# *CS-Man: Computation Service Management for IoT In-Network Processing*

*Qian Wang, Brian Lee, Niall Murray, Yuansong Qiao*
Software Research Institute
Athlone Institute of Technology
Athlone, Co. Westmeath, Ireland
{qwang|ysqiao|nmurray}@research.ait.ie, blee@ait.ie

***Abstract*—The Internet of Things (IoT) expects to link billions of devices to the Internet, which will produce massive amounts of data. Current approaches move the IoT data out of the network for processing. It results in long delays and increases the network traffic. The Named Function Networking (NFN) proposes a generic computation architecture for in-network data processing. But it does not consider a scheduling scheme or provide details of how to deploy services. This paper designs a computation service management (CS-Man) protocol by utilizing the NFN concept to assign and schedule computation tasks within IoT network. It is implemented by two procedures: service discovery and service deployment. Thus, the whole network is capable of assigning an advanced task to the specific node as well as fetching necessary pieces required by that task. Experiments including five use cases have been done to prove the feasibility of CS-Man. It also lowers the network traffic approximately by four times when compared with the out-of-network processing method.**

***Keywords—Internet of Things; Named Function Networking; Computation Service Management; In-network Processing***

## I. INTRODUCTION

There are many standalone Internet of Things (IoT) systems appearing in different domains during the past years. The forecast shows that the number of IoT devices will reach 50 billion by 2020 [1]. Efficient data aggregation techniques are needed to reduce the rapidly increased IoT network traffic. Early IoT approaches embedded application logic into the devices' firmware [2]. It is therefore a huge task to upgrade and not practical to reprogram IoT devices to accommodate every change in applications. To solve these problems, authors in [3] propose to move the application logic from the firmware to the Cloud server for out-of-network processing, which sacrifices the direct interaction between IoT devices. Additionally, their design increases the risk of network congestion as data needs to be sent in and out of the network for every processing procedure. An ideal IoT platform should support various application logic running on top of numerous IoT devices. To this end, the Task Resource Abstraction (T-Res) [4] utilizes Constrained Application Protocol (CoAP) to send users' requests with processing logic together to the IoT devices, which enables IoT in-network processing as well as decoupling high-level applications from the network infrastructure.

The above-mentioned solutions for IoT data processing (in/out network) are built over a traditional network model that is host centric. However, in relation to IoT devices, some are mobile while most are resource constrained. Current Internet Protocol (IP) does not fit well for communication between these devices, not to mention sometimes the interaction becomes dynamic [5]. What's more, IoT applications always tend to be information centric as consumers usually need data itself without being concerned to the object that provides the required data [6].

The emerging network architecture, Information Centric Networking (ICN) [7], offers each data a unique name in order to link consumers requests directly to the content. One of the popular paradigms in ICN, Named Data Networking (NDN) [8] is an overlay design running on top of any protocol, this feasibility makes NDN applications to minimize modification for constrained devices. NDN also supports data caching along the routing path to shorten delivery time in case the same data is requested again. As an extending concept of ICN, Named Function Networking (NFN) [9] suggests naming functions too, so that consumers could get the desired result by expressing data and function names. However, there are potential problems mentioned above in the proposed NFN architecture although it is not specified for the IoT usage.

As discussed before, many IoT devices are resource-constrained in power, storage, computing and so on, which means they may be not capable of executing functional codes to process data. It is also not reasonable to ask users who are out of the IoT network to assign tasks to each IoT device according to its capability. Hopefully, consumers send a request to get what they want and not worry where to find or processed. According to NFN design, the processing procedure can be executed by the data owner or the function owner, or maybe even other adjacent nodes [9]. On the downside, it has not provided detailed schedule rules to determine the execution location, which brings issues in IoT scenarios. To name a few, the IoT device may fail to meet the execution requirement although it has both the data and function locally. Duplicated work could be done if multiple IoT devices process the data at the same time as they are all capable of.

To remedy this situation, this paper aims to implement computation service deployment for specific IoT devices to make sure the task can be successfully completed. Many requirements exist for building an IoT over ICN. Zhang et al. [6] discuss challenges in detail, involving functional naming, limited caching, and processing efficiency and so on. Our

design limits the research scope on how to manage task execution when applying the NFN computation model into IoT in-network processing. The main contributions of this paper include:

- A new naming scheme to simplify service decomposition
- The CS-Man protocol, which achieves efficient IoT in-network processing by two procedures: the service discovery and the service deployment
- A real-world testbed. Analyzed result are presented to verify the feasibility and better performance of our design

This paper is organized as following: Section II introduces the related work of this paper. We use Section III to show the system architecture and explain the protocol in detail. The real-world testbed deployment and the analysis of experiments are presented in Section IV. Section V makes the conclusion and then discusses future work.

## II. RELATED WORK

Small devices with sensing and communicating capabilities could be placed in a remote or dangerous location to capture and transfer data for human. Such wireless sensor networks (WSN) can link hundreds or even thousands of sensors. [12] and [13] survey efficient ways to reduce WSN traffic. Data aggregation is regarded as the beneficial solution to transform raw data into meaningful information. Recently advanced technology has spurred the upgrade from WSN to IoT, which not only collects sensing data but also produces processed data. The amount of IoT data is much more than that of WSN. Popular computing operations are the sum, max, min, count and mean, which are implemented in this paper as computation tasks within the IoT network.

Most devices in both WSN and IoT are resource-limited. It is difficult to keep them working as well as save power. TinyDB [14] advocates a novel query processing mechanism for sensing network in SQL-format. The smart devices are able to control when, where and how often the task should be assigned. Another query system called NDN-Q [17] is designed for data collection in the V2X (Vehicle-to-Vehicle and Vehicle-to-Infrastructure) scenarios, which uses NDN, one of the future network architecture. However, this paper concentrates on more general computation services running at heterogeneous devices to implement in-network processing for IoT applications.

The communication in NDN includes transimission of two types of addressless packets [8]: Interest and Data, which identify unique content within the network. A consumer puts the name of desired content in Interest packet and sends it out. Then the data owner returns matched content using the Data packet. NDN routers enable the above procedure by means of Forwarding Information Base (FIB, to forward Interest to next hop), Pending Interest Table (PIT, to add waiting Interest and return data in the reverse route of Interest) and Content Storage (CS, to temporally store received data).

Common Client Library (CCL) [10] is the common API offered by the NDN. It has been written in different languages. This paper chooses NDN-Javascript (ndn-js) Nodejs version because of the functional programming feature provided by Javascript language.

The distinct character of NDN is to obtain existing content within the network by their name. Named Function Networking (NFN) extends the network's functionality to be capable of re-producing data and storing processed result [9]. It proposes to give each data processing logic a name so that the network could work as a powerful machine when it owns both data and function. Named service is also adopted by NestServer [16] that is built on Content Centric Networking (CCN) [18]. It enables users to invoke multiple services including parameters together in one expression. The network looks for the service by matching its unique name and transfers the data to the location of the required service(s) for sequence execution. NestServer designs for service-chaining and assumes the service owner has the capability to complete assigned task, which is different from our goal.

The NFN architecture adopts a novel naming scheme using Lambda calculus, which embodies functional parameters into the Interest expression. The Interest name resolution guarantees to recursively get sub-expressions (could be function or data name) until it cannot be decomposed. The processing starts only if both function and data are obtained. According to the authors in [9], the locality-of-execution is discovered and decided by the network according to the processing policy or resource availability. However, they leave it blank for any related management rules.

PIoT [15] is the fundamental and simple implementation for CS-Man. It improves the NFN computation concept by adding a role as computation unit in the network. However, PIoT currently considers only one computation node, which is not reasonable in the real world. As a result, it is not powerful enough to deal with the situation that more than one node could be able to offer computation service.

To the best of the authors' knowledge, CS-Man is the first computation service management protocol to improve IoT network performance by combining the NFN model with in-network processing method.

## III. SYSTEM DESIGN

The goal of this paper is to enable approprite IoT devices to join the in-network computation process for each task according to its current status. It inherently poses great challenges to apply ICN-based approaches into the IoT domain, such as: functional naming scheme, task decomposition and computation balance and so on. This paper mainly focuses on verifying the necessity of the scheduling scheme to enable IoT devices to offer in-networking computation service in their appropriate way. The proposed computation service management (CS-Man) protocol includes two parts: the service discovery and the service deployment. It is simple in the current version, yet practical. More parameters can be added later in this architecture to optimize the scheduling result.

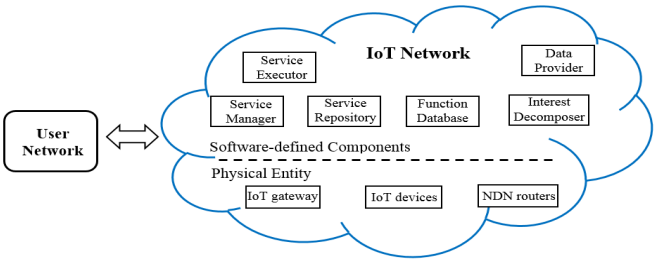


Fig. 1. System Components

### A. System Components

The performance of heterogeneous devices connected to the IoT network will not be the same. It is necessary to deploy tasks according to the device's capability. Fig. 1 shows the system components in our design, there are broadly three types of the physical entities within the IoT network: IoT gateway, NDN routers and IoT devices. All the physical machines play different roles to support the computation service functionality. The following is the basic introduction of each software-defined component drawn in Fig. 1:

- Data Provider (DP): captures the data based on consumers' request, such as: sensors.
- Function Database (FuncDB): stores all functions that are used to process data.
- Service Executor (SE): generates processed result by executing functions on data.
- Service Repository (SR): saves the information of all SEs within the IoT network.
- Service Manager (SM): updates the SR and assigns task to specific SE.
- Interest Decomposer (ID): communicates with the SM to get the name of SE; re-organizes the Interest from consumers, and then forwards the newly-constructed Interest to the specified executor.

### B. Functional Interest Expression

CS-Man is built over the NDN platform that is a consumer-driven network. All objects that provide data or functions need to publish the content's name to the network before anyone can access it. To help explain the Interest expression, we present the published name with corresponding IoT device in Table I.

The functional Interest in our design consists of the computation service name and data name. For the whole expression, each independent piece starts with the slash symbol hierarchically and is divided by the short dash to separate from the others. For example:

/ait/service-/sum-/sensor1-/sensor2 (1)

The above Interest naming scheme is used in our design. It means to add two values captured from sensor 1 and sensor 2 respectively. The Interest contains three types of information: (i) /ait/service, to require the network to perform a computation service, (ii) /sum, to express a function name and (iii) /sensor1-/sensor2, for requested data. It is worth mentioning that current implementation only supports a single computation service each time while the number of required sensing data is flexible (range from one to six, as there are six data providers in our testbed).

TABLE I. PUBLISHED NAMES IN IOT NETWORK

| Role in IoT Network | Published Name in IoT Network |
|---|---|
| Interest Decomposer | /ait/service- |
| Service Executor | /node*/compute |
| Data Provider | /sensor* |
| Function DB | /sum; /mean; /max; /min; /count |
| Service Manager | /scheduler |

### C. Computing Function

For the purpose of data processing, this paper realizes the following computing operation: sum, max, min, mean and count. They are also the published names to invoke corresponding service module. All the computation service is stored in the Function Database (presented in Table I) and can be requested by the SE. Because NDN allows the content to be sent in the format of string, array or number, we get help from json-fn [11] to parse the service name in a string (sent by FuncDB) to be executable function code (used by SE).

### D. CS-Man: Computation Service Management

The CS-Man aims to enable IoT devices working together to offer advanced computation service. It is done by use of two work flows: the Service Discovery is used to collect information about the capable nodes within the network to provide service; the Service Deployment enables assignment of the current Interest to specific nodes for processing based on the previous procedure. To be clear with how the protocol works, we illustrate the details with related figures and examples.

- *Service Discovery*

Within the IoT network, a node should publish its capability to indicate it is able to process data. This kind of node is called Service Executor. As shown in Fig. 2, Service Executor (1 to j) publishes its name as /node*/compute (* changes as 1 to j according to the node) to the network. The Service Manager is responsible for collecting all executors within the IoT network and store them in the Service Repository. Meanwhile, the Service Manager also needs to pu-

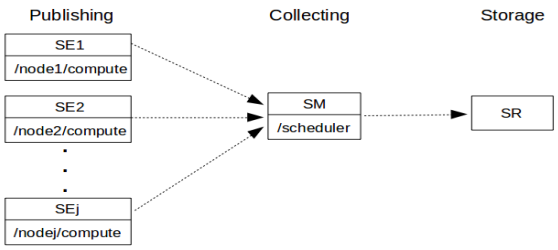


Fig. 2. Service Discovery Procedure

-blish its name (/scheduler) to the network, which will be used in the service deployment.

CS-Man merely saves the name of Service Executors in the repository as the ICN-based approach uses the names for routing.

- *Service Deployment*

The Service Deployment starts when every new Interest firstly arrives at the Interest Decomposer. It could be guaranteed because all Interest in our design begins with the prefix /ait/service. The Interest Decomposer publishes its name as /ait/service-. The ending symbol, short dash, matches all Interest's name according to the longest match rule. For convenience, we use (1) as the example of the initial Interest sent by the user and also assume the Content Storage (CS) of all nodes is empty. Fig. 3 describes the processing sequence that includes the following steps:

Step 1: the user sends the initial Interest to the IoT network.

Step 2: Interest Decomposer receives the Interest, and then adds the whole initial Interest name to its Pending Interest Table (PIT). It also sends an Interest named /scheduler to Service Manager.

Step 3: Service Manager requests the capable node to provide service from the Service Repository.

Step 4: Service Repository returns the name (/node*/compute) of one Service Executor to Service Manager.

Step 5: Service Manager forwards the name (/node*/compute) of Service Executor to Interest Decomposer.

Step 6: Interest Decomposer builds a new Interest by adding the name of specific Service Executor as (2), and then forwards the re-organized Interest.

/node*/compute-/sum-/sensor1-/sensor2 (2)

Step 7: Service Executor decomposes the received Interest to get function name and data name(s). Then, it sends out all sub-interests.

Step 8: Function Database gives back the required function same path that the Interest came from. The corresponding Data Provider(s) returns data as well.

Step 9: Service Executor does the computation service when all content is obtained and returns processed result to Interest Decomposer.

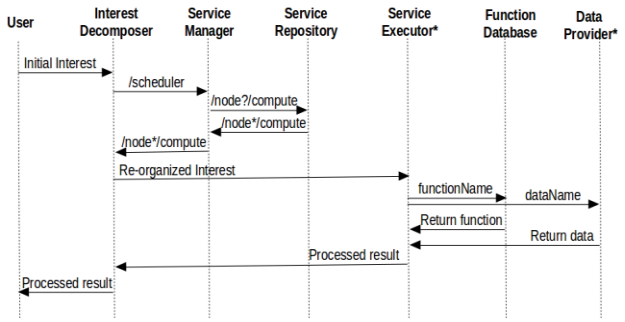


Fig. 3. Service Deployment Procedure

Step 10: Interest Decomposer delivers the processed result to the user.

After the first round of the service deployment, the used function will be stored along the routing path. In later processing, the request and return for the same function will be omitted between Service Executor and Function Database, which minimizes network traffic and saves resources. It is the same between the Service Executor and Data Provider, except the sensing of data is real-time and requires updating for each Interest.

## IV. Experimental Setup and Evaluation

This section shows the real-world testbed and analyzes the result of our design. Five user-cases are implemented to prove the feasibility of the CS-Man protocol and comparative study with out-of-network processing approach has done to evaluate the processing time and network traffic.

### A. Experimental Setup

The CS-Man is running at a small-scale and the network topology is displayed in Fig.4. We place the Interest Decomposer on the IoT gateway and combine the Service Manager with the Service Repository. There are three nodes acting as Service Executor (SE1~SE3), six as Data Providers (s1~s6) and also two NDN nodes for routing and storing. The Function Database is put on a separate node. The lines between nodes denote that they are neighbors in the routing table.

All nodes in the testbed are simulated by the Docker containers running on an HP server (DL380-G7 with 96GB memory, 24 Intel Xeon CPU cores and 1 TB hard drive) with the Ubuntu 14.04 operating system. They are installed with ndn-js and Named Data Forwarding Daemon (NFD). In addition, Named Data Routing Protocol (NLSR) is running on each node to generate and update the forwarding table.

There are five test cases used in the experiment to get specific parameters for comparison and the following are the initial Interest expressions:

/ait/service-/count-/s1-/s2-/s3-/s4-/s5-/s6 (3)

/ait/service-/sum-/s2-/s3-/s4-/s5 (4)

/ait/service-/max-/s3-/s5 (5)

/ait/service-/min-/s2-/s6 (6)

/ait/service-/mean-/s3-/s4-/s5 (7)

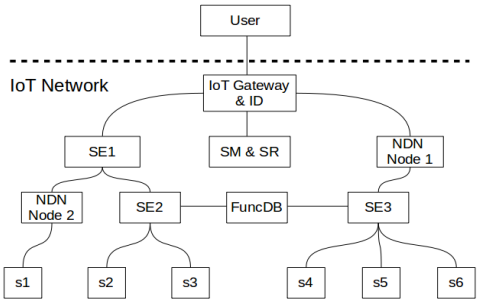


Fig. 4. Testbed Topology

The user in Fig. 4 is the execution location for the out-of-network processing. For this case processing is outsourced and data must be sent out of the network for each task. For example, Interest (7) will be divided into several sub-interests: /mean, /s3, /s4, and /s5. The exectution node waits for all pieces return so that to have resources ready for the computation.

### B. *Evaluation Results*

The evaluation considers measuring network traffic flow and calculating processing time. The designed use cases are executed by using CS-Man and out-of-network approach.

As shown in Fig.5, the traffic is counted at the user node for each function. CS-Man composes both function and data into one Interest packet, which significantly reduces the packet length. However, in the out-of-network scenario, the number of Interest increases resulting in large traffic. The average value for packet length is 409.4 for CS-Man while it is 1568 when the user is computing.

It takes more time when using CS-Man for in-network processing than the out-of-network processing method. The result of the comparison is presented in Fig.6. CS-Man spends around 40-50 milliseconds but processing outside the IoT network uses no more than 25 milliseconds.

The result is obviously that CS-Man reduces network traffic at the cost of processing time. Delivering resource out of the network increases data flow while saving time. According to the current experiment, we still argue that CS-Man performs b-

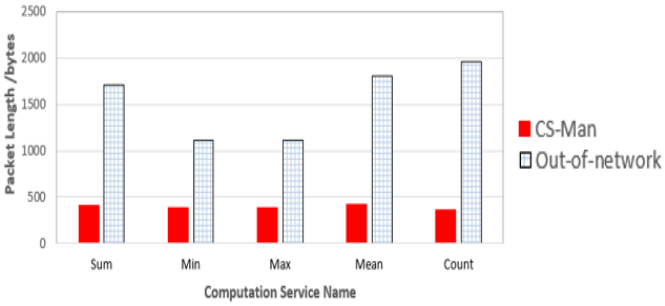


Fig. 5. Packet Length Comparison Result

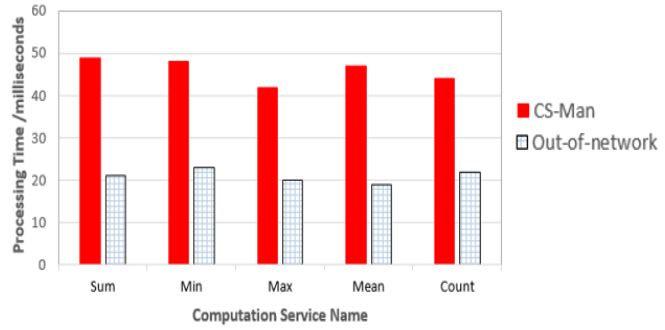


Fig. 6. Processing Time Comparison Result

-etter than the out-of-network processing approach. Because, the amount of time taken for CS-Man is twice that of out-of-network in milliseconds, but the latter increases network traffic approximately by four times. It is known that massive data is transferred within IoT networks, which is very likely to cause congestion or even crash if the amount of data multiples. What's more, data transmission requries energy consumption on battery life, which definitely increases operational cost. As a result, CS-Man is very promising to achieve efficient and economical in-network processing for IoT.

## V. Conslusions and Future Work

This paper proposes CS-Man, a scheduling protocol to enable IoT devices cooperating with each other to provide in-network compuation. As resource-constrained devices are common in IoT scenarios, some of them may not able to process data for the received task. It requires a manager role to know the capability of each node and assign a task to a suitable one. CS-Man solves the problem by the service discovery scheme which collects and stores the execution-capable nodes within the network. It then invokes the service deployment procedure to assign a task to the specific node when a new Interest comes to the IoT gateway. The available computation service (sum, max, min, mean, count) is an application layer design, which facilitates reconfiguration and upgrades.

The real-world testbed is set up on multiple Docker containers for running advanced logic as well as producing sensing data. The evaluation tasks are designed to execute five user-cases, whose processing time and data flow are recorded for comparison. The result has proven the feasibility of our design and shown reduced network traffic in almost fourfold.

It is at the beginning stage to apply ICN-based approaches to deliver IoT traffic. An essential future work for CS-Man is to optimize the service schedule, such as: according to the real-time status of IoT devices, compose multiple function names in an Interest expression for complex tasks, implement computing-balance by distributed sub-service execution and lowering processing cost and so on. CS-Man is an ongoing project to develop a powerful schedule protocol for optimizing ICN-based computation service within IoT network.

## Acknowledgment

This publication has emanated from research supported by research grants from Institutes of Technology Ireland (IOTI) under Postgraduate Scholarship Initiative 2014, Science Foundation Ireland (SFI) under Grant Number 13/SIRG/2178, and Enterprise Ireland (EI) under the COMAND Technology Gateway programme.

## References

[1] Cisco, Available: "Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019," Accessed: Dec-2015.

[2] M. Kovatsch, M. Lanter, and S. Duquennoy, "Actinium: A restful runtime container for scriptable internet of things applications," 2012 3rd International Conference on the Internet of Things (IOT), Wuxi, China, pp. 135–142, 2012.

[3] M. Kovatsch, S. Mayer, and B. Ostermaier, "Moving application logic from the firmware to the cloud: Towards the thin server architecture for the internet of things," 2012 Sixth International Conference on

Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Palermo, Italian, pp. 751–756, 2012.

[4] D. Alessandrelli, M. Petracca, and P. Pagano, "T-res: Enabling reconfigurable in-network processing in IoT-based WSNs," 2*013 IEEE* International Conference on Distributed Computing in Sensor Systems (DCOSS), Cambridge, MA, pp. 337–344, 2013.

[5] J. Li, Y. Shvartzshnaider, J. Francisco, R. P. Martin, and D. Raychaudhuri, "Enabling Internet-of-Things services in the MobilityFirst future internet architecture", 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), San Francisco, CA, pp. 1-6, 2012.

[6] Y. Zhang, et al, "Requirements and challenges for IoT over ICN", IRTF ICN Research Group, Nov. 2015.

[7] G. Xylomenos, C. N. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos, and others, "A survey of information-centric networking research," Commun. Surv. Tutor. *IEEE*, vol. 16, no. 2, pp. 1024–1049, 2014.

[8] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," SIGCOMM Comput Commun Rev, vol. 44, no. 3, pp. 66–73, 2014.

[9] M. Sifalakis, B. Kohler, C. Christopher, and C. Tschudin, "An information centric network for computing the distribution of computations," in Proceedings of the 1st international conference on Information-centric networking, pp. 137–146, 2014.

[10] NDN, "NDN Common Client Libraries API 0.4.0 documentation," Available: http://named-data.net/doc/ndn-ccl-api/, Accessed: Feb-2016.

[11] V. Kiryukhin, Available: http://www.eslinstructor.net/jsonfn/, Accessed: Feb-2016.

[12] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," Computer Networks 52 ELSEVIER, pp.2292-2330, 2008.

[13] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," IEEE Wireless Communication, vol.14, pp.70-87, 2007.

[14] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," ACM Transactions on Database Systems (ToDS), pp.1-47, 2014.

[15] Y. Ye, Y. Qiao, B. Lee, and N. Murray, "PIoT: Programmable IoT using information centric networking," IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 2016.

[16] D. Mansour, T. Braun, and C. Anastasiades, "NextServe Framework: supporting services over content-centric networking," 12th International Conference, Wired/Wireless Internet Communications, Paris, France, pp.189-199, 2014.

[17] W. Drira, and F. Filali, "NDN-Q: An NDN Query Mechanism for Efficient V2X Data Collection," IEEE SECONW 2014.

[18] F. Oehlmann, and H. Nedermayer, "Content Centric Networking," Seminar Future Internet & IITM WS2012/2013, pp.43-49, 2013.