ICN Research Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2018

Y. Zhang
D. Raychadhuri
WINLAB, Rutgers University
L. Grieco
Politecnico di Bari (DEI)
S. Sabrina
Universita degli studi dell Insubria
H. Liu
The Catholic University of America
S. Misra
New Mexico State University
R. Ravindran
G. Wang
Huawei Technologies

July 16, 2017

ICN based Architecture for IoT
draft-zhang-icnrg-icniot-architecture-01

Abstract

The Internet of Things (IoT) technology promises to connect billions of objects to Internet. Today, the IoT landscape is very fragmented from both the technology and service perspectives. As a consequence, it is difficult to integrate and cross-correlate data coming from the heterogeneous contexts and build value-added services on top. This reason has motivated the current trend to develop a unified de-fragmented IoT architecture so that objects can be made accessible to applications across organizations and domains. Several proposals have been made to build a unified IoT architecture as an overlay on top of today's Internet. Such overlay solutions, however, inherit the existing limitations of the IP protocol: mobility, security, scalability, and communication reliability. To address this problem, A unified IoT architecture based on the Information Centric Networking (ICN) architecture, which we call ICN-IoT [1] has been proposed. ICN-IoT leverages the salient features of ICN, and thus provides seamless device-to-device (D2D) communication, mobility support, scalability, and efficient content and service delivery. Furthermore, in order to guarantee the real diffusion of ICN-IoT architecture it is fundamental to deal with self-configuring features such as device onboarinding and discovery, service discovery, scalability, security and privacy requirements, content dissemmination since the IoT system will comprise of diverse applications with heterogenous requirements including connectivity, resource constraints and mobility. Towards this, the draft elaborates on a set of middleware functions to enable large operation of an ICN-IoT deployment.

This draft begins by motivating the need for an unified ICN-IoT architecture to connect heterogeneous IoT systems. We then propose an ICN-IoT system architecture and middleware components which includes device/network-service discovery, naming service, IoT service discovery, data discovery, user registration, and content delivery. For all of these components, discussions for secure solutions are offered. We also provide discussions on inter-connecting heterogeneous ICN-IoT domains.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2018.

Copyright Notice

Table of Contents

1. ICN-Centric Unified IoT Architecture

In recent years, the current Internet has become inefficient in supporting rapidly emerging Internet use cases, e.g., mobility, content retrieval, IoT, contextual communication, etc. As a result, Information Centric Networking (ICN) 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 architecture, in which the middlware IoT services are proposed for administrative purposes such as towards onboarding devices, device/service discovery and naming. Other functions such as name publishing, discovery, and delivery of the IoT data/services is directly implemented in the ICN network. Figure 1 shows the proposed ICN-IoT approach, which is centered around the ICN network instead of today's Internet.

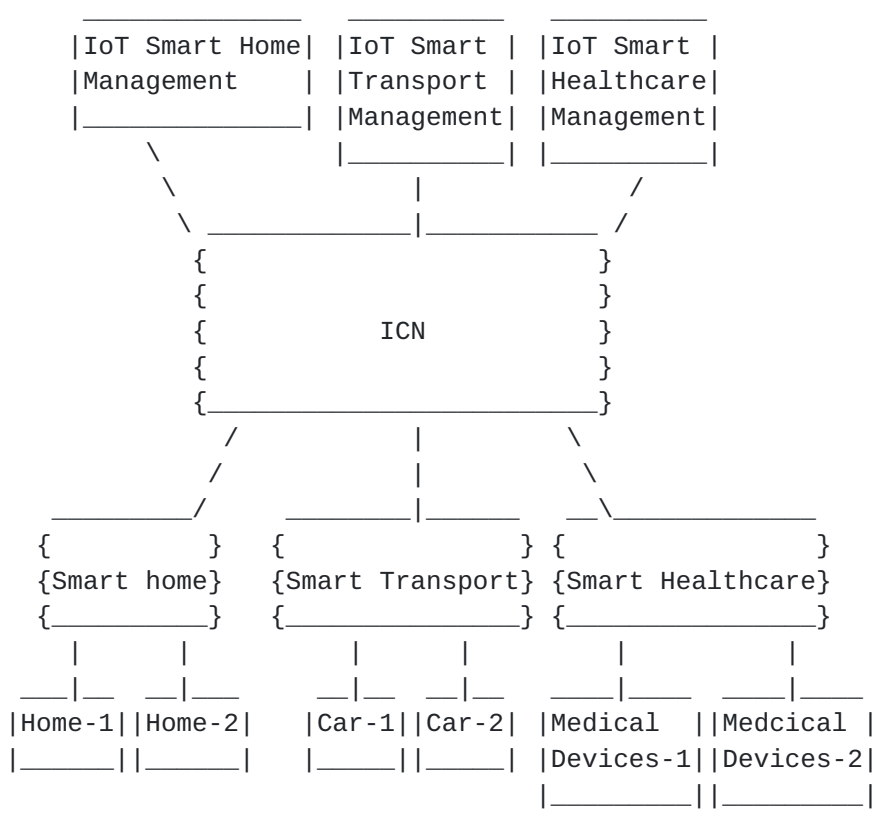

Figure 1: The proposed ICN-IoT unified architecture.

1.1. Strengths of ICN-IoT

Our proposed ICN-IoT architecture delivers IoT services over the ICN network, which aims to satisfy the requirements of the open IoT networking system (discussed in the ICN-IoT design considerations draft [1]):

- 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.

- Security and privacy. In ICN-IoT, secure binding between names and data objects instead of IP addresses to identify devices/data/services, is inherently more secure than the traditional IP paradigm [16].

- Scalability. In ICN-IoT, the name resolution is performed in the network layer in an online or using a mapping system with name state distributed within the entire network. Thus, it can achieve

high degree of scalability exploiting features like content locality, local computing, and multicasting.

- Resource efficiency. In ICN-IoT, in both the constrained and non-constrained parts of the network, only those data that are subscribed by applications or requested by clients in the specified context will be delivered. Thus, it offers a resource-efficient solution.

- Local traffic Pattern. In ICN-IoT, we can easily cache data or deploy computation services in the network, hence making communications more localized and reducing the overall traffic volume.

- Context-aware communications. ICN-IoT supports contexts at different layers, including device layer, networking layer and application layer. Contexts at the device layer include information such as battery level or location; contexts at the network layer include information such as network status and statistics; contexts at the application layer are usually defined by individual applications. In ICN-IoT, device context may be available to the network layer, while network elements (i.e., routers) are able to resolve application-layer contexts to lower-layer contexts. As a result of adopting contexts and context-aware communications, communications only occur under situations that are specified by applications, which can significantly reduce the network traffic volume.

- 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, and using multicasting and late-binding techniques to realize seamless mobility support of the producing nodes.

- Self-Organization: Name based networking allows service level self configuration and bootstrapping of devices with minimal manufacturer or administrator provisioned configuration. This configuration involves naming, key distribution and trust association to enable data exchange between applications and services.

- Data storage and Caching. In ICN-IoT, data are stored locally, either by the mobile device or by the gateway nodes or at service points. In-network caching [4] also speeds up data delivery and also offer local repair over unreliable links such as over wireless mediums.

- Communication reliability. ICN-IoT supports delay tolerant communications [23], which in turn provides reliable communications over unreliable links as mentioned earlier. Also opportunistic caching allows to increase the content copies in the network to help consumers with diverse application requirements (such as operating at different request rates) or situations dealing with mobility scenarios.

- Ad hoc and infrastructure mode. ICN-IoT supports both applications operating in ad-hoc [2] and infrastructure modes.

- In-network processing. ICN offers in-network processing that supports various network services, such as context resolution, data aggregation and compression.

2. ICN-IoT System Architecture

The ICN-IoT system architecture, which is also a general abstraction of an IoT system, is shown in Figure 2, has the following six main system components:

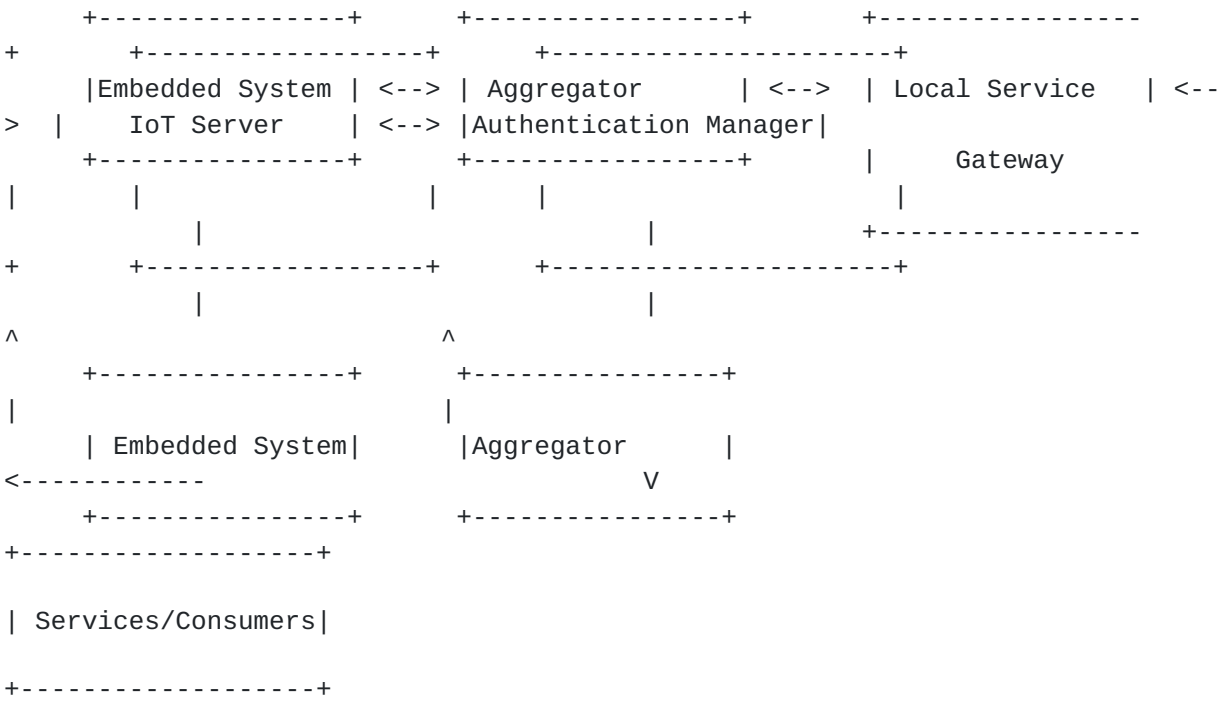

Figure 2: ICN-IoT System Architecture

- Embedded Systems (ES): The embedded sensor has sensing and actuating functions and may also be able to relay data for other sensors to the Aggregator, through wireless or wired links.

- Aggregator: It interconnects various entities in a local IoT network. Aggregators serve the following functionalities: device discovery, service discovery, and name assignment. Aggregators can communication with each other directly or through the local service gateway.

- Local Service Gateway (LSG): A LSG serves the following functionalities: (1) it is at the administrative boundary, such as, the home or an enterprise, connecting the local IoT system to the rest of the global IoT system, (2) it serves to assign ICN names to local sensors, (3) it enforces data access policies for local IoT devices, and (4) it runs context processing services to generate information specified by application-specific contexts (instead of raw data) to the IoT server.

- IoT Server: Within a given IoT service context, the IoT server 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, certificates, and impose other security functions such as access control.

- Authentication Manager (AM): The authentication manager serves to enable authentication of the embedded devices when they are onboarded in the network and also if their identities need to be validated at the overall system level. The authentication manager may be co-resident with the LSG or the IoT server, but it can also be standalone inside the administrative boundary or outside it.

- Services/Consumer: These are other application instances interacting with the IoT server to fetch or be notified of anything of interest within the scope of the IoT service.

The logical topology of the IoT system can be mesh-like, with every ES attached to one or more aggregators or to another ES, while every aggregator is attached to one or more LSG and all the LSGs connected to the IoT server. Thus, each ES has its aggregators, and each of which in turn has its LSG. All ES belonging to the same IoT service share the same IoT server. All the aggregators that are attached to the same LSG management are reacheble to to one another hence capable of requesting services or content from them. While such richer connectivity improves reliability, it also requires control plane sophistication to manage the inter-connection. Though our discussion can be generalized to such mesh topologies, in the rest of the draft, we will focus on the tree topology for the sake of simplicity.

3. ICN-IoT Middleware Architecture

The proposed ICN-IoT middleware aims to bridge the gap between underlying ICN functions, IoT applications and devices to achieve self-organizing capability.

The middleware functions are shown in Fig. 3 and it includes six core functions: device discovery, naming service, service discovery, context processing and storage, pub/sub management, and security which spans all these functions.

In contrast to centralized or overlay-based implementation in the legacy IP-based IoT platform, ICN-IoT architecture pushes a large portion of the functionalities to the network layer, such as name resolution, mobility management, in-network processing/caching, context processing, which greatly simplifies the middleware design.

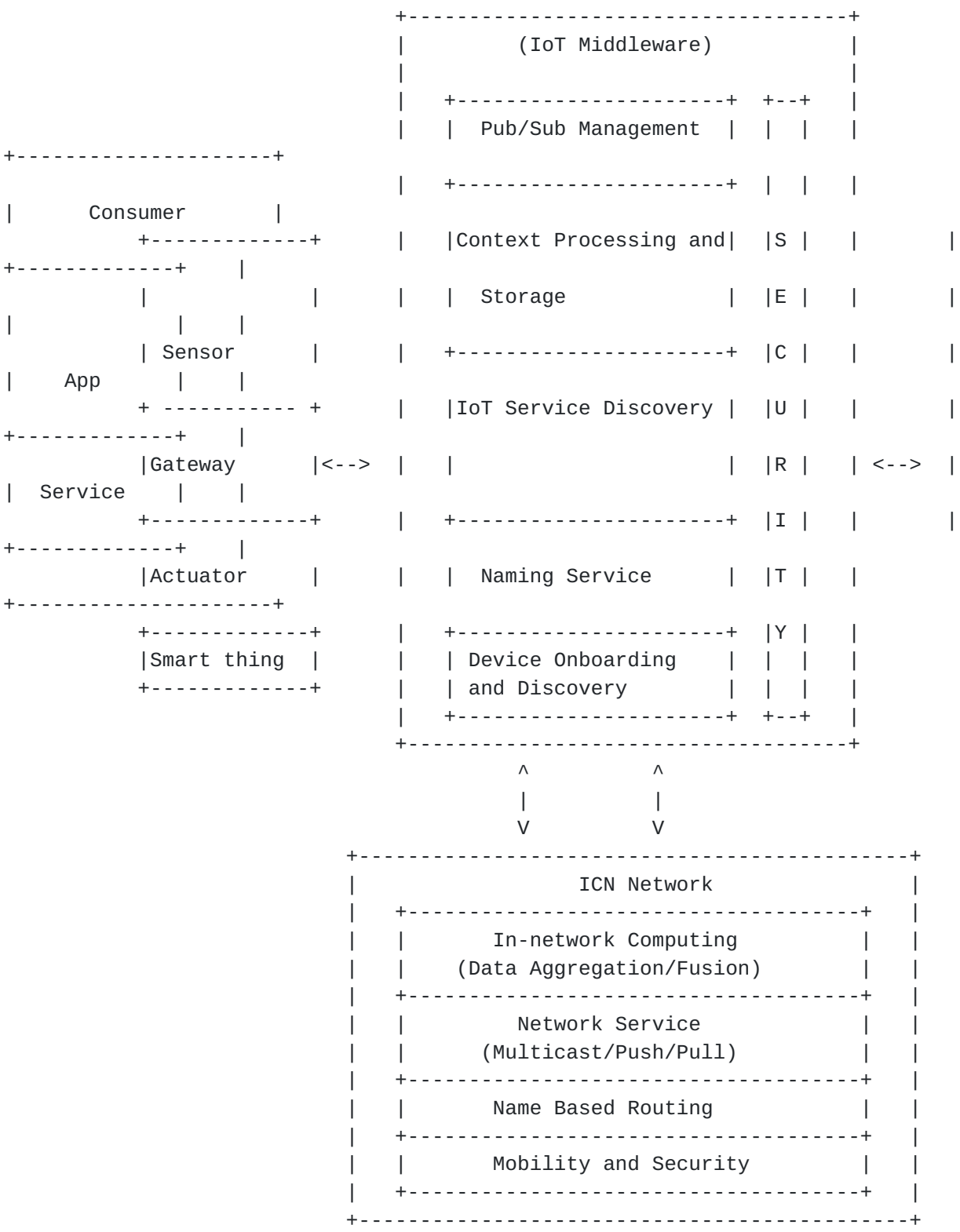

Figure 3: The ICN-IoT Middleware Functions

4. ICN-IoT Middleware Functions

For each of these middleware functions we highlight what these function achieve, advantages an ICN architecture enables in realizing each function, and provide discussion of how the function can be realized considering two ICN protocols i.e. NDN [6] and MobilityFirst (MF) [5].

Please note most of these middleware functions are implemented on unconstrained aggregators, LSGs and the IoT servers, only very limited functions (mainly for device discovery and naming service) are implemented on resource-constrained ES, while unconstrained devices within an aggregator can execute more functions such as service discovery.

4.1. Device Onboarding and Discovery

In the literature several works do not differentiate between device onboarding and device discovery. In this draft, we make a distinction. The objective of onboarding is to connect new devices to the rest and enable them to operate in the ecosystem. Every entity should be exposed to its direct upstream neighbor and may be another embedded system or aggregator. Specifically, it includes the following three aspects: (1) a newly added ES should be exposed to its neighbor (ES or aggregator) and possibly to its LSG, AM, and the IoT server; (2) a newly added aggregator is exposed to its LSG, and possibly to its neighbor aggregators; (3) a newly added AM should be exposed to the IoT server and the LSG; and (4) a newly added LSG should be exposed to the IoT server. Device discovery serves two functions: 1) it is used in the context of discovering neighboring ESs to form routing paths, where existing mechanims can be use; 2) for device onboarding, on which we focus here. During onboarding, the ES passes its device-level information (such as manufacturer-ID and model number) and application-level information (such as service type and data type) to the upstream devices. In the NDN architecture (and other name-based approaches), there is no need to identify the devices with IDs. But, if the device is required to have a globally unique ICN ID, it can be provided one by the naming service (described in Section 4.3), and recorded by the LSG (and possibly the aggregator and the IoT server). As part of the device discovery process, each ES will also be assigned a local network ID (LID) that is unique in its local domain. Then the device will use its LID for routing within the local domain (i.e. between ESs and the aggregator) because the globally unique ICN ID associated with a device is quite long and not energy efficient for constrained IoT devices. One approach to generate a short LID is to hash its persistent ID. In the name-based ICN approaches where device identity is not needed, a device can publish within the name scope of the aggregator (e.g., /iot/agg1/dev10 for Device numbered 10 under aggregator numbered 1). In most IoT systems, devices interact with the aggregator for data or information processing or aggregation, hence there is no direct communication between devices under an aggregator. If in some set-up devices under the aggregator need to communicate with each other a scalable mechanism is to allow direct neighbors to communicate with each other while others communicate through the aggregator.

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 sharing the same wireless resources. 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 by isolation. 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.

4.2. Detailed Discovery Process

A device can be an embedded device, a virtual device, a process, or a service instance such as a sensing service. We assume that the device has pre-loaded secure keys. Specifically, we consider both resource-constrained devices and resource-rich devices, and assume that the pre-loaded secure keys are symmetric keys or passwords for the former, while the asymmetric key pair (public key certificate and the corresponding private key) for the latter.

Below we discuss the detailed device discovery process considering both resource-constrained devices and resource-rich devices. As assumed for the former there is a mechanism for either securely pre-loading symmetric keys or passwords, while for the latter asymmetric key-pair using the public key infrastructure and certificate are used to exchange/generate the symmetric key (denoted as SMK_{device}). We note that the use of asymmetric keys follows the standard PKI procedures with the the use of either self-certificates, certificates generated by a local (or domain specific) or global authority. The bootstrapping of the constrained devices with symmetric keys can be performed in several ways. As mentioned, pre-loading the device with keys before shipping is one approach, or the symmetric keys can be loaded by the administrator (or home owner or site-manager) at the time of bootstrapping of the device on-site. The approach is based on the level of trust and the threat model in which the system operates. We also note that with ongoing research constrained devices are becoming increasingly powerful and new low-cost and computation based PKI approaches are being proposed. In the future, constrained devices may be able to also use PKI mechanisms for creating symmetric keys. In addition, we assume that there is a local authentication service (AS) that performs authentication, authorization and transient action key distribution. The local authentication service is a logical entity that can be co-hosted at the LSG or IoT server. The location of the AS may be informed by

efficiency, security, and trust considerations. The design offloads the complexity to the local AS and simplifies the operations at the devices. Mechanisms can be devised for authenticating and onboarding a device onto the IoT network even if the device does not trust its neighbors and the aggregator using the AS. This can be done by having the key SMK_{device} shared with an AS, which can be communicated this information during device bootstrapping. The AS is used to authenticate the device in the network. The mechanism can be extended for the device to authenticate the network it is connecting to as well [10].

The general steps for device discovery assuming pre-shared symmetric keys are as follows :

- New device polling: The configuration service periodically sends out messages pulling information from new devices. Then the device can communicate its manufacturer ID to the aggregator who forwards the message to the AS. The AS will send back a discover-reply, with a nonce encrypted with SMK_{device} and another nonce to be used by the device in the reply; this message is forwarded back to the device. The device decrypts the message obtains the nonce, and encrypts a concatenation of the two nonces with SMK_{device}, which it sends back to the AS. The AS decrypts the content again and authenticates the device. Then it creates a symmetric key to be shared between the aggregator and the device and encrypts a copy of the key with a symmetric key shared by the aggregator and the other copy with SMK_{device} shared with the device and replies back to the device. The reply message reaches the device via the aggregator and they both receive the key to perform secure communication. In NDN, this process can be initiated by the configuration service running on the LSG, which periodically broadcasts discovery Interests (using the name /iot/agg1/discover message) or the discovery can be initiated by the new device in the network which can send a discover message using its globally unique ID (e.g., manufacturer ID). If no authentication of the device's identity is required, then the discover message can be forwarded to the aggregator, which can reply with its namespace and a locally unique ID for the device, say dev10, so the namespace for the device is /iot/agg1/dev10). Or the globally unique ID of the device can also be used as the locally unique ID. This mechanism does not preclude the communication between the device and the aggregator going over a multi-hop path consisting of other IoT devices that are already on-boarded. If device authentication is required, In MF, we can set a group-GUID as the destination address, and the configuration service issues a polling via multicasting. Once the new device enters the IoT domain and receives the polling message, it sends a association request (AssoReq) message, including its manufacture

ID, or ICN ID (if it has been assigned one before), its network and security capabilities, and a message ID which is used for the further message exchange between the new device and aggregator. If the device is already assigned a symmetric key, it can use the symmetric key to communicate with the LSG (the ID can be transmitted in clear or by using a pseudonym [17]) to facilitate quick identification by the LSG. If the device can use the PKI, a symmetric key can also be generated. After the aggregator receives the AssoReq from the new device, it will request the LSG naming service to issue a local LID for this new device. The aggregator shall send an Association Reply (AssoRep) message to the new device, which includes the message ID copied from the previous AssoReq message from the new device to indicate this association reply is for this new device, the selected authentication method according to the new device security capabilities, the assigned LID. The AssoRep is sent to the new device via a specific multicast group (as the new device does not have a routable ID yet at this moment). The LID is a short ID unique in the local IoT domain, and is used for the routing purpose in the local domain. This specification will not limit the format of the LID and the method to generate a LID. If authentication is not required, the device discovery is completed and the device can communicate with the aggregator using the LID. If the Authentication is required, this LID is blocked by the aggregator from passing general data traffic between two devices until the authentication transaction completes successfully with the local authentication service. The unauthenticated LID can only send traffic to the authentication service. The aggregator forwards the traffic between the device and the local AS. The aggregator may also implement the policy to regulate the amount of traffic to be sent by an unauthenticated LID to mitigate the DoS attack. If the authentication is required, the following steps shall be performed.

- Mutual Authentication: The mutual authentication allows only authorized device to register and use the network, and to provide the device with assurance that it is communicating with a legitimate network. If the authentication is required in the AssoRep, the device shall send a Authentication Request (AuReq) message to the aggregator using the selected authentication method. The AuReq is signed with the pre-loaded SMK{device} for authentication. The aggregator forwards the AuReq to the local AS. The local AS performs authentication locally or contacts a third-party AS according to the authentication method. If the authentication is successful, the local AS generates a master symmetric key SMK{device, aggregator} for the communications between the device and the aggregator. It sends Authentication Reply (AuRep) with master SMK{device, aggregator} to the device

via the aggregator. The master SMK{device, aggregator} is protected with the pre-loaded SMK{device}. The local AS also sends a copy of master SMK{device, aggregator} to the aggregator through the secure connection between the local AS and the aggregator. This same approach will work equally well in the NDN architecture as well.

- Key generation and distribution: Once the master SMK{device, aggregator} is placed on the device and aggregator, the session keys (AKs) and group keys (GTKs) are generated and placed on the device and the aggregator for unicast and multicast communications, respectively, using the master SMK{device, aggregator}.

- Protected Data Transfer: The session keys (AKa and AKe) are used for message integrity and data confidentiality, respectively, which can be renewed periodically. The renewal can happen using key generation functions with the shared secrets and some nonces used for generating the new session keys [10].

The other case is when devices have sufficient resources to run asymmetric keys. That is, the device is pre-loaded with a manufacture ID, a pair of public/private keys (PK_{device}, SK_{device}) and a certificate which binds the device identity and public key. In this case, we also go through the above three steps, with the only difference being in the second step which is Mutual Authentication. To illustrate it using MF as the architecture, in this case, the AuReq message shall include the device certificate and a message authentication code signed by the device private key SK_{device}. The local AS will authenticate the device once receiving the AuReq. If the authentication succeeds, then the local AS will send the master SMK{device, aggregator} along with its certificate in AuRep. AuRep contains a MAC signed by the local AS private key. The mater SMK{device, aggregator} is encrypted using the device public key PK_{device}. SMK{device, aggregator} will be used for generation of AKs to ensure the integrity and confidentiality of future data exchange between the device and the aggregator.

4.3. Naming Service

The objective of the naming service is to assure that either the device or the service itself is authenticated, attempting to prevent sybil (or spoofing) attack [18] and that the assigned name closely binds to the device (or service). Naming service assigns and authenticates ES 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-persistent. IP addresses also have relatively poor scalability, due to their fixed structure. Instead, ICN separates names from locators, and assigns unique and persistent names to each ES, which satisfies the above requirements.

If a device needs a global unique name/ID, but does not have one, it may request the naming service to obtain one after it is authenticated. Alternatively, the IoT domain (LSG or aggregator) may determine ID (name) for an authenticated device is required based on the policy. The proposed naming process works as follows. After a device has been authenticated, it may request an ID from the naming service (or the aggregator, if it can give the device a locally unique name). It sends a ID request (IDReq) to the naming service or aggregator. If the aggregator can accept request to give a unique name to the device, it will do that. For instance, in NDN the device can create content within the aggregator's namespace. If the aggregator cannot then it can serve as the devices' proxy and sends the IDReq to the naming service at the LSG. The naming service assigns a ID to the device, which can be self-certified or a URI. . The naming service also generates a certificate, binding the ID/ public key with the devices' manufacture ID or human-readable name. The LSG sends the ID reply (IDRep) message to the aggregator that sends the IDRep to the device. The IDRep includes the ID certificate and the corresponding private key. The private key is encrypted and the entire message is integrity-protected with AK_{device} when the message is delivered to the device. Alternatively, if the LSG determines that an authenticated device requires an ID when the aggregator registers this device, it will contact the naming service to generate the ID, certificate, and corresponding private key for the device. It sends the ID information to the device. If the device already has a pre-loaded public key, the naming service may use this pre-loaded public key as the device's ID.

The LSG maintains the mapping between every devices' LID and the ID. When the LSG receives a message from the external network that is intended for a device within the domain, the LSG will translate the destination devices' ID (which is included in the message) to its LID and then route the message to the device using its LID. Similarly, when the LSG receives a message from within the domain, it will replace the source devices' LID with its ID and then forward the message to the next-hop router. Such a mechanism can ensure global reachability of any IoT device as well as energy efficiency for resource-constrained devices.

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

4.4. Service Discovery

Service discovery intends to learn IoT services that are hosted by one aggregator by its neighbor aggregators. The aggregators themselves learn service capability of the devices during the device discovery process or separately after authenticating (or during or after naming) them. The requirements for any discovery mechanism includes low protocol overhead (including low latency and low control message count), and discovery accuracy.

In today's IoT platforms, ESs, 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.

Service discovery is widely accepted as an essential element in pervasive computing environments. Many research activities on service discovery has been conducted, but privacy has often been ignored. While it is essential that legitimate users can discover the services for which they have the proper credentials, it is also necessary that services were hidden from illegitimate users. Since service information, service provider's information, service requests, and credentials to access services via service discovery protocols could be sensitive, it is important to keep them private. In [8], the authors present a user-centric model, called Prudent Exposure, as an approach designed for exposing minimal information privately, securely, and automatically for both service providers and users of service discovery protocols.

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 following procedures are performed: 1) 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 sent while discovery may require to learn multiple entities. Efficient discovery can be realized using exclusion via Selectors in the protocol or as an overlay protocol [7]; 2) 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.

As an example of an 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 the request containing the service name and certificate to 1234. In both cases, the AC hosting this service replies with acknowledgment if all conditions match.

As regards to secure multicast service request, it is possible to use the following solution in MF. In fact, especially in MF IoT, secured group GUID can be utilized, which may be owned by multiple hosts, hence conventional public/private key scheme may not be suitable for this case. For secure service discovery, a secured name needs to assigned to the service host. As an alternative, group key management protocol (GKMP) [31] can be adopted to resolve the issue above -- A naming service residing at LSG or IoT server (depending on application scope) generates a group public key that is used as group GUID for a service, then this group public/private keys pair is assigned to each Aggregator that hosts this service. The service hosting 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 because of the use of self-certified names by MF.

4.5. Context Processing and Storage

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 ES'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 ES 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 supports in-networking computing (e.g. using named functions [14] and computing layer in MF) 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 ESs, aggregators, and LSGs), including device-level attributes (such as location, data type, battery level, multiple interfaces, available cache size, etc), network-level attributes (such as ICN names , latency per interface, packet loss rates, etc.), and service-level attributes (such as max, min, average, etc.).

Secondly, we need to have means to expose ES/aggregator/LSG contextual attributes to the rest of the system, through centralized naming resolution service or distributed routing protocols.

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, network-level , and service-level contextual information.

Once the contextual requirements and the corresponding mappings are in place, then in-network caching can be leveraged to optimize overall network performance and system QoS. In an IoT network, cached data can be used to perform data aggregation, in-network processing, and quick turnaround for answering queries. In-network caching can also be used to store data that may be relevant to many nodes and has temporal popularity in a region, and the processed data to serve the users. The contextual requirements can help define in-network processing. This goes beyond the traditional way of aggregators doing the data gathering, processing, and reduction, but also moving computation to the data (also termed network functions). Network functions in essence serves to move the computation into the network for it to happen where the context and the information is available, with the results returned to the requester. In an ICN-IoT these functionalities can be easily incorporated at scale. A good use case for both in-network caching and processing is that of an IoT network of cameras working together to gather a complete field-of-view (FoV) of an area and transmit it for aggregation at the aggregator (may be implemented in the pub/sub model). A user could fire a query that involves processing on the gathered field-of-views at the aggregator (or some other node storing the FOV-data) to answer the query.

In-network processing can be implemented at the network level and application level. For example, a user may request the information regarding the maximum car speed in an area. With the network-level implementation, the user issues a request with a function expression /max(/area/carspeed)[14]. The network returns the user the computed results. This result can be obtained in two steps (a)name manipulation and orchestration of computation, an aggregator or LSG will check whether it already has the cached result for /max(/area/carspeed). If not, it will analyze the function expression, retrieve the function /max and the data /area/carspeed, and compute the result, or forward the request to another node for execution (b) Actual function execution for computation and data processing, that is, calculate /max(/area/carspeed). Alternatively, a user may issue a request /carspeed_service{max_car_speed in the area}[22]. The

request is sent to a car_speed service application. The car speed service processes application-level request, i.e. max_car_speed in the area. With the former approach, the data processing and result retrieval may be more efficient. However, the network, that is, the aggregators and LSGs should have a runtime execution environment at the network layer to understand and process the function expression logic. The later approach is simple and robust because the more complex function execution can be performed at the application layer using dedicated virtual machines.

4.6. 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 that inherently does not support the Pub/Sub mode naturally, there are a couple of approaches proposed to create a pub/sub model on top of NDN: namely COPSS [19] and persistent interests. COPSS integrates a push based multicast feature with the pull based NDN architecture at the network layer by introducing Rendezvous Node(RN). RN is a network layer logical entity that resides in a subset of NDN nodes. 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 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, 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 no 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.

MF takes advantage of Group-GUID to identify a service provided by multiple resources. This Group-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-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-x, then listen on GUID-x. The subscriber sends the request containing GUID-x through multicasting which ultimately reach the producers at the 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-GUID can be reused to group the consumers to further save bandwidth using multicast.

Another approach to extend the NDN architecture to enable pub/sub is for the subscriber(s) to send Interests that are identified as long-term Interests [11]. The Interests do not expire from the PIT of the intermediate forwarding routers on the path from the publisher to the subscribers. Each time the publisher creates a new content, it publishes it into the network, the content reaches each subscriber along the reverse-path from the producer based on the faces stored in the PIT entry. However, the Interest is not removed from the PIT entry. This allows the creation of a multicast tree rooted at the

publisher, reaching every subscriber and enables the pushing of content from the publisher to the subscribers as needed.

With a pub/sub framework, important considerations should be given towards user registration and content distribution.

User Registration: A user, who wants to access/subscribe to a service, has to perform the registration operation by sending information that depends on the specific application domain to the IoT server. The information can be secured with the help of the PKI infrastructure. Upon successful registration the IoT server securely transmits an identifier, a user signature key SK_{user} (to be used to sign messages), a user encryption key EK_{user} (to communicate data confidentially), and an access password to the user in an encrypted message. Upon reception of the message, the user accesses the system to modify his/her password (function changePassword). With respect to existing secure application-layer solutions, a further benefit of the presented approach is the introduction of a second level of security, represented by the use of a temporary password (immediately replaced) and a couple of keys (signature SK_{user} and encryption EK_{user}), which is well suited for the heterogeneous and distributed IoT environment.

Content Distribution: In literature, there are some solutions able to guarantee content security [9] [15][12]. In fact, the work presented in [9] [12] aims to ensure a high availability of the cached data only to legitimate users. The authors design a security framework for ICN able to deliver trusted content securely and efficiently to legitimate users/subscribers. They assume that the content is encrypted by the content provider, either at the servers or in the content distribution network (if it is trusted), by means of a popular symmetric key encryption algorithm. A group of contents may be encrypted using the broadcast encryption key, which only legitimate users can decrypt. The goal is to ensure that the encrypted content cannot be used by an entity that is not a legitimate user/customer. The authors achieve this goal by guaranteeing that only a legitimate user can obtain the symmetric key to decrypt the content, whereas a fake or a revoked user cannot. In this way, the framework does not require any user authentication, for example by an online server, each time a content is requested. Instead, Zhang et al. in [15] consider trust and security as built-in properties for future Internet architecture. Leveraging the concept of named content in recently proposed information centric network, the authors proposed a name-based trust and security protection mechanism. Their scheme is built with identity-based cryptography (IBC), where the identity of a user or device can act as a public key string. Uniquely, in named content network such as content-centric network (CCN), a content name or its prefixes can be used as a public

identity, with which content integrity and authenticity can be achieved by means of IBC algorithms. The trust of a content is seamlessly integrated with the verification of the content's integrity and authenticity with its name or prefix, instead of the public key certificate of its publisher. In addition, flexible confidentiality protection is enabled between content publishers and consumers. For scalable deployment purpose, they further propose to use a hybrid scheme combined with traditional public-key infrastructure (PKI) and IBC. Keeping in mind the available solutions, in our proposed method, the device sends to the aggregator its ICN name, its ID encrypted with its signature key SK_{device} and the data encrypted with its own action key AK_{device}, in order to guarantee confidentiality and integrity. The action key AK_{device} has been distributed during the device discovery (see Section Device discovery). The aggregator is able to decrypt the data using the corresponding action key AK_{device}, stored with the device ID, the signature key SK_{device} and the device ICN name obtained during the name service (see Section Name service), in particular the aggregator uses the device name for identifying the related action key AK_{device} (function contentDecryption). Note that the data are encrypted only if it is required by the application domain (i.e., some contexts may not have any security requirements - in this case the function contentDecryption is not applied). As regards the content delivery towards a user who subscribes to a service, the ICN IoT server transmits to the user the data encrypted with the user action key AK_{user} in order to guarantee security and privacy, if it is a requirement of the application domain. The user decrypts the received data using his/her action key AK_{user}(function contentDecryption). In such a situation, the services are treated as multiple-unicast ones, since the aggregator has to use different keys for different devices. In order to address a multicast approach, a group signature key system may be adopted, as in MF approach.

4.7. Security

This spans across all the middleware functions. Generally speaking, the security objective is to assure that the device that connects to the network should be authenticated, the provided services are authenticated and the data generated (through sensing or actuating) by both devices and services can be authenticated and kept privacy (if needed). To be specific, we consider the approach to secure device discovery, naming service and service discovery, because other services, such as pub/sub management and context processing and storage, can be properly secured according to application-specific demands.

5. Support to heterogeneous core networks

5.1. Interoperability with IP legacy network

Interoperability between the IP legacy network and ICN networks is an important property that the middleware must meet in order to ensure the co-existence and gradual migration from the today IP-based technologies and protocols. This could provide a market strength to the deployment of the ICN technologies. To this end, the Internames architecture [21][22] provides an embedded capability to manage different network domains (or realms), and to support legacy web applications and services. In this sense, a crucial role is played by the Name Resolution Service (NRS), whose functionalities can decouple names from network locators as function of time/location/context/service, and provide ICN functionalities in IP networks. By integrating these functionalities on appropriated nodes a distributed database is created to ease internet-working among heterogeneous protocol stacks in the core network.

5.2. Named protocol bridge

In an heterogeneous network, composed of different ICN networks and legacy IP-based networks, interoperability can be pursued, thanks to the name-to-name primitives. To this end, a name-based protocol bridge could be deployed at different points of the heterogeneous core network so as to provide bridging functionalities at the border of different administered network domains. In order to correctly forward the message through the network, the NRS node could aid the name-based protocol bridge providing inter-domain routing functionalities.

5.3. Inter-domain Management

In heterogeneous networks the IoT server has to strictly cooperate with the NRS nodes in the core network in order to build a virtual network topology to efficiently support Req/Res and Pub/Sub functionalities. The IoT Server could provide the names of the internal resources to the NRS, so that when the internal network changes, hence the connectivity to the resources. This ensures that the NRS database is always synchronized and updated with every IoT subsystems. In order to support Req/Res and Pub/Sub services management efficiently in an heterogeneous core network, the IoT Servers of the different administered domains have to strictly cooperate with the NRS nodes in the core network. This is to provide internal information of their own administered domain, such as the names and or the locators of the internal resources. When the internal network changes, the status of the resources are synced in

order to maintain an accurate database of the virtual network topology view comprising of various IoT subsystems.

6. Informative References

[1] Zhang, Y., Raychadhuri, D., Grieco, L., Baccelli, E., Burke, J., Ravindran, R., Wang, G., Lindgren, A., Ahlgren, B., and O. Schelen, "Design Considerations for Applying ICN to IoT", draft-zhang-icnrg-icniot-01 (work in progress), June 2017.

[2] Grassi, G., Pesavento, D., and Giovanni. Pau, "VANET via Named Data Networking.", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) , 2014.

[3] ICN based Architecture for IoT - Requirements and Challenges, ICN-IoT., "https://tools.ietf.org/html/draft-zhang-icnrg-icniot-requirements-01", IETF/ICNRG 2015.

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

[5] NSF FIA project, MobilityFirst., "http://www.nets-fia.net/", 2010.

[6] NSF FIA project, NDN., "http://named-data.net/", 2010.

[7] Ravindran, R., Biswas, T., Zhang, X., Chakrabort, A., and G. Wang, "Information-centric Networking based Homenet", ACM, Sigcomm, 2013.

[8] Feng, Z., Mutka, M., and L. Ni, "Prudent Exposure: A private and user-centric service discovery protocol", Pervasive Computing and Communications, 2004, Proceedings of the Second IEEE Annual Conference on. IEEE, 2004.

[9] Misra, S., Tourani, R., and N. Majd, "Secure content delivery in information-centric networks: design, implementation, and analyses", Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013.

[10] Mick, T., Misra, S., and R. Tourani, "LASeR: Lightweight authentication and secured routing for NDN IoT in smart cities", arXiv:1703.08453v1 (in submission in IEEE IoT Journal).

[11] Tourani, R., Misra, S., Mick, T., and S. Biswal, "iCenS: An information-centric smart grid network architecture", In IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 417-422, 2016.

[12] Misra, S., Tourani, R., Mick, T., Brahma, T., Biswal, S., and D. Ameme, "iCenS: An information-centric smart grid network architecture", In IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 417-422, 2016..

[13] Liu, H., Eldaratt, F., Alqahtani, H., Reznik, A., De Foy, X., and Y. Zhang, "Mobile Edge Cloud System: Architectures, Challenges, and Approaches", IEEE Systems Journal, pp. 1-14, DOI: 10.1109/JSYST.2017.2654119, Dec. 2016..

[14] Sifalakis, M., Kohler, B., Scherb, C., and C. Tschudin, "An information centric network for computing the distribution of computations", In Proceedings of the 1st ACM International conference on Information-Centric Networking, pp. 137-146, 2014..

[15] Zhang, X., "Towards name-based trust and security for content-centric network", Network Protocols (ICNP), 2011 19th IEEE International Conference on. IEEE, 2011.

[16] Nikander, P., Gurtov, A., and T. Henderson, "Host identity protocol (HIP): Connectivity, mobility, multi-homing, security, and privacy over IPv4 and IPv6 networks", IEEE Communications Surveys and Tutorials, pp: 186-204, 2010.

[17] Misra, S. and G. Xue, "Efficient anonymity schemes for clustered wireless sensor networks", International Journal of Sensor Networks, volume 1, number 1/2, pp: 50-63, 2006.

[18] Newsome, J., Shi, E., Song, DX., and A. Perrig, "The sybil attack in sensor networks: analysis and defenses", IEEE, IPSN, 2004.

[19] Jiachen, C., Mayutan, A., Lei, J., Xiaoming, Fu., and KK. Ramakrishnan, "COPSS: An efficient content oriented publish/subscribe system", ACM/IEEE ANCS, 2011.

[20] Marica, A., Campolo, C., and A. Molinaro, "Multi-source data retrieval in IoT via named data networking", ACM ICN Siggcomm, 2014.

[21] Blefari-Melazzi, A., Mayutan, A., Detti, A., and KK. Ramakrishnan, "Internames: a name-toname principle for the future Internet", Proc. of International Workshop on Quality, Reliability, and Security in Information-Centric Networking (Q-ICN), 2014.

[22] Piro, G., Signorello, S., Palatella, M., Grieco, L., Boggia, G., and T. Engel, "Understanding the Social impact of ICN: between myth and reality", AI Society: Journal of Knowledge, Culture and Communication, Springer, pp. 1-9, 2016.

[23] Nelson, S., Bhanage, G., and D. Raychaudhuri, ""GSTAR: generalized storage-aware routing for mobilityfirst in the future mobile internet", Proceedings of the sixth international workshop on MobiArch, pp. 19--24, 2011.

Authors' Addresses

Prof.Yanyong Zhang
WINLAB, Rutgers University
671, U.S 1
North Brunswick, NJ 08902
USA

Email: yyzhang@winlab.rutgers.edu

Prof. Dipankar Raychadhuri
WINLAB, Rutgers University
671, U.S 1
North Brunswick, NJ 08902
USA

Email: ray@winlab.rutgers.edu

Prof. Luigi Alfredo Grieco
Politecnico di Bari (DEI)
671, U.S 1
Via Orabona 4, Bari 70125
Italy

Email: alfredo.grieco@poliba.it

Sicari Sabrina
Universita degli studi dell Insubria
Via Mazzini 5
Varese, VA 21100
Italy

Email: sabrina.sicari@uninsubria.it

Hang Liu
The Catholic University of America
620 Michigan Ave., N.E.
Washington, DC 20064
USA

Email: liuh@cua.edu

Satyajayant Misra
New Mexico State University
1780 E University Ave
Las Cruces, NM 88003
USA

Email: misra@cs.nmsu.edu

Ravi Ravindran
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ravi.ravindran@huawei.com

G.Q.Wang
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: gq.wang@huawei.com