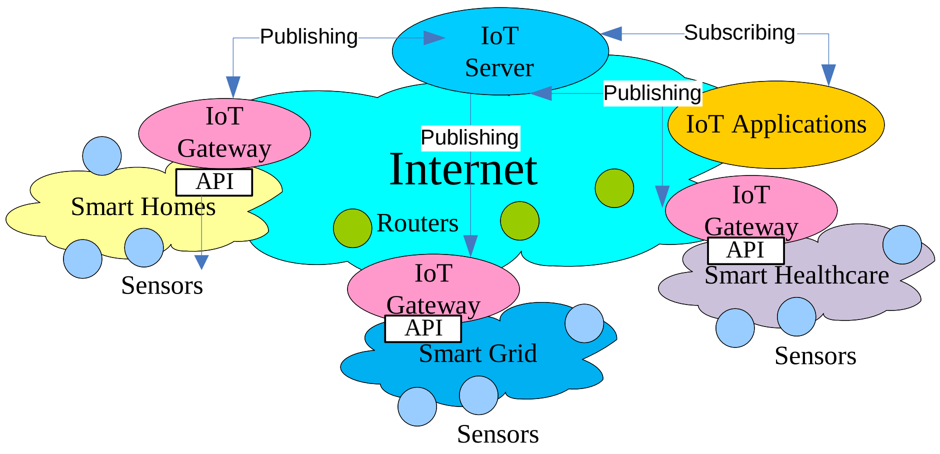


Figure 2: Implementing an open IoT platform through standarized APIs on the IoT gateways and the server

4.2.1 Weaknesses of the Overlay-based Approach

The above overlay-based approach can work with many different protocols, but the system is not designed in a holistic manner. Another limiting factor is that it is built upon today's IP network, which has a few inherent weaknesses towards supporting a unified IoT system. As a result, it cannot satisfy some of the requirements we outlined in Section 3:

- *Naming*. The overlay-based approach uses IP addresses as names at the network layer, which are not persistent for mobile devices/services.
- *Scalability*. The overlay-based approach uses IP addresses as names at the network layer, which does not provide the scalability needed to support device/service mobility or flexible name resolution.
- *Resource constraints*. The overlay-based approach requires every device to send data to the IoT server. Resource constraints of the IoT devices, especially in power and bandwidth, will seriously limit the performance of this approach.
- *Traffic Characteristics*. In this approach, applications are written in a host-centric manner, instead of being information-centric. As a result, it is challenging for the underlying system to satisfy the communication patterns of the applications. Further, characteristics of today's Internet, such as the lack of multicast, will make the matters worse.
- *Contextual Communications*. This overlay-based approach cannot react to dynamic contextual changes in a timely fashion. The main reason is that context lists are kept at the IoT server in this approach, and they cannot help efficiently route requests/information at the network layer.
- *Mobility*. The overlay-based approach cannot seamlessly support device mobility. In this approach, communications are IP driven, which is inefficient for mobility support.
- *Storage and Caching.* The overlay-based approach does not provide efficient storage/caching support. Also, applications are written in a host-centric manner, wherein network requests are bound to a specific destination host/server instead of a specific piece of content.
- *Communication Reliability*. The overlay-based approach offers insufficient communication reliability when the devices/services/data are mobile.
- *Self-Organization.* The overlay-based approach is topology-based as it is bound to IP semantics, and thus does not sufficiently satisfy the self-organization requirement.
- Ad-*hoc and infrastructure mode.* As mentioned above, the overlay-based approach lacks self-organization, and thus does not provide efficient support for the ad-hoc mode.

5 Proposed ICN-Centric Unified IoT Platform

In recent years, the current Internet has become inefficient in supporting rapidly emerging Internet use cases, e.g., mobility, content retrieval, IoT, context, etc. As a result, Information Centric Network ([5]) has been proposed as a future Internet design to address these inefficiencies. ICN has the following main features: (1) it identifies a network object (including a mobile device, a content, a service, or a context) by its name instead of its IP address, (2) a hybrid name/address routing, and (3) a delay-tolerant transport. These features make it easy to realize many in-network functions, such as mobility support, multicasting, content caching, cloud/edge computing and security.

Considering these salient features of ICN, we propose to build a unified IoT platform using ICN, in which the overlay IoT services are only needed for administrative purposes, while the publishing, discovery, and delivery of the IoT data/services is directly implemented within the ICN network. Figure 3 shows the proposed ICN-centric IoT approach, which is centered around the ICN network instead of today's Internet.

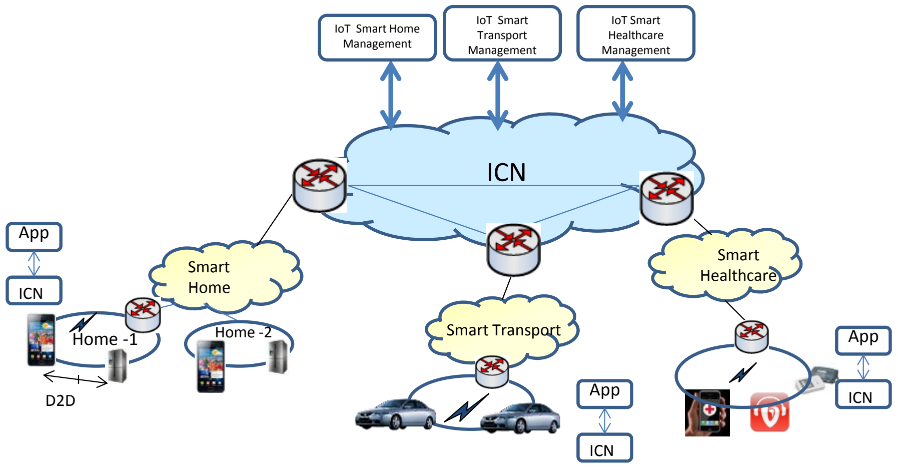

IoT Smart Home Management
IoT Smart Transport Management
IoT Smart Healthcare Management
ICN
App
ICN
Smart Home
Home -1
Home -2
D2D
Smart Transport
App
ICN
Smart Healthcare
App
ICN

Figure 3: The proposed ICN-centric IoT unified platform.

5.1 Strengths of ICN-IoT

Our proposed ICN-IoT is a network-layer IoT solution, which can satisfy the requirements of the open IoT platform:

- *Naming*. In ICN-IoT, we assign a unique name to an IoT object, an IoT service, or even a context. These names are persistent throughout their scopes.
- *Scalability*. In ICN-IoT, the name resolution is performed at the network layer, distributed within the entire network. Thus, it can achieve high degree of scalability exploiting features like content locality, local computing, and multicasting.
- *Resource constraints*. In ICN-IoT, mobile devices only need to upload their data to the gateway when some applications subscribe to the data. Thus, it offers a resource-efficient solution.
- *Local traffic Pattern.* In ICN-IoT, we can easily cache data/services in the network, facilitating local communications.
- *Context-aware communications*. In ICN-IoT, we assign unique names to contexts, which are then mapped to the devices that are involved in the context by the name resolution service. The support of multicast can greatly ease the communications to these devices.
- *Seamless mobility handling*. In ICN-IoT, ICN's name resolution layer allows multiple levels of mobility relying on receiver-oriented nature for self-recovery for consumers, to multicasting and late-binding techniques to realize seamless mobility support of producing nodes.

- *Data storage*. In ICN-IoT, data are stored locally, either by the mobile device or by the gateway nodes or at service points. We also implement in-network storage/caching [6] to speed up data delivery.
- *Security and privacy*. In ICN-IoT, secure binding between names and content instead of IP addresses to identify devices/data/services, is inherently more secure allowing pervasive caching.
- *Communication reliability*. In ICN-IoT, we support seamless mobility, which in turn guarantees reliable communications.
- *Ad hoc and infrastructure mode*. ICN-IoT support both ad-hoc and infrastructure modes.

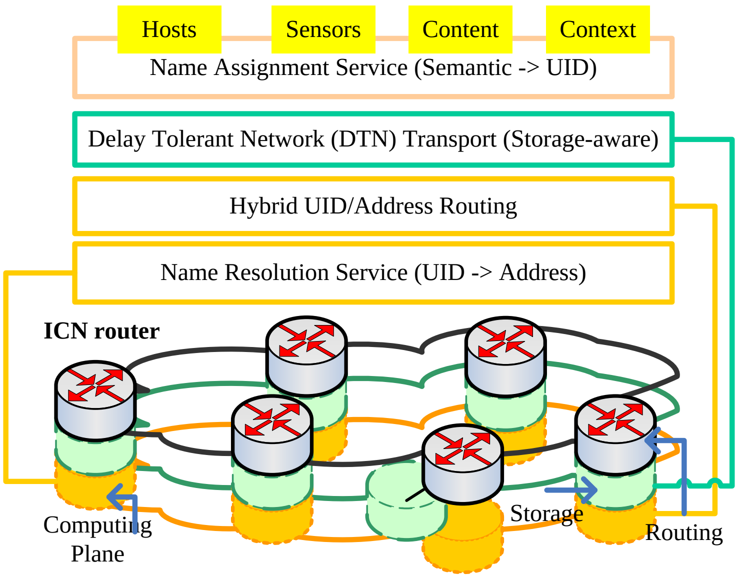

Hosts
Sensors
Content
Context
Name Assignment Service (Semantic -> UID)
Delay Tolerant Network (DTN) Transport (Storage-aware)
Hybrid UID/Address Routing
Name Resolution Service (UID -> Address)
ICN router
Computing Plane
Storage
Routing

Figure 4: An Example ICN-IoT Architecture

5.2 Example ICN-IoT Architecture

The design of ICN-IoT is to support a scalable, unified IoT architecture with tens of billions of devices. The ICN-IoT introduces a unique identifier (UID) for every network object, separated from its dynamic access network locations represented by network addresses. The separation of naming and addressing allows ICN-IoT to support identity based routing, which provides seamless mobility support. Multicast and anycast can then become a native network capability since a UID can be bound to multiple addresses.

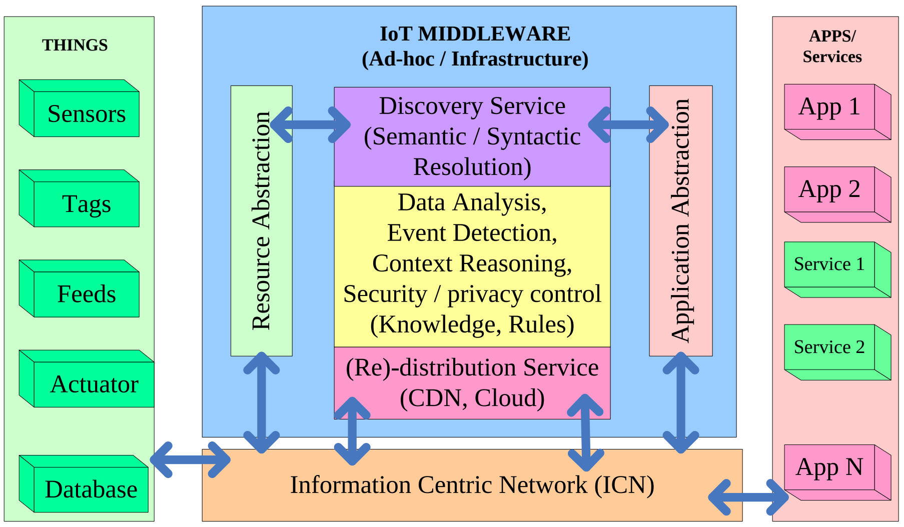
Figure 4 depicts the example ICN-IoT core network architecture. The assumption is that ICN routers have a sizeable storage and a computing plane in addition to the basic routing function. The proposed ICN-IoT network includes the following three basic services:

- *Name Resolution Service* (NRS): A core ICN-IoT function that maintains mappings between UID to network address (NA). This can be a logically centralized service distributed on all ICN routers.

- *Hybrid UID/network address routing*: An ICN router makes routing decisions for UID identified data blocks. It uses NRS to find the network address(s) for a UID. In case an explicit network address cannot be obtained from NRS, it may route based on intermediate network address and use “late-biding” from UID to its actual network address.

- *Delay-tolerant network (DTN) transport*: An ICN router can deliver UID-only identified data blocks from/to a networked object. The transport performs a cache and forward (CNF) style, hop-by-hop delivery. The data block is cached in router storage and forward to next hop based on routing decisions.

- In addition to baseline services, ICN-IoT assumes there is a name assignment service (NAS) chosen by the owner of a networked object. Through the NAS, the owner assigns and publishes a UID with its semantic descriptions for his networked object to a searchable space, such as Google.



THINGS
Sensors
Tags
Feeds
Actuator
Database
IoT MIDDLEWARE
(Ad-hoc / Infrastructure)
Resource Abstraction
Discovery Service (Semantic / Syntactic Resolution)
Data Analysis, Event Detection, Context Reasoning, Security / privacy control (Knowledge, Rules)
(Re)-distribution Service (CDN, Cloud)
Application Abstraction
Information Centric Network (ICN)
APPS/ Services
App 1
App 2
Service 1
Service 2
App N

Figure 5: The ICN-IoT Service Middleware

In Figure 5, we summarize the ICN-IoT milddleware into three service categories: 1) Data Discovery Service, 2) Data Processing Service and 3) Data (Re)Distribution Service. These services span both in Ad hoc and Infrastructure settings.

Data Discovery Service: Discovery service allows applications finding things/services/data through certain semantic / syntactic resolution. In ICN-IoT, each data/service has both a user-readable name with certain semantic structures and unique identifiers (UID). Based on the human readable names, the data discovery service can help discover the requested data/services.

Data Processing Service: The data processing service generates new knowledge or information through applying knowledge and/or rules on existing data. Data processing can be distributed based on its scope and relevance as in the local node, versus edge service routers, or in data centers. Data processing procedures are often defined by the applications/services.

Data Re-distribution Service: The (re)distribution service delivers data from their sources to different locations for better accessibility. These services will be directly supported by the ICN. Firstly, ICN can cache data/content on forwarding ICN routers, and secondly, these routers will be added to the NRS service so that requests for the cached data/content can be forwarded to them, thus reducing the data/content retrieval latencies.

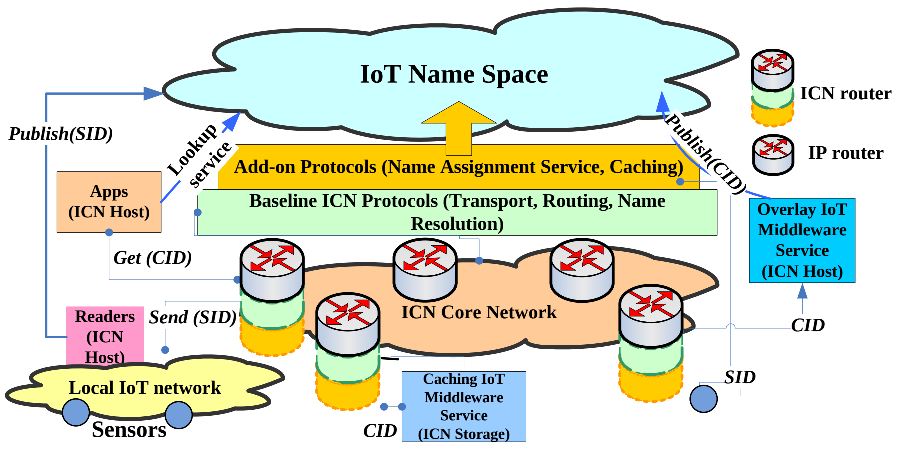

IoT Name Space
ICN router
IP router
Publish(SID)
Lookup service
Publish(CID)
Add-on Protocols (Name Assignment Service, Caching)
Baseline ICN Protocols (Transport, Routing, Name Resolution)
Apps (ICN Host)
Overlay IoT Middleware Service (ICN Host)
Get (CID)
ICN Core Network
Readers (ICN Host)
Send (SID)
CID
SID
Local IoT network
Sensors
CID
Caching IoT Middleware Service (ICN Storage)

Figure 6: ICN-IoT Data and Services

Figure 5 shows how the ICN-IoT middleware services (identified in Figure 4) integrated into the ICN infrastructure. The components of the IoT middleware can be described with respect the services identified before.

- **Realizing Data discovery Service.** Two tasks are involved in achieving this. First is a way to name services/content/devices of interest within the IoT application scope. Each network entity that is registered at the Name Assignment Service (NAS) has both a human readable name and a unique identifier (UID). The human readable name includes semantic key words that characterize the entity. Once the entities are named, applications based on its policy requirements make it accessible considering access control policies. To enable accessibility a discovery function is required, which is enabled through NAS.

  Figure 5 shows an example with sensor *SID* and service *CID*. Both SID and CID are published to the searchable name space through NAS, which can be efficiently discovered later by the applications.

- **Realizing Data processing Service**. ICN-IoT uses the in-network storage and computing capability to cache and execute *data processing* service, such as a context reasoning service, traditionally implemented on overlay servers. Having the context reasoning service within the network, however, requires additional computing capabilities offered by routers or separate computing facilities in the network. In Figure 5, the service, identified by *CID,* is cached and executed on an ICN router.

- **Realizing Data (re)distribution Service**. ICN-IoT performs *distribution or re-distribution* of IoT data and service through its native support of in-network caching and multicast based on identity based routing.

5.3 ICN-IoT Scenario

We will use a context-aware IoT application as an example from UbiCab [8] to demonstrate how IoT middleware service is enabled inside the proposed ICN-IoT core network. This example is based on MobilityFirst architecture [7] which is based on secured names and off-path name resolution infrastructure.

The example is stated as: "*James walks on NYC streets and wants to find an empty cab closest to his location*". In this example, we assume that James and taxi drivers have sensors on their phones, providing GPS location information as data. The context of this application is the relative locations of James and taxi drivers. A context service, as an IoT middleware, is needed to bridge sensors (providing GPS location) and the application (phone call program) automatically.

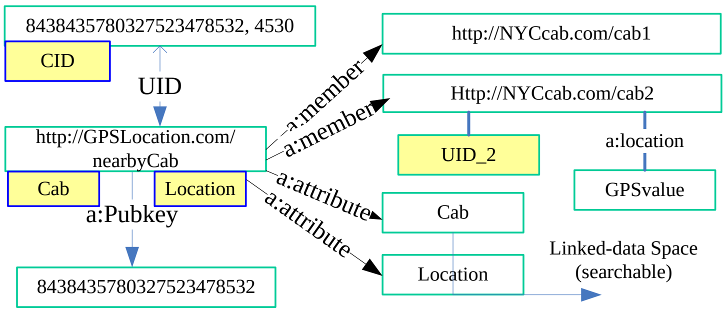


Figure 7: Location context service in RDF

We use a linked-data inspired approach to implement the middleware service for this location aware context. First, a company GPSlocation.com offers this service at *http://GPSlocation.com/nearbyCab*, implemented in an RDF graph shown in Figure 7. This context service has two attributes to describe itself, tagged as *cab* and *location.* A taxi driver can query this RDF graph and joins as a member of the service by adding a link to its own URI *http://NYCcab.com/cab1* in the RDF graph with a predicate "*a:member*". The sensor data, taxi driver's GPS location, is linked to taxi driver's URI with a predicate "*a:location"* and the value is updated by his phone periodically. The database containing this RDF graph must be searchable by users in a searchable space, which we refer as IoT resource space in Figure 6.

In ICN-IoT, we also add an additional 3-tuple or triple in the RDF graph with a predicate "UID". *CID* is assigned to the context service and *UID_2* is assigned to cab2.

If user James wants to find a taxi using such location context service, he can query the searchable IoT name space to find this context service *CID*. For example he can use an RDF query with "cab" and "location" as attributes to be matched.

This context service can also run as an overlay to the Internet. Taxi drivers and passengers can directly call the URI of the context service to join the membership and get a nearby cab, respectively. As an Internet overlay, the location context service is subject to longer delay due to demands are overloaded or unbalanced.

In this example, since the context service is implemented using an RDF graph, it can be cached on ICN routers in a similar way to caching content. This RDF graph can be updated by taxi drivers by joining membership, current GPS location etc, since the context service is UID identified it can be routed efficiently by the ICN routers. The utility of such information is determined by specific application requirements.

The application scenario is comprised of the following steps, illustrated by Figure 8.
Name Assignment. The owner of the context service publishes its *CID* and the service description (RDF graph of Figure 7) through ICN-IoT *NAS* function, running on both service host and routers, to a searchable Link-data space.
NRS Update. When the context service connects to the ICN network, the access router calls *NRS* update to provide a UID to network address mapping.
Service Discovery. Taxi driver or James can find *CID* through an RDF query requesting services with tags "location" and "cab".
Service Request. Taxi driver makes a service request to *CID* with an RDF query to join the service membership.
Routing and GNRS Lookup. The service request toward *CID* is routed to the *NA_c,* obtained from a *NRS* lookup.
Service Request Redirection. James makes a service request to *CID* with an RDF query to match GPS location to a nearby taxi. James' request to *CID* is redirected to *UID_2*, the driver of cab2.

Note that in this example, no CDN overlay is needed to support *multi-homing*, *multicasting* for a location context service because these are embedded in the ICN core network.

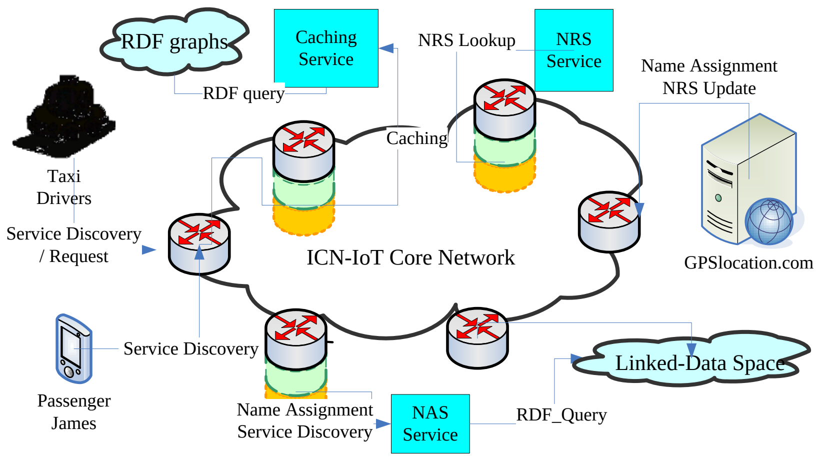


Figure 8: Location context application scenario

As usual, the benefit of caching is more significant for location dependent services. In our example, it is best to have the middleware service being offered at a server close to taxi drivers and passengers so that the traffic can be localized with lower latency. Traditionally, a service provider has to buy edge computing service from content distribution network (CDN); recently, cloud computing offers another alternative to deploy a distributed service. The native support of caching in ICN provides with the ability to distribute a service directly through core network routers. This option may not be suitable for heavyweight service requiring a high end web server. Nevertheless, it could be particularly useful for lightweight IoT middleware service. As shown in our example, when the context service is implemented in an RDF graph, caching a service is as simple as caching a data and service processing is as simple as a database query. The algorithms behind cache replacement can also be similar to what for data caching.

Once, for example, the service *CID* is cached, any request to *CID* is intercepted by the ICN router. In our example, when it is a request from taxi driver, it must contain an RDF query requesting either join membership or an updated GPS location. The computing layer of ICN router runs a SPARQL engine to interpret the RDF query and makes a corresponding database operation. When it is a request from James, a passenger, it must contain an RDF query containing its own GPS location and looking for the ID of a taxi with a GPS location nearby.

Another benefit of ICN caching is that there is no need for end-to-end connectivity between the requester and the location context server, if the service is currently cached in an accessible ICN router. This is especially useful for ad-hoc or DTN use cases, where the application network can be dynamically formed and isolated for a period of time.

This UbiCab example has low data rate. Next imagine a context service requires high data rate, for example, a person wants to see snapshots of the street nearby. And these snapshots are uploaded by the people on the street. Assuming every access network can get enough number of people uploading pictures, it is best for the network operator to cache the pictures as close to edge as possible.

Through this location context service example, we can see ICN-IoT offers significant native supports for IoT data and services to be visible, routable, cacheable and executable in the core network of future Internet architecture.

REFERENCES

[1] Cisco visual networking index: Global mobile data traffic forecast update, 2009-2014.

[2] M. Shafig, L. Ji, A. Liu, J. Pang, and J Wang, "A first look at cellular machine-to-machine traffic: large scale measurement and characterization," Proceedings of the ACM Sigmetrics 2012.

[3] ETSI. http://www.etsi.org/.

[4] Constrained RESTful Environments (CORE). https://datatracker.ietf.org/wg/core/charter/.

[5] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-Centric Networking: Seeing the Forest of the Trees," Hot Topics in Networking, 2011.

[6] L. Dong, Y. Zhang and D. Raychaudhuri, “Enhance Content Broadcast Efficiency in Routers with Integrated Caching”, in Proceedings of the IEEE Symposium on Computers and Communications (ISCC), 2011.

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

[8] Byoungjip, Kim , SangJeong, Lee , Youngki, Lee , Insoek, Hwang , Yunseok, Rhee “Mobiiscape: Middleware Support for Scalable Mobility Pattern Monitoring of Moving Objects in a Large-Scale City”, Journal of System and Software, July, 2011