A Comparative Study of MobilityFirst and NDN based ICN-IoT Architectures

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

***Abstract*—To develop unified IoT platforms where objects can be made accessible to applications across organizations and domains, popular solutions are based on client-server overlays on today's Internet. These solutions, however, inherit the inefficiencies of the current Internet – especially in terms of mobility, scalability, and communication reliability. To address this problem, we propose to build the unified IoT platform leveraging the salient feats of Information-Centric Network (ICN) architectures, which we call ICN-IoT. Specifically, we explore two ICN architectures – MobilityFirst and NDN – to support IoT, and refer to them as MF-IoT and NDN-IoT, respectively. Through detailed simulations, we find that though these two architectures fare comparably, MF-IoT incurs lower control overheads.**

I. Introduction

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 are rapidly deployed that will make their data and services accessible to general Internet applications. In such a unified platform, physical resources can be accessed over Internet and shared across many applications.

Building a unified IoT platform, however, poses a set of unique challenges on the underlying network and systems. Firstly, it needs to support a large number of networked objects – Cisco predicts there will be around 50 Billion IoT devices (sensors, RFID tags, actuators, etc) on the Internet by 2020 [1] – and many of these objects are mobile, for e.g transport systems. The underlying platform needs to scale smoothly with respect to metrics like response time, throughput, resolution and routing scalability. Secondly, IoT devices will have heterogeneous means of connecting to the Internet, and often have severe resource constraints, e.g., constrained resources in power, computing, storage, bandwidth. Thirdly, interactions between the applications and objects are often private, contextual,real-time and dynamic, requiring strong security and privacy protections. Finally, a unified IoT platform should be able to provide seamless services in the presence of device mobility.

Current approaches towards a unified IoT platform are mostly based upon Internet overlays, whose inherent inefficiencies hinders the platform from satisfying the challenges outlined earlier, particularly in terms of scalability and mobility [8]. In recent years, in order to address the inefficiencies of today's Internet, Information-Centric Network has been proposed. ICN identifies a network object (including a mobile device, content, or service) by application centric name instead of its IP address, and adopts a hybrid name/address routing, lending itself to supporting the unified IoT platform. In this paper, we propose to build a unified IoT platform using ICN (illustrated in Figure 1), in which 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. We call the resulting network architecture as ICN-IoT. Specifically, we discuss and evaluate two different ICN architectures – MobilityFirst [8] and NDN [10] – and refer to them as MF-IoT and NDN-IoT respectively.

In this paper, we discuss the detailed design of MF-IoT and NDN-IoT, focusing on their service discovery and pub/sub model. For evaluation purposes, we consider two realistic IoT applications scenarios: a smart building scenario and a smart campus bus scenario, with the former representing stationary IoT devices while the latter focusing on mobile IoT devices. We have also compared the performance of these two approaches through detailed simulations. Our simulation results show that these two architectures have comparable delays and throughput, while MF-IoT incurs less overhead in terms of both routing table size and the number of control messages.

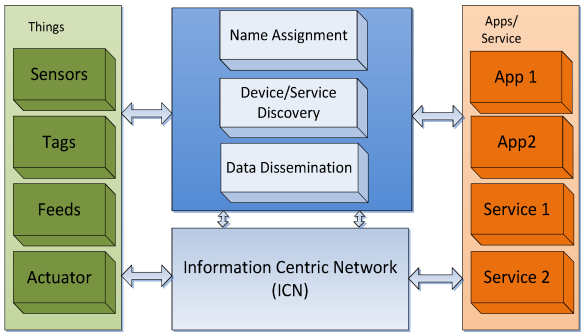

Fig. 1: ICN-IoT architecture

II. ICN Background

In this section, we provide a brief overview of the two ICN architectures.

QSHINE 2014, August 18-20, RHODES, Greece

DOI 10.4108/icst.qshine.2014.256488

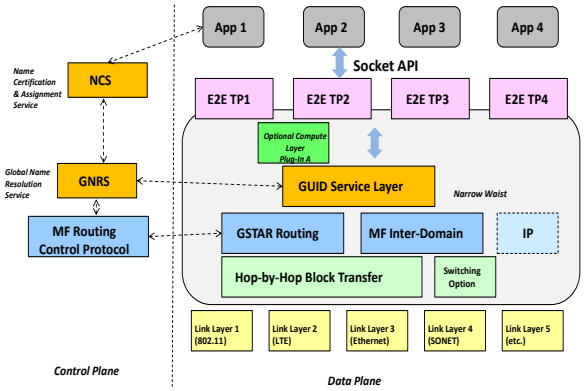

Fig. 2: MobilityFirst protocol stack

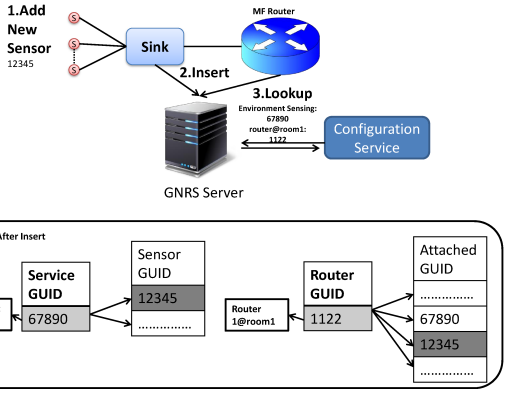

Fig. 3: Three steps for discovery

A. *Named Data Networking (NDN) Architecture*

NDN adopts a receiver-driven architecture, engaging two types of packets : Interest and Data. A NDN transaction starts with the user (or data consumer) issuing an Interest packet. When the Interest arrives at the node that own this piece of content (i.e., data producer), the data producer will reply with a Data packet along the reverse path to the consumer. A NDN router has three key data structures: (1) Forwarding Information Base (FIB) that contains forwarding information such as content prefix and outgoing face mapping, (2) Pending Interest Table (PIT) that maintains the set of pending Interests waiting for Data packets, and (3) Content Store(CS) that stores Data packets. NDN adopts hierarchical content names which are suitable for aggregation.

B. *MobilityFirst (MF) Architecture*

MF assigns a flat, globally unique identifier (GUID) [7] to every network object – the separation of the network object's identifier and its network address (NA) allows MF to support dynamic address binding, hence good mobility support. Figure 2 shows the MF core network architecture. MF has the following three key components. The first component is Global Name Resolution Service(GNRS), a centralized service that manages all the dynamic $GUID \rightarrow NA$ mappings. When a device changes its network association due to mobility, its $GUID \rightarrow NA$ mapping must be updated in the real time [9]. The second main component is hybrid GUID/NA routing, in which a MF router can make routing decisions based on either NA or GUID, in a hop by hop manner. The third main component is delay-tolerant networking, in which the storage in each MF router provides the capability of caching data packets.

III. Device Discovery in ICN-IoT

Device discovery, one of the key features of an IoT system, can be easily achieved from the adoption of ICN. In traditional IP-based IoT systems, in order to identify IoT devices using application-readable names (such as those derived from manufacturer-assigned IDs), system middleware needs to implement the mapping from names to their IP/PORT addresses, while the IP transport layer remains oblivious of application delivery. The resulting middleware is thus complex and requires a significant amount of development effort, which in turn limits the level of mobility the system can support. In contrast, while ICN's network layer uses such names, the middleware is not involved in name translation.

Device Discovery in NDN-IoT: The NDN-based lighting system discussed in [3] demonstrates a practical use case for NDN-IoT. In [3], it is assumed that each lighting device comes with a manufacture assigned public key, a shared secret for initialized authorization, and a well-known hierarchical NDN name, such as "/ndn/lighting" for easy discovery. When a new object (e.g., a lighting fixture) connects to the system, it registers itself and publishes its public key. The configuration manager (CM) authenticates the object using the share secret key and assigns a name (e.g., "/ndn/lighting") to the application. The CM periodically expresses its interest using a well-known service name, such as "/ndn/lighting", to locate any available device on a broadcast channel. The IoT device attached to the object then replies the Interest with a Data packet.

Device Discovery in MF-IoT: The design of MF-IoT is centered around the overloading of GUIDs. First, each device has a device GUID, which can be derived from its manufacturer serial number. Further, the IoT end device access point(EDAP), e.g., a sink node (of a local sensor network), a fixture or even a mobile phone, uses a specific service GUID to identify their service type, such as environmental sensing, light control or health sensing. Next, let us consider what happens when a new device is attached to a sink – first it will announce its device GUID, i.e., d, and the corresponding EDAP with service GUID s will then insert a "device to service" mapping $d \rightarrow s$ to GNRS. If a configuration service (CS) running on the remote server queries GNRS with s, the new device's GUID (d) will be included in the query result.

Figure 3 illustrates the above-mentioned discovery process in MF-IoT, in which the device GUID is 12345, and the service GUID is 67890.

IV. Data Publish/Subscribe in ICN-IoT

Data Publish/Subscribe (Pub/Sub) is an important function for ICN-IoT, responsible for resource sharing and management. In traditional IP network, most of the IoT platforms provide a centralized server to aggregate all IoT devices, data and services, publish them to the web portal, and manage the subscription membership. While such a centralized architecture ensures data/resource availability, it renders poor scalability and high bandwidth consumption due to the high volume of control and data exchange as a result by such an architecture. In ICN-IoT, we thus consider a decentralized pub/sub model.

A. Pub/Sub in NDN-IoT

NDN is a Pull-based architecture, where the Pub/Sub model is not naturally supported, but it has been discussed in COPSS [4]. It integrates a push based multicast feature with the pull based NDN architecture at the network layer by introducing Rendezvous Nodes (RN). RN is a logical entity resided on a NDN node. The data publisher first forwards a Content Descriptor (CD) as a snapshot to the RN. RN then maintains a subscription table, and receives Subscribe message (similar to Interest Packets in NDN) from subscriber nodes. The data publisher just sends the content using Publish packet by looking up FIB instead of PIT. If the same content root is required by multiple subscribers, RN will deliver one copy of content downstream, hence reduced bandwidth consumption.

B. Pub/Sub in MF-IoT

Publishing in MF-IoT is rather straightforward: the Configuration Service directly publishes new devices to the IoT server, and in the rest of this section, we will focus on the subscription part, in which the objective is to maintain the convenience of a traditional centralized Pub/Sub model while minimizing the bandwidth consumption. Towards this objective, we consider two communication models – push mode and pull mode – for sensor data retrieval in MF-IoT. The choice between these two options is driven by application requirements. In general, the push mode allows immediate synchronization of system state, hence suitable for mission critical applications; the pull mode suffices for applications that do not have such stringent requirements, but can achieve fast enough synchronization by dynamically adapting pull frequency.

Basic Push Mode: The push mode is suitable for event-driven data delivery, where data is sent to applications whenever specific events are detected. Here, we introduce a new type of GUID – subscription-GUID – as well as the mapping from subscription-GUID to application-GUID that maps the subscription-GUID to the list of applications that subscribe to the service. As such, the sink node does not need to know each individual subscribing application's GUID, but can simply specify the subscription-GUID in the destination-GUID field in a MF packet header.

On Demand Pull Mode : Many IoT applications do not subscribe to specific sensors/sources, but to a certain type of IoT service which may be offered by any sensor in a group. In this case, the pull mode is more suitable.

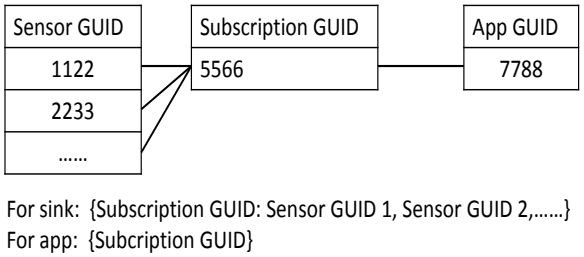

Fig. 4: Mapping relationship

In MF, to associate a group of sensors with a subscriber application, we make use of the subscription-GUID - This allows to send data requests to multiple sensors (connected to different sinks) through a single subscription-GUID. The challenge in this approach is however, when the sink receives a request, it is unaware of which sensor is needed. To address this challenge, the IoT server needs to provide additional information to the sink node. As shown in Figure 4, we establish a relationship between the sensors and the application in the IoT server. Similar to the basic push model we have discussed above, the IoT server assigns a message with the subscription-GUID that is mapped to multiple device GUIDs in the subscription, such that the sink node knows the sensors to be accessed.

V. IoT Application Scenarios

There are a wide range of IoT applications, each with varying system requirements. To evaluate the performance of ICN-IoT, we choose two different application scenarios in the scope of a smart campus: (i) a building management system (BMS) that has a large number of sensors, sinks, and actuators (referred to as ICN-BMS); and (ii) a school bus system that requires mobility (referred to as ICN-BUS).

A. ICN-based Building Management System (ICN-BMS)

BMS is responsible for controlling complex in-building ecosystems such as climate control, security monitoring, smoke detection, etc. Most of these systems run on heterogeneous communication protocols, and our objective is to inter-connect them with homogeneous network protocols and enable the reachability through global, persistent names or identifiers. Most of the traffic in BMS systems is generated by the sensors and delivered to the BMS server, hence largely a data collection system. For control purposes, the BMS server communicates with actuators whenever one or more environment parameters reach the pre-set threshold(s) or due to operations by an administrator. Alternatively, the BMS server can pull sensor data at a certain frequency, but this operation mode might consume unnecessary bandwidth and incur longer round-trip delays, especially when the number of sensor become large. Therefore, we prefer the push-based control mode.

System Architecture: In ICN-BMS, we assume four types of network devices: sensors, routers, BMS server, and actuators.

Property	Value
Object Identifier	Analog Output #1
Object Name(MF/NDN)	GUID (1234) **OR** /ndn/TempCtrl/huawei/building_a/1001/tempstat
Object Type	Thermostat Temp
Present Value	70.0
Status Flag	FAULT/OVERRIDEN /IN_ALARM/OUT_OF_SERVICE
Event Status	Normal
Out of Service	False
Units	Degrees-Fahrenheit

Fig. 5: Data structure of BACNet over ICN

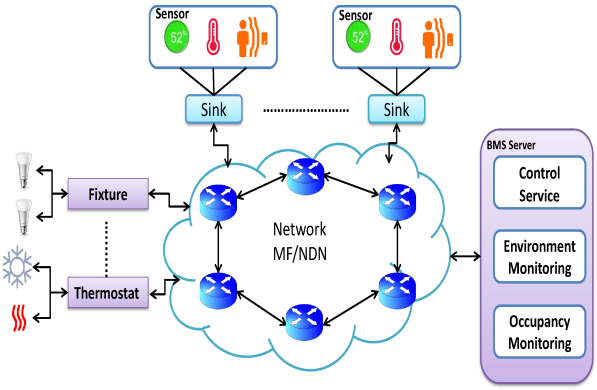

Fig. 6: BMS over MF/NDN

Sensors are attached to ICN-enabled sinks, and the associated objects can either be updated to BMS server by the sink directly, or be pulled by remote data consumers on demand. The actuators are also ICN enabled – e.g. thermostat or light fixture – which provide a network interface for control purposes. The system architecture of ICN-BMS is illustrated in Figure 6, in which we include three example building automation services: environment monitoring service, occupation monitoring service and control service.

As a reference BMS system architecture, we refer to Building Automation and Control Network (BACnet) protocol [5], which has been standardized by American Society of Heating, Refrigerating and Air Conditioning Engineers (ASHRAE), and adopted in many commercial BMS deployments. Based on BACnet, we define ICN objects following the format shown in Figure 5. Although BACnet is an application layer protocol that requires middleware to provide a mapping from names to network addresses (NA), we substitute this identifier with an ICN name so that BACnet objects can be made accessible via the ICN network layer.

B. ICN-Based School Bus System (ICN-BUS)

The second scenario we consider is an intelligent campus bus system that provides both vehicle-to-infrastructure (V2I) and infrastructure-to-user interfaces for real time status updates. Today the most common network device on buses is Mobile Data Terminal (MDT), a GSM-based communication module supporting data exchange between the control center and the bus using the SMS service. Most implementation today is based on the Controller Area Network bus (CANbus) protocol [6] which allows on-vehicle sensors to communicate with each other without a host computer. Also, sensor data such as velocity, seat occupation, and GPS obtained by CANbus can be directly transmitted to the infrastructure via MDT. However, the SMS-centric communication model has obvious shortcomings, namely limited media support and large delays. In ICN-BUS, we assume that a smart campus is mostly covered by WiFi, including all bus routes. Thus, we can migrate the system to ICN for better performance.

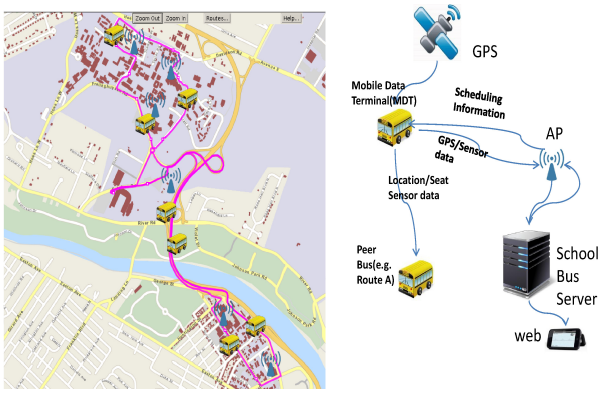

Fig. 7: Route A & school bus system architecture

System Architecture: In ICN-BUS, shown in Figure 7, we use ICN-based MDT on each bus. We assume three types of sensors: velocity sensor, seat sensor, and GPS. Similar to our ICN-BMS design, we adopt a centralized ICN-BUS server that handles updates from MDT, as well as sends notification to one or more buses. In order to support applications, we also establish peer-to-peer communications in a vehicle-to-infrastructure (V2I) manner.

The MF-based ICN-BUS combines the mobility and pub/sub features of MF. Let us assume that drivers in Route A(shown in Figure 7) are interested in the position and occupancy of other buses on the same route. They subscribe to the"Route A Data Sharing from Bus#1" service, and obtain a subscription GUID (*subGUID*) from the ICN-BUS server. The access routers will then add this *subGUID* to the routing table and insert $subGUID \rightarrow routerGUID$ mapping to the GNRS server. At the sender, Bus#1 sends position and occupancy information to the subGUID. If the packet arrives at the edge router, but the bus has moved to the next access point, the packet will be stored locally based on GSTAR protocol. The router then performs a GNRS lookup for the latest NA (GUID of the access router) of the subGUID, and forward the packet to this NA.

Handling Mobility: ICN-BUS involves mobility as sensors are installed on buses, which is handled differently in MobilityFirst

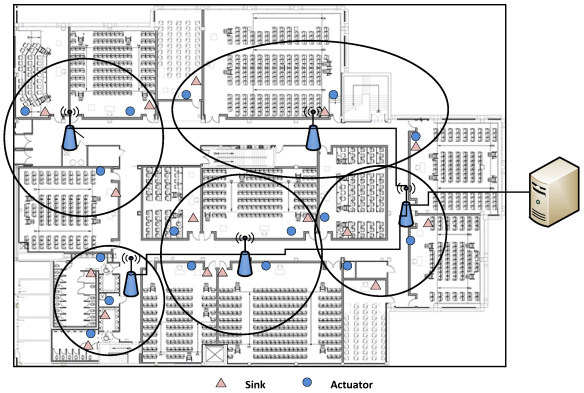

Fig. 8: BMS Topology based on flood plan

and NDN. NDN differentiates mobility of a data consumer from that of a producer. When a consumer moves to a new location after sending out an Interest, the Data may get lost, which requires the consumer to simply resend the Interest. Depending on the network topology and data availability, the new Interest might be forwarded to the same or a different data producer. If the data producer itself has moved, the solution is to flood it across the network which might incur very high control overhead [2]. On the other hand, MobilityFirst does not differentiate between producer mobility and consumer mobility.

VI. Simulation Results

We have conducted detailed evaluations of our ICN-BMS and ICN-BUS systems using NS-3 simulator. We realize the functional components of these systems on MF-IoT and NDN-IoT architectures as discussed in Section V. The architecture efficiency is compared in terms of delay, throughput, and control overhead metrics.

A. Performance of ICN-BMS

Figure 8 shows the topology of the simulated ICN-BMS system that includes three types of network entities: wireless sink nodes (a sink node represents a local wireless sensor network that includes both the sink and sensors), ICN-BMS server, and the actuators (such as a thermostat or a light fixture). Each actuator is associated with the sink in the same room.

Delay: We first look at the average data reporting delay, which is the average delay between a sink and the server, in Figure 9. In the simulations, each sink aggregates 10 sensors, with each sensor reporting data every 0.5 seconds. We observe that the average reporting delay is mostly influenced by the number of hops between the sink and the ICN-BMS server. Since MF router implements link state control messages and acknowledgement for reliable delivery due to which extra delay is introduced. Hence we observe, the average delay of MF is slightly higher than that of NDN.

Next we compare the performance when actuators are involved. Actuators are co-located in the same room as the sink, but controlled by the BMS server which issues notifications after processing the sensed data. This introduces additional round-trip delay as observed in Figure 10. The performance of MF is about 10% higher than NDN due to processing overhead at every hop.

Goodput: We next report the throughput of ICN-BMS in Figure 11. As the number of sensors per sink (while having each sensor maintain the same reporting frequency) increases, the throughput of the system increases. Between MF and NDN, the former shows better goodput due to less per-packet overhead afforded by MF protocol, even considering per-hop reliable data transport.

Routing state: Next, we compare the two architectures in terms of the routing state. In the NDN-BMS implementation, the Interest packet (either form a sink to the server, or from the server to an actuator) carries sensor data, and hence, the Pending Interest Table (PIT) on the sink node should be large enough to accommodate the maximum rate of Interest load. Also the Interest packets to actuators need to be stored until the corresponding ACK is returned from the actuator. Due to these factors, the PIT size grows with increasing sensing activity in the network. On the other hand, MF routing state only record each device in the network, which is much smaller than the number of data items produced by these devices. Figure 12 shows this affect as the aggregate rate of content generation increases in our setup.

B. Performance of ICN-BUS

We next evaluate ICN-BUS to examine the mobility support of NDN and MF. The setup simulates the MDT (i.e. bus), Bus schedule server as discussed in Section V. In the simulations, we deployed 8 access points following the uniform random distribution within a rectangle area of 400×800, and connected them to the ICN network via a binary tree topology (typical in a access setup), maintaining equality of hop counts from all access points to the server. Interests (in NDN)/data requests (in MF) are issued every second, expecting 100 bytes of data in response. In order to clearly demonstrate each ICN's mobility support mechanism, we eliminated application-layer mobility handling – i.e., data requests retransmission in MF and Interest retransmission in NDN.

Delivery success rate: We first look at consumer mobility and report the data delivery success measured at the mobile data terminal (MDT) in Figure 13. The results of the two ICN architectures are rather comparable with increasing speed of the MDT. The reason that NDN results appear better is mainly due to how the simulators are implemented: NDNSim operates at the chunk level while MFSim operates at the byte level by segmenting a chunk into bytes for better hop-by-hop reliable transmission.

Control overhead: We next evaluate the performance in the case of producer mobility. In NDN, a mobile producer has to be dealt with some form of flooding to ensure the Interest delivery; while in MF employees late-binding, hence the network only needs to issue a GNRS query at the last hop(this query may be issued multiple times until the new

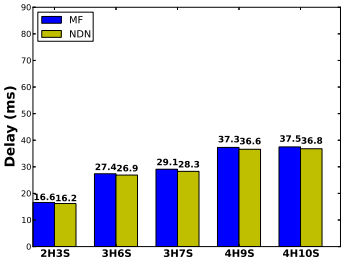

Fig. 9: Average reporting delay between the sink and the ICN-BMS server.

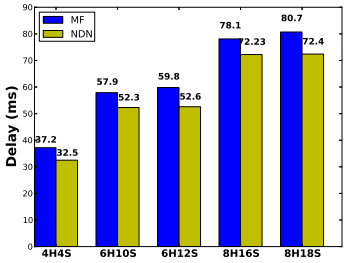

Fig. 10: Average delay between the sink and the actuator in BMS.

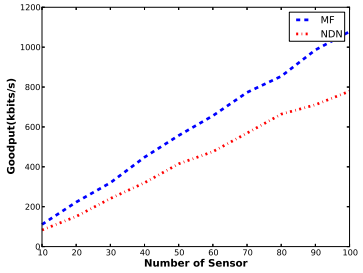

Fig. 11: Goodput at BMS Server as number of sensor increasing.

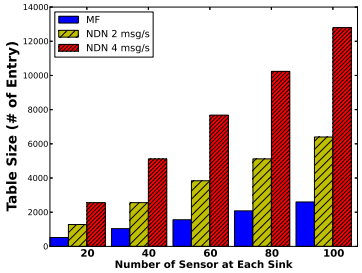

Fig. 12: Table size on the sink node in BMS.

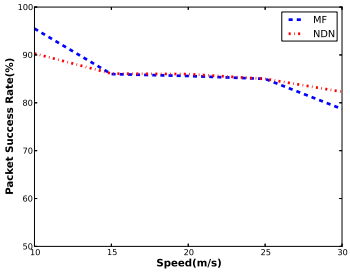

Fig. 13: Packet success rate in a pull mode.

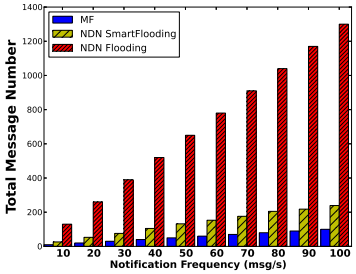

Fig. 14: Control overhead in producer mobility.

location is obtained). Though NDN ensures Interest delivery through flooding, MF incurs a much less control overhead, as shown in Figure 14. Finally, we point out that NDN's strategy layer can adopt a smart flooding strategy to avoid network scale flooding.

VII. CONCLUSION

In this paper, we argue the potential of using ICN networks to support IoT applications, and provide a detailed performance comparison of NDN and MF in their capability of supporting IoT scalability and mobility. Through detailed simulations, we find that NDN and MF achieves comparable performance results – in terms of delay, goodput and packet success rate. However, the results also show that MF shows better performance in terms of control overhead when mobility is involved and less routing state in general. Towards realizing a full-fledged ICN-based IoT platform, our next steps will involve prototype evaluation on actual testbeds, especially validating the performance of MF Pub/Sub model.

REFERENCES

[1] "Cisco visual networking index: Global mobile data traffic forecast update," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, 2009-2014.

[2] A. Azgin, R. Ravindran, and G. Wang, "Mobility study for named data networking in wireless access networks," to appear in International Conference on Communication, 2014.

[3] J. Burke, A. Horn, and A. Marianantoni, "Authenticated lighting control using named data networking," *University of California, Los Angeles, CA, NDN Technical Report NDN-0011*, 2012.

[4] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "Copss: An efficient content oriented publish/subscribe system," in *Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2011*.

[5] L. K. Haakenstad, "The open protocol standard for computerized building systems: Bacnet," in *Proceedings of the IEEE International Conference on Control Applications, 1999.*

[6] W. LAWRENZ., *CAN system engineering*. Springer, 2013.

[7] X. Liu, W. Trappe, and Y. Zhang, "Secure name resolution for identifier-to-locator mappings in the global internet," in *22nd IEEE International Conference on Computer Communications and Networks (ICCCN), 2013*.

[8] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, 2012.

[9] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), 2012*.

[10] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," 2010.