PIoT: Programmable IoT using Information Centric Networking

Yuhang Ye[†], Yuansong Qiao[†], Brian Lee[‡], Niall Murray[†]
Software Research Institute
Athlone Institute of Technology
Athlone, Co. Westmeath, Ireland
[†]{yye|ysqiao|nmurray}@research.ait.ie, [‡]blee@ait.ie

Abstract—**The Internet of Things (IoT) places significant demands on network infrastructure in order to process data captured by ubiquitous sensor devices. One existing technique to support this sensor data processing involves transporting captured data to cloud servers. This approach suffers from numerous issues such as increased transmission costs i.e. bandwidth consumption and delays. To help resolve these issues, this paper proposes Programmable IoT (PIoT), a novel IoT data processing architecture. It is an application layer design which operates over Named Data Networking (NDN) to enable the execution of reconfigurable processing-logic in the network. In addition, a novel naming scheme and computation service for IoT is presented to describe the processing requirements using Lambda Expressions. To verify the feasibility of our design, a real-world implementation was created and evaluated. It compares efficiency of the in-network versus out-network approaches.**

Keywords—Internet of Things; Information Centric Networking; In-network processing

I. Introduction

The Internet of Things (IoT) is a rapidly growing technology which connects many non-traditional devices to the Internet. Cisco predicts that 50 billion IoT devices will be connected to the Internet by 2020 [1]. It is a requirement that these battery powered or passive wireless devices achieve efficient Machine-to-Machine (M2M) communication. Data processing is one essential technique to enable the intelligent and seamless interactions between devices. This is achieved by converting the captured raw data into a format that machines can interpret. Early approaches burned the processing-logic into the devices' firmware and enabled the devices to communicate directly and execute processing-logic in-network i.e. within the sensor network. However, these approaches are limited to inflexible logic re-configuration and upgrades [2]. To address this issue, Kovatsch et al. [3] suggested moving the processing procedure external to the network to cloud servers i.e. outside the sensor network. This solution enabled flexible processing reconfiguration but sacrificed the direct communications between devices. The sensed data transmission to and from the IoT network places significant demands on the network resources and can also cause congestions [4]. To address these issues, Alessandrelli et al. [2] proposed Task Resource Abstraction (T-Res). It moved the processing procedure to a node within the sensor network (in-network) by sending processing requests and processing-logic using the Constrained Application Protocol (CoAP) [5].

978-1-5090-0223-8/16/$31.00 ©2016 European Union

Alessandrelli et al. [2] stated that universal interfaces and network standards are essential components to achieve efficient IoT communication. Existing approaches that aim to support communication, including in/out-network processing [2] [3], are built over the conventional host centric networking protocol, i.e. the Internet Protocol (IP). IP has the benefit of universal deployment [6]. However, the IP stack is too heavy for constrained devices. Other issues also exist in terms of scalability and a lack of the mobility support [7].

To address some of the above mentioned limitations, Information Centric Networking (ICN) [8] is proposed as an infrastructure to support IoT requirements [6]. ICN is a novel information based architecture to describe data acquisition by linking consumers' requests to content directly. Named Data Network [6] (NDN) is one implementation of ICN that removes the relationship between the data consumer and the data producer. Specifically, NDN retrieves "*Data*" packets by sending "*Interest*" packets with unique names to the network. NDN is an overlay design. It is agnostic in terms of the protocol it operates over. This design allows NDN applications to run on IoT specified protocols and minimize the protocol stack overheads for constrained devices.

Some existing works have reported the benefits of employing ICN for the IoT network. To support the large-size data delivery, [9] proposed a novel compression and fragmentation mechanism that supported NDN applications running over the link layer. Their results indicated efficiencies in terms of energy consumption and memory footprint compared to the 6LoWPAN approach. Considering the multi-source traffic pattern of IoT, the work reported in [10] presented a modified NDN architecture to support a single-Interest and multiple sources (sensor). The experiment results indicated increased data diversity and reduced collection times compared to conventional approaches. In general, the existing works on ICN in IoT scenarios are mainly focusing on the traffic optimization problems. In this context, no works have considered the in-network data processing in the IoT, e.g. data aggregation to minimize the traffic size of IoT network.

Related to ICN, but not specified for the IoT, the Named Function Networking (NFN) presented in [11], proposed a novel ICN framework to extend the data processing functionality. NFN allows consumers to request the network to perform computation and return the corresponding results. The novel naming scheme in NFN satisfies the processing requirements but has limitations for the IoT scenarios i.e. the computation node may not have the data nor the function to process the data.

This paper proposes a novel network architecture called Programmable IoT (PIoT). To the best of the authors' knowledge, this is the first approach to enable in-network processing for IoT requirements using ICN.

The main contributions of this paper include:

- A naming scheme for IoT to overcome the limitation in NFN (Section III Part A)
- An Interest processing service to use the IoT network resources efficiently (Section III Part B)
- A real-world PIoT testbed to verify the feasibility of the design and a comparative study to evaluate the performance of in/out-network processing (Section IV)

This paper is organized as following: Section II introduces the background knowledge. The PIoT architecture design is illustrated in Section III. The real-world testbed deployment and experimental evaluations are presented in Section IV. Section V provides the conclusions and future work.

II. Background

A. Named Data Network

NDN defines 2 types of packets [8]: Interest and Data. These packets are named by a Uniform Resource Identifier (URI). A client or consumer of content sends an Interest packet in order to search for data. Once the Interest reaches a data producer, the producer encodes the content in the Data packet, and returns it back to the consumer. In this way, the content is not tied to any one host. This addressless naming mechanism enables the direct communication between consumer and any data owner.

B. Named Function Network

In this paper, to distinguish the Interests in NDN and NFN, we specify that interests in NDN are called Data Interests and interests in NFN are called Functional Interests. The Functional Interest is a superset of the Data Interest. It achieves this by constructing the Interest Name in Lambda Calculus semantics (Lambda Expression). In this way, the Functional Interests are able to request the network to perform processing tasks and return results using FOX (Find Or eXecute) [11] rules as below:

- Find: The network returns the expression result immediately, if the result is cached in the network.
- Execute: The network will collect necessary data (raw data/processing-logic) and evaluate the result if it has not been cached.

Lambda Calculus is a Turing Machine equivalent model. The Lambda expressions can be expressed as follows:

$$expr ::= v \mid expr\text{-}l\ expr\text{-}r \mid (\lambda x.expr)arg \qquad (1)$$

The *"v"* represents the basic Lambda expression to find a variable value (i.e. Data Interest in NFN). The expression "*expr-l expr-r*" applies the expression left (*expr-l*) to the expression right (*expr-r*). For example, the expression *inc 1* indicates increase *1* by *1*, the result is *2* in Haskell. The expression"*(λx.expr)arg*" illustrates that all the values of *x* occurring in *expr* will be replaced by the *arg* using *β-reduction*. For example, *(λk.add k 1)1* indicates the *k* in *add k 1* is *1*, the result is *2* in Haskell).

NFN uses Lambda expression due to its simplicity and parallelizability. It is also able to build powerful and complex higher order function by combining basic functions.

E.g. $f(u) = 3u^2 + 2u + 1$

$$+\ (+\ (\times\ 3\ (\times\ u\ u))\ (\times\ 2\ u))\ 1 \qquad (2)$$

The $\times\ u\ u$ indicates the $u \times u$, and the × (multiplication) is the predefined function while the + (plus) infers the addition of the two values. Similarly, NFN predefines some useful processing-logic (e.g. /words_count) for specific applications. NFN works out the result once the data of a expression is available. This scheme helps with data reduction in many situations (e.g. compression of video content).

NFN is a solid framework for processing solutions. However it does not consider the constraints of the IoT scenarios outlined above. The next section will provide the PIoT architecture which utilizes the NFN concept to support IoT processing requirements.

III. Programmable IoT (PIoT) Architecture

PIoT supports flexible processing deployment and the re-configurable processing-logic upgrades in IoT networks. The novelty of PIoT includes: 1) a naming scheme for IoT scenario; 2) the Functional Interest processing service design.

A. Naming Scheme in the PIoT

PIoT uses reverse-type writing similar to NFN [11], which places the last term of the expression at the beginning of the expression to represent the data/processing-logic owner. In contrast to NFN, the first term in the PIoT expression is the computation service prefix. For greater readability, all the expressions in this paper are written in non-reversed type. NFN assumes the data/processing-logic owner is responsible for computation which is not suitable in IoT scenarios. The sensor nodes (data owner in IoT) may not have sufficient resources to perform processing. Moreover, to upgrade the logic flexibly, the function repository (function owner) can be deployed independently from the computation node. Therefore, the computation service prefix defines the computation location as anywhere suitable that has published the service. To maintain the consistency of the Name, this service prefix is also defined using a Lambda term in the expression.

According to the inverse operation of the 3rd form of the Lambda expression [11] (also known as λ-binding), below (eq. 3) can be rewritten to introduce a virtual term without changing the result,

$$\lambda k.k \leftrightarrow (\lambda z.\lambda k.k)g \quad \text{if } \lambda z \text{ does not exist in } \lambda k.k \tag{3}$$

E.g. $f(u, v) = 3u^2 + 2u + 1,\ v = /comp$

$$(\lambda v. + (+ (\times 3 (\times u\, u)) (\times 2\, u)) 1)/comp \leftrightarrow + (+ (\times 3 (\times u\, u)) (\times 2\, u))\ 1 \tag{4}$$

The λv and the corresponding prefix */comp* is the virtual term and it defines the computation service. Any network node that has published its computation service, will capture the appropriate Interest and return the result.

B. *Interest Processing Service in PIoT*

The PIoT service is an application layer design which is installed on the computation nodes. The computation nodes retain the NDN networking layer services such as: the Forwarding Information Base (FIB, to forward the Interest), Pending Interest Table (PIT, to return the data to the consumer), and Content Storage (CS, to temporal storing Data packets for enabling the delay tolerant multi-casting). It also introduces 3 additional computation services: Expression Resolver (ER), Expression Pusher (EP) and Expression Evaluator (EE) as per Fig. 1.

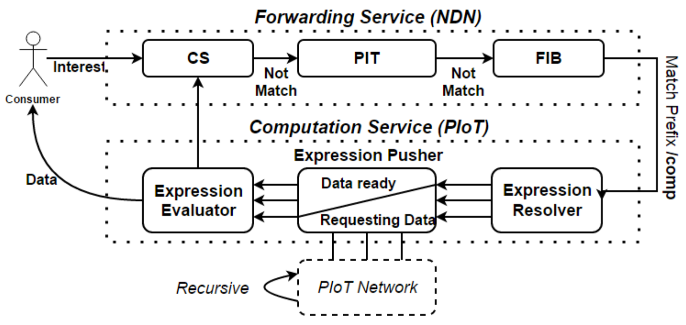

Fig. 1. PIoT service design [adapted from [6]]

The ER is responsible for analysing the Lambda expressions (the Functional Interests) and resolving them into new sub expressions. The EP creates new Interests to fetch the data of sub expressions and forwards them to the network. Once all necessary data has been retrieved from the network, EE evaluates the Lambda Expression and generates the result. After the computation is completed, the computation node returns this result.

The PIoT network works in a recursive way to support the higher order functions. It continuously breaks up the higher order functions to lower orders and forwards them to the network until the function is fully decomposed. In a converse way, the EE aggregates the returned resources (raw data and expression-(sub)results) until the result of whole expression is completely calculated. For example, consider a compute max value of average temperatures from 2 rooms (r1, r2). Each one has two temperature sensors: s1, s2. The associated Lambda Expression is reflected by eq. 5:

$$/\lambda z./max\ (/avg\ /r1/s2\ /r1/s2)\ (/avg\ /r2/s1\ /r2/s2)\ /comp \tag{5}$$

To evaluate the Functional Interest, it will arrive at a computation node which has published */comp*. The ER_1 eliminates the Lambda Term λz and then splits the expression into 3 terms: */max*, */avg /r1/s1 /r1/s2*, and */avg /r2/s1 /r2/s2* according a defined rules (i.e. In this paper, the expression is split according to the bracket notation). The 1st term (*/max*) is the processing-logic which can be retrieved from the network directly. The 2nd and 3rd terms are resolvable expressions which requires further decomposition. EP_1 creates a Data Interest for the 1st term and creates 2 Functional Interests by appending the computation service prefix to 2nd and 3rd terms. EP_1 pushes these Interests to any computation node. After ER_2 splits the Functional Interest *(λz./avg /r1/s1 /r1/s2)/comp* into 3 terms: */avg*, */r1/s1* and */r1/s2*, EP_2 pushes them to the data owner. Once the processing-logic *(/avg)* and the sensor data (*/r1/s1*, */r1/s2*) is returned to EE_2, EE_2 can then evaluate the expression and return the result back. The 3rd term *(/avg /r2/s1 /r2/s2*) is computed in the same way. Finally, the EE_1 solves the expression (eq.5) by aggregating the processing-logic (1st term) and the results of the 2nd and 3rd expressions. It then returns the final result back to the consumer.

IV. Real-World Deployment and Experiment Evaluation

A. *Real-World Deployment*

The real-world deployment in this paper contains one computation node for processing Functional Interest and NDN nodes for routing, processing-logic storage and sensor data capture. The PIoT computation services are implemented via the Common Client Library (NDN-CCL) [12]. Meanwhile, all the nodes are installed with the Named Data Forwarding Daemon (NFD) [12]. The Named Data Routing Protocol (NLSR) [13] generates the forwarding tables. The platform is built over 5 PCs and 3 Raspberry PIs. The PCs simulate the consumer, logic repositories and the routing infrastructure. The Raspberry PIs act as the sensors (with digital thermometers that return the temperature).

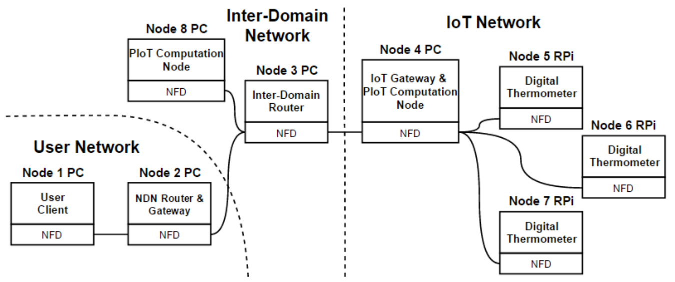

Fig. 2. PIoT network topology

The topology shown in Fig. 2 consists of two sub-networks (an User Network and an IoT Network) that communicate though the inter-domain router (Node-3). The User network contains a client (Node-1) and a gateway (Node-2). Three sensors (Node-5~7) are connected to the gateway (Node-4) in the IoT network. The processing logic is placed at Node-2 to simulate the repository. Furthermore, to compare the performance between in-network and out-network processing. Node-4 is set as an in-network computation node. In contrast, Node-8 connected to Node-3 simulates an out-network computation node. The experiment uses UDP over IPv4 as the default tunnel to transmit packets which simplifies the development. However, NDN is an overlay design which is able to communicate over arbitrary tunnels e.g. TCP/UDP, Ethernet and etc.

B. *Experiment Evaluation*

The task to evaluate this proposed system is to compute the average temperature of one office room by collecting data from 3 thermometers (registered as /s1~/s3). Node-1 sends an Functional Interest in the form as eq.6:

$$/\lambda x.(/mean\ /s1\ /s2\ /s3)/comp \tag{6}$$

In the in-network scenario, Node-8 is disabled and the Node-4 is registered with the computation service prefix. Once Node-4 receives the Functional Interest, its ER resolves the Lambda Expression into 4 sub terms, i.e. /mean, /s1, /s2, and /s3. The EP then creates new Interests for each term and pushes them to network. The sensor nodes (Node-5~7) and the processing-logic owner (Node-2) return the data back once they receive the corresponding Interest. Once all the data has been retrieved for the entire expression, the EE calculates the result and returns it back to the Node-1.

Comparatively, to deploy the out-network computation, the change is to disable the computation service of Node-4 and enable the service at Node-8. In the in-network scenario, the sensor data only travels from Node-5~7 to Node-4 (1 hop). In contrast, the out-network approach needs to transport all the sensor data (*/s1, /s2 and /s3*) to Node-8 for processing (3 hops). We assume the out-network approach results in higher network resource costs. To verify our assumption, the performance of the different designs were evaluated in two aspects: 1) round-trip time as the time interval between consumer sending the Interest and receiving the Data; 2) total bandwidth consumption of gateway in the IoT network. Experiments were executed for both the in-network and the out-network designs. The consumer (Node-1) sends out a Functional Interest (eq.6) every 0.1 seconds (10Hz) for 300 seconds.

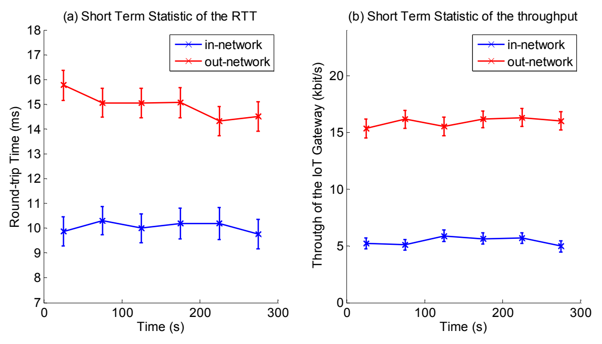

Fig. 3. Performance of the in/out-network processing

Fig. 3 compares the performance the two approaches by calculating the mean and deviation of the RTT and bandwidth consumption every 50 seconds. As per Fig. 3(a), the round-trip time of the out-network design is approximately 0.5 times higher than the in-network design. From Fig. 3(b), we see the bandwidth consumption for out-network is almost double that of in-network processing. This preliminary comparative study validates the feasibility of PIoT and shows that the combination of Functional Interest and the proposed PIoT processing service enables ubiquitous deployment of the IoT computation node. It also demonstrates that deploying the computation node inside the IoT network can significantly reduce network resource consumption.

V. Conclusion and Future Work

This paper presents a novel computation architecture to enable efficient IoT data processing. PIoT provides a flexible, re-configurable data processing architecture using Functional Interests and a novel computation service. The target of our work is to extend the NFN computation model to satisfy the computation requirements in IoT scenarios and simplify the IoT application development. In the PIoT network, sensor data and processing-logic resources are stored in the network. Users send Functional Interests to the computation node. The computation node searches necessary resources (sensor data and processing-logic) in the network automatically and calculates the results of the Functional Interests recursively. In our case, the in-network processing provides many benefits, i.e. reducing the bandwidth consumption, lowering down the round-trip time, and consequently save the transmission energy. PIoT also guarantees the dynamic upgrades of the processing-logic without accessing the devices' firmware.

PIoT is an ongoing work that reports the initial IoT experiments to verify the feasibility of the proposed in-network processing mechanism. In the future work, we plan to perform more thorough tests to compare PIoT with other conventional approaches (E.g. 6LoWPAN) from different perspectives, e.g. the end-to-end latency, network bandwidth cost, and energy consumption. Another essential research issue for PIoT is to develop a routing protocol to efficiently allocate the computation tasks to distributive computation nodes.

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 Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper," *Cisco*. [Online]. Available: http://cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html. [Accessed: 21-Dec-2015].

[2] D. Alessandrelli, M. Petracca, and P. Pagano, “T-res: Enabling reconfigurable in-network processing in iot-based wsns,” in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, 2013, pp. 337–344.

[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,” in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, 2012, pp. 751–756.

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

[5] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” 2014.

[6] 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, Jul. 2014.

[7] Y. Zhang, D. Raychadhuri, R. Ravindran, and G. Wang, “ICN based Architecture for IoT,” IETF Internet Draft draft-zhang-iot-icn-architecture-00. IRTF, 2013.

[8] 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.

[9] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, “Information centric networking in the IoT: experiments with NDN in the wild,” *ArXiv Prepr. ArXiv14066608*, 2014.

[10] M. Amadeo, C. Campolo, and A. Molinaro, “Multi-source data retrieval in IoT via named data networking,” in *Proceedings of the 1st international conference on Information-centric networking*, 2014, pp. 67–76.

[11] 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*, 2014, pp. 137–146.

[12] “Libraries / NDN Platform - Named Data Networking (NDN),” *Named Data Networking (NDN)*. [Online]. Available: http://named-data.net/codebase/platform/. [Accessed: 14-Jan-2016].

[13] A. K. M. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, “NLSR: Named-data Link State Routing Protocol,” in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, New York, NY, USA, 2013, pp. 15–20.