Energy Efficient Context based Forwarding Strategy in Named Data Networking of Things

Lenord Melvix J.S.M.,* Vikas Lokesh*

*Dept. of Computer Science and Engineering
Jacobs School of Engineering
University of California, San Diego
La Jolla, CA 92093-0114, USA

{lmelvix, vlokesh}@eng.ucsd.edu

George C. Polyzos*,#

#Mobile Multimedia Lab, Dept. of Informatics
School of Information Sciences and Technology
Athens University of Economics and Business
Evelpidon 47A, 113 62 Athens, Greece

polyzos@acm.org

ABSTRACT

Named Data Networking (NDN) of Things has gathered interest among the Internet-of-Things (IoT) research community due to its hassle-free deployment and configurability. This approach to interconnecting devices provides the network with additional information about the data transported, which can be used for improved efficiency. This is one of the targets for mass IoT or Machine-to-Machine (M2M) communications modes of 5G. In this paper, we propose a context based forwarding strategy that makes forwarding decisions based on the tolerance to inaccuracies in data defined for the application requesting it. We also propose a modification to the hierarchical naming scheme by appending the name of the application generating the interest. In addition to faster context based forwarding, this modification simplifies the naming as it is easier to understand the context under which the interest was generated even with generic names. We demonstrate an improvement in energy efficiency of the system using the proposed technique compared to an IP based implementation. In addition, the proposed technique offers a platform to implement application level access control and Quality-of-Service support.

CCS Concepts

•**Networks** → **Naming and addressing; Programmable networks; In-network processing;** Programming interfaces; Middle boxes / network appliances; Location based services; •**Computer systems organization** → **Sensors and actuators; Embedded systems;** Distributed architectures; •**Hardware** → **Power and energy;** •**Information systems** → Process control systems;

Keywords

Internet of Things; IoT; Sensor and actuator networks; 5G; Information-Centric Networking; ICN; in-network processing; middleware; network query processing; power efficiency

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICN′16, September 26-28, 2016, Kyoto, Japan

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

1. INTRODUCTION

The Internet of Things (IoT) has opened up a plethora of opportunities for objects to co-ordinate amongst themselves in order to perform complex tasks such as home automation, health monitoring, autonomous smart farming and many more. These constitute the "verticals" of the Fifth Generation wireless and mobile networking (5G), on which much of the hope for profitable business models currently rests. In particular, IoT applications are exemplifying the massive wireless sensor and actuator network requirements of 5G [12] Most embedded systems in these applications rely on batteries for power and are energy and power constrained.

Intel [9] expects that for 5G, intelligence, communication capabilities, and processing power will need to be diffused across networks and mobile devices: 5G will be the first network designed to be scalable, versatile, and energy smart for the hyper-connected Internet of everything world. Considering the limited energy availability and the high replacement or recharging cost in large scale applications, power optimization strategies become critical in order to extend the lifetime at both individual device and the network level.

Power optimization at the embedded level is supported via multiple low power operating modes available in the devices. For example, the TI MSP430FRx series offers 7 low power programmable operating modes, with each mode placing a specific set of components and peripherals in sleep mode, depending on application requirements [15]. Based on the nature of the application, embedded system developers program the system to switch between the operating modes, taking in to account the speed at which the system needs to operate in active mode and the additional power consumed due to transitions from high to low power mode and vice-versa. Thus, the nature of the application determines the right power mode for these devices.

Many IoT based applications such as the *SmartFarm* [8] involve a large number of users accessing system information for a wide range of functions. Designing an effective power mode switching strategy for such a complex system is not trivial. However, this challenge can be addressed with Named Data Networking (NDN) [6] of Things [13], since this architectural approach can provide the network with additional information about the application and the specific requested data and their characteristics and tolerances. Effective forwarding strategies can be realized dynamically based on this information, which allow the embedded sys-

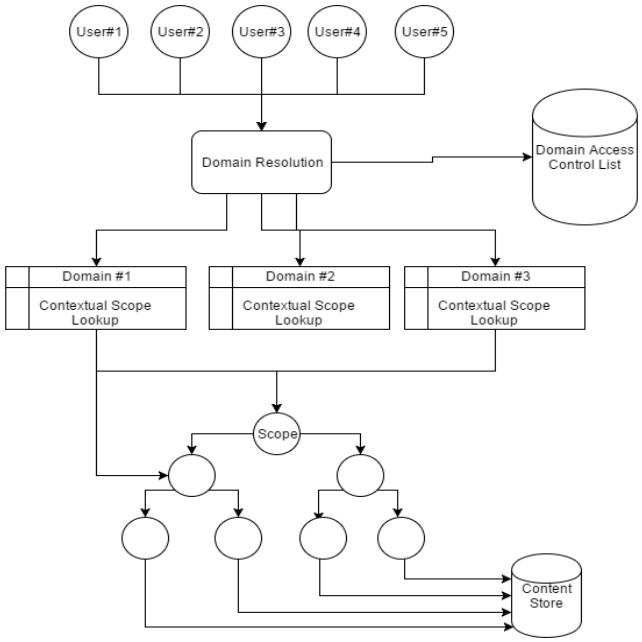

Figure 1: Block-level view of Contextual Scope Architecture

tem to set its operating power mode more efficiently. It is interesting to note that in spite of the huge volume of data producers and consumers in these applications, the type of data requested and the context under which the data would be used is fairly limited and often explicitly declared. This offers the network an opportunity to configure intelligent forwarding startegies and apply appropriate access control policies at the right contextual level in order optimize over all goals, including conserving energy and improving performance, while conforming to application specifications.

In this paper, we propose a novel forwarding strategy that makes use of the NDN data-centric approach to determine the context of the requested data and thereby its corresponding application's tolerance to inaccuracies in data. We use this additional information to make intelligent forwarding decisions by reusing cached data, or data from other similar queries (interests), thereby reducing the power consumed by the system in various ways, e.g., fewer transmissions and longer sleep time for energy constraint devices.

The remainder of this paper is structured as follows: Section 2 describes the design of a contextual scope architecture and the context based forwarding strategy. Section 3 demonstrates improvement in energy efficiency with the proposed technique through a specific relevant case study and also discusses advantages, disadvantages and potential directions for future work. Section 4 discusses related work and positions this research with respect to other research in the area. Section 5 provides a summary and our conclusions.

2. CONTEXTUAL SCOPE ARCHITECTURE

2.1 Overview

To facilitate application-context based energy efficient forwarding, we modified the name resolution technique into a three-stage process: name resolution, context based classification, and forwarding table matching. Figure 1 shows a block level view of the proposed modification. At the first stage, a domain resolution engine classifies the interest packet based on the nature of the application requesting data to perform application level access control. In the second stage, after authentication, the interest is mapped to a corresponding "scope" node in the context resolution lookup, depending on the nature of the application and the range of acceptable producers who can service the consumer's interest. In the third stage, the "scope" node finally maps to one or more faces corresponding to producers, based on application requirements and tolerance limits.

Table 1: Interest Name with Application Name appended

Consumer	Application	Interest Name
Thermostat	Temp. Ctrl	/temp/feedback
Decision	Decision Ctrl	/Tomato/temp/decision
Data Logger	Logging	/Tomato/Block1/temp/logging
Alarm	Fire Monitor	/Block1/temp/emergency
Phone	Supervision	/Tomato/Block1/temp/monitor

2.2 Naming

In this approach, the names of *interest* packets follow the hierarchical naming scheme described in [2] and [13] with the name of the application requesting data appended to the end of the interest's name. For example, consider the case of a temperature control module in a smart farm setup shown in Figure 2, with multiple consumers, such as a thermostat, a decision controller, an emergency alarm and mobile farm monitoring applications interested in temperature data. Names of NDN interest packets sent out by these consumers are listed in Table 1. Notice that unlike the naming approach described for building automation in [13] where specific details, like the part number of the producer, had to be included in the name, in our approach the names of the packets are more general. The pre-fix of the name, e.g., Tomato/Block1/temp, has a hierarchical structure and in this example refers to a crop and then limits the reference to a block of the farm (location) and a particular type of (sensor) data. The capability of receiving data from a specific application despite using a generic name is enabled by the inclusion of an application name at the far end (post-fix) of the the interest's name (since this can be resolved to a corresponding producer in the context of the application by the Domain Resolution Engine).

2.3 Domain Resolution Engine

At the Domain Resolution Engine, the interests are classified into pre-defined domains based on the applications generating them. In spite of a wide range of applications that could potentially be built to use data generated from the setup, the domains of these applications are fairly constrained given the limited types of data generated by IoT systems (i.e., just temperature in the case of the example described here). A Domain is defined based on the tolerance to inaccuracies in data that the application under this category possesses and the set of hierarchies that the appli-

Table 2: Application to Domain mapping

Application	Domain
Temperature Control Feedback control Disinfect Control	Feedback
Decision Control Power supply regulation	Decision

Table 3: Context lookup table

Interest Name	Scope ID	User ID
/tomato/temp/decision	sid_1	uid_43
	sid_141	uid_14
	sid_65	uid_42, uid_453
/lettuce/temp/decision	sid_7	uid_233
	sid_42	uid_09
	sid_93	uid_22, uid_3

cation set could possibly query. Table 2 shows an example of application to domain mapping in a smart farm module. Note that even though interests generated across multiple domains could probably be mapped to the same producer, the classification helps the gateway to understand the tolerance limits of interested applications, thereby applying energy efficient forwarding strategies. This will be discussed in detail in section 2.6. Application based access control policies can be implemented at this layer to verify whether the requesting user has sufficient privileges to operate in the corresponding domain.[1]

2.4 Context Lookup

The authenticated interest packet is mapped to a "scope-node," denoted by a name in the domain's context lookup table based on the longest prefix match of the remainder of the interest's name. At this level the generic interest name is mapped to a specific set of one or more producers who can serve the requesting consumer's interest under the context of the categorized domain. This mapping plays a crucial role in maintaining data privacy as consumers cannot directly access a producer's data unless it is mapped at the context lookup table for a given application. The scope-node (addressed by a scope ID) mentioned above is part of a hierarchically structured scope trie for defined corresponding domains. This node defines a set of producers falling under its sub-tree that can service the interest based on parent-child relations of the trie defined by the domain. The scope trie will be discussed in detail in section 2.5. Table 3 shows a sample Context Lookup Table for a Feedback Domain in a smart farm setup, as shown in Figure 2.

The generic interest for temperature generated by a feedback control application of a thermostat in Block 14 of the smart farm is directly mapped to a scope (sid_141) that is serviced by a temperature sensor in Block 14 (one of the many blocks the farm is divided in). In the context of the feedback control action for the given topology, the thermostat in Block 14 will never require temperature data from blocks other than the one which it handles. As a result, hard-coding the lookup table to map a simple generic interest to the corresponding producer is justifiable. Similarly, in the case of an interest generated by a decision controller to make farming decisions based on temperature data of tomato crops distributed across multiple blocks, with each block supporting different types of plants, a relatively targeted interest name, as shown in the table, maps to a scope

[1]This could be overridden by a malicious user requesting data with an authorized application name appended in interest packets, but using it for an unauthorized application. However, such circumventions may not be effective as the unauthorized application would still be serviced with data under the domain for which it already has access rights and cannot get reliable data frpm outside its authorized domain, thereby limiting the scope of attack.

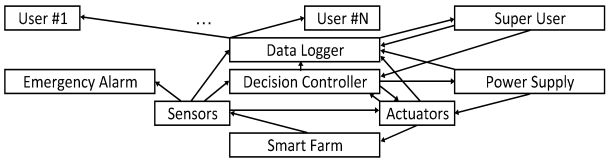

Figure 2: High-level view of a smart farm

node that has multiple nodes under its subtree mapped to producers in tomato plant blocks. Then, the interest can be served using data received from one or more producers depending on the parent-child relation in the scope tree and the forwarding strategy decisions discussed later (in section 2.6). In IoT systems with (frequent) topology changes (e.g., after every harvest, in case of the smart farm), this technique offers a simpler approach for applying the changes, since all the consumers need not be informed about the changes, but just remapped to new scope-nodes depending on the changes to the setup. This mapping further provides security by discarding packets whose prefixes do not match the defined keys for the corresponding application.

2.5 Scope Organization

The scope node to which the interest gets mapped is a part of a hierarchically structured context trie for each domain, organized based on the multiple hierarchies supported in it. It is at this layer that the forwarding strategies can be applied to improve the energy efficiency of the system. The leaf nodes in the trie correspond to forwarding blocks of each producer associated with the gateway. Interests mapped to these leaf nodes may or may not be looked up in the cache, depending on the tolerance of the domain, and then are forwarded to corresponding producers for data. The non-leaf nodes in the trie carry information about the nature of the interaction with children nodes (whether to forward to one or multiple nodes among the children) determined by the forwarding strategy for that domain (discussed further in section 2.6). In addition to this, if gateway routers are specialized to perform Named Function Networking (NFN) [16], the non-leaf scopes can take advantage of it to invoke code blocks to implement arithmetic functions (such as average, or maximum) on the data received from children nodes based on requirements of the application domain.

For example, consider an interest packet from an emergency alarm to monitor the temperature of a group of blocks in the smart farm. The interest gets mapped to the root node of a subtree encompassing leaf nodes that correspond to temperature producers in the sub-blocks via context lookup. Given that the emergency monitoring application is interested in the maximum temperature recorded among the temperature producers in the sub-block (as defined by the administrator or super-user in this case), the parent node invokes the function to return the maximum of the temperature data received from all leaf nodes of the sub-tree as its response data. Context-tries defined for all domains in the setup can be compressed to a single trie for improved space and performance efficiency.

2.6 Tolerance-based Forwarding

To design the forwarding strategy we make use of the idea that applications designed for IoT systems have varying degrees and types of tolerance to data depending on the application. For example, a thermostat requesting temperature

data of the smart farm water tank to perform feedback based temperature control requires the data it receives to be precise, accurate and real-time in order to perform feedback control effectively. Whereas a mobile-phone based application for a user to monitor the temperature of a farm need not necessarily be accurate and relatively stale data supplied to the application will not severely hurt its usage. Thus, it can be seen that different applications have different levels of tolerance to data, which can be exploited to make packet forwarding decisions with improved energy efficiency.

The proposed name resolution strategy facilitates this decision making because we can derive additional information about the nature of the application requesting data at the network layer itself to make scheduling decisions. This would not be possible in IP based networks as no appropriate information % other than source and destination address would be known to the gateway at the network layer; the packet would have to be processed at the application layer in order to implement a similar approach, which in turn would add to the latency in the network and the complexity and brittleness of the design.

We can broadly classify tolerance types of applications into: *accurate*, *approximate*, *stale*, and *arithmetic*. The applications for building automation and smart farm systems that we considered demonstrated one or more of these tolerance types, at varying degrees. In many cases applications demonstrate one or more tolerance types and can be grouped based on their application requirements into domains. The domains under which the applications are grouped are defined based on applications having similar tolerance.

The tolerance of a domain can be marked as *accurate* if it requires real-time accurate data from a producer. Interests from a thermostat for feedback based temperature control falls under this category. A domain can be marked as *stale* if it can accept relatively old data from a producer that could be already available in the Content Store of the gateway. Interests from mobile applications to monitor the temperature of a smart farm falls under this category. A domain can be marked as *approximate* if it can accept data from any producer that falls under the context of the requesting consumer. Interests from the decision controller for the temperature of tomato plants (in the smart farm to make farming action decisions), fall under this category as they can use data from any of the blocks holding tomato crops to make the decision. Finally, a domain can be marked as *arithmetic* if it expects data from multiple producers, either as individual packets, or as a single packet with some arithmetic operation performed on the data received from multiple producers (provided the gateway is NFN capable). An interest from the emergency alarm monitoring the temperature of a block expects the maximum temperature recorded among the data sensed by multiple producers in that block.

For *accurate* domains, the interest has to be served with the latest data from the system. In these cases, Content Store lookup can be skipped and interests can be directly forwarded to the producer, eliminating the time spent on the cache lookup. While in the case of *stale* domains the interest can be served with data from the Content Store itself (if available), provided it falls within the tolerance limit of the domain, or it can be forwarded if producer if data is not already available in the cache. Using the cache more often reduces the time spent by the embedded system in active mode, thereby reducing the consumed energy. In *approximate* domains, where the interest can be handled by any producer among a set of devices, interests can be forwarded to the currently active device (if known), or the most recently used device in the subset (as these devices are less likely to have transitioned back to sleep mode), thereby saving on the energy spent in waking up a sleeping device (the transition power penalty). For *arithmetic* domains, though it does not save energy by itself, by reducing interest forwarding to producers, applying named functions on data from producers reduces the number of packets transmitted through the network, saving energy at the network level, and also reduces congestion. If NFN is combined with caching of the computed values, then additional energy savings are materilized by not forwarding interests to (final) producers. Often, the arithmetic tolerance coexists with other tolerance types, thereby reducing the energy consumed by the overall operation.

Table 4: Tolerance based strategic forwarding decisions

Tolerance	Strategy
Accurate	Do not check cache; forward to Producer
Approximate	Forward to MRU Active Producer
Arithmetic	Employ Named Functions
Stale	Use Cached data if available

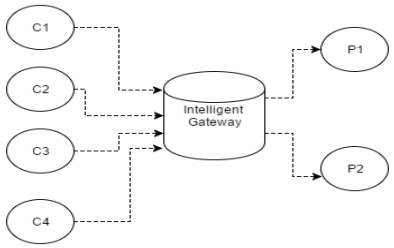

Figure 3: Sample temperature sensing setup

Table 4 summarizes strategic forwarding decisions for each tolerance type. This decision can be implemented at the context trie layer to determine how to pass the interest in a subtree rooted at a scope-node to which the interest has been mapped. This approach conserves energy by reducing interest forwarding to embedded devices and minimizing power mode transitions by taking advantage of tolerance to inaccuracies for applications. In the next section, we will discuss a sample scenario a demonstrate through a case study how this NDN strategic forwarding approach conserves energy compared to a straightforward (e.g., IP based) approach.

3. CASE STUDY

3.1 Scenario and Analysis

To demonstrate the energy conserved using the context based forwarding strategy proposed in this paper, we implemented this algorithm for a simple temperature control application shown in Figure 3. The setup comprises four consumers: a thermostat controller (C1), a decision controller (C2), a mobile application (C3) and an emergency alarm (C4). All four consumers are connected to an NFN capable intelligent gateway that is connected to two temperature sensors P1 and P2. For the given scenario, we compute the energy consumed by the system using the proposed forwarding strategy and compare it with the energy consumed using an IP based approach for the same scenario.

Figure 4 describes the scenario used in the analysis as a timeline of events, starting from an initial state where all

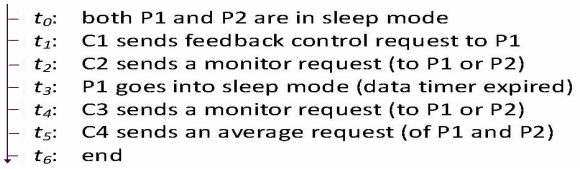

Figure 4: Timeline of events in the case study

producers are in sleep mode. We assume that there is no congestion using either of the approaches and the forwarding table of the router is statically filled with producer IDs to avoid packet flooding. At instant t_1, the thermostat (C1) uses temperature data from P1 to perform feedback based temperature control that falls under tolerance category *accurate*. Such feedback requests require "continuous" data from the producer for up to $(t_3 - t_1)$ time to complete the action. During this period, P1 remains active serving the C1 request. At t_2, the decision controller (C2) uses temperature data from a block that includes both P1 and P2 to understand tomato plant response. In the NDN scenario, this interest can be served by either P1 or P2, while in the IP scenario the controller asks and receives data from P2. This application falls under tolerance category *approximate*. At t_4 the mobile application (C3) uses data from P1 to monitor the temperature. This falls under category *stale*, since the data for monitoring applications need not be the absolutely latest. Finally, at t_5 the emergency alarm uses data to get the true (real-time) maximum temperature of the setup. This falls under the categories *arithmetic* and *accurate*. Then, all devices move to sleep mode at t_6.

Let us assume both devices are identical, with the same power ratings, and have single Sleep and Active modes. Let P_A and P_S be the power consumed in Active and Sleep mode, respectively. Let t be the amount of time taken to transition from Sleep to Active mode and vice-versa and P be the power consumed during mode transition.

For the given scenario, in IP based packet forwarding, P1 transitions to active mode at t_1 to serve the request from C1 and remains in Active mode till time t_3 when the "continuous data" timer runs out and transitions back to sleep mode. At t_4 it wakes up again to serve a data request from C3 and moves back to sleep. Then it wakes up again to serve a request from C4. The total amount of energy consumed by P1 during this series of events is the sum of:

$t_0 \rightarrow t_1 : P_S(t_1 - t_0)$
$t_1 \rightarrow t_3 : Pt + P_A(t_3 - t_1) + Pt$
$t_3 \rightarrow t_4 : P_S(t_4 - t_3) + Pt + Pt$
$t_4 \rightarrow t_5 : P_S(t_5 - t_4) + Pt + Pt$
$t_5 \rightarrow t_6 : P_S(t_6 - t_5)$

Similarly, P2 transitions to Active mode at t_2 to service a request from C2, moves back to Sleep mode and wakes up again at t_5 to serve a request from C5. The total amount of energy consumed by P2 during this series of events is the sum of:

$t_0 \rightarrow t_2 : P_S(t_2 - t_0)$
$t_2 \rightarrow t_5 : Pt + Pt + P_S(t_5 - t_2)$
$t_5 \rightarrow t_6 : Pt + Pt + P_S(t_6 - t_5)$

Repeating the scenario for NDN with the proposed forwarding strategy, P1 initially transitions to Active mode to serve C1 and moves back to sleep at t_3. However, at t_4, the request from C3 will not be forwarded to P1 as it can accept *stale* data, which would be serviced by the gateway itself from the latest cached data from P1 that it stored at t_3. It wakes up at t_5 to serve C5 and moves back to sleep. The energy consumed by P2 is given by the sum of:

$t_0 \rightarrow t_1 : P_S(t_1 - t_0)$
$t_1 \rightarrow t_3 : Pt + P_A(t_3 - t_1) + Pt$
$t_3 \rightarrow t_5 : P_S(t_5 - t_3) + Pt + Pt$
$t_5 \rightarrow t_6 : P_S(t_6 - t_5)$

For P2 the transition to Active mode at t_2 occurring in the IP scenario is skipped in the NDN case as the *approximate* tolerance of the C2 application would be satisfied by P1 directly, since it is already active. So the only event for which it wakes up is at t_5, to serve C5, and later moves back to sleep. The energy consumed by P2 is given by the sum of:

$t_0 \rightarrow t_5 : P_S(t_5 - t_0)$
$t_5 \rightarrow t_6 : Pt + Pt + P_S(t_6 - t_5)$

Thus, the additional energy consumed for the given scenario in the IP over the NDN approach is: $2Pt + P_S(t_4 - t_0)$.

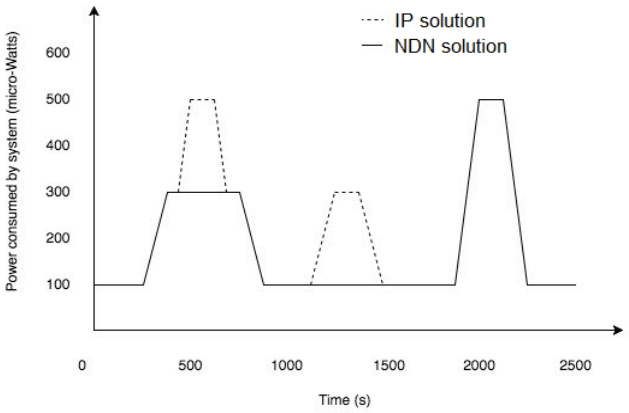

Figure 5: Power consumed by a temperature sensing module during the timeline of events in the case study. The dashed IP-case line is superimposed on the NDN-case solid line.

Figure 5 shows the power consumed by the setup at discrete time intervals in the IP and NDN scenarios, respectively, using MSP430FR5724 as the embedded platform to act as the producer and the power ratings from [15] and the implementation described below. It was found that the proposed technique conserves 23.90% of the energy compared to the IP based forwarding approach. Note that typical applications are more complex, with more overlapping requests likely to occur, where this approach promises to perform more efficiently. In the worst case, when there are no overlapping requests and all are accurate data requests, this approach performs exactly as the IP based approach. However, there could be a marginal increase in latency at the gateway due to the series of lookups to identify the context of an interest, but we believe that with hardware based implementation of hash lookups this overhead will be low.

3.2 Implementation

We implemented the proposed forwarding strategy in Python using dictionary based hash lookup tables for Domain Resolution and Context Lookup. The scope trie was implemented as a separate trie structure for each domain and compressed into a single data structure using a Ternary Search Trie based approach. The case study scenario was simulated using *MiniNet* and the forwarding strategy was applied at the gateway router. The IP scenario was implemented with static routing table entries to avoid packet flooding

4. RELATED WORK

Katsaros et al. [7] introduced a generic ICN architectural framework and showed how it can address emerging smart grid communication challenges by demonstrating the feasibility of an ICN solution for the support of real-time smart grid applications and quantifying the performance benefits brought by ICN over the host-centric paradigm.

Several optimization techniques have been proposed to improve the energy efficiency of ICN at various levels ranging from embedded system optimization to intelligent forwarding strategies. Fang et al., provide an extensive survey [4] on various proposed techniques adopted in ICN.

Past work in this area includes hardware optimization at content routers by shutting down the equipment or putting it to sleep when not in use [14], intelligent caching strategies [11], router placement strategies [10] and advanced SSD based storage strategies [3]. Most of these techniques focus on large-scale networks with a strong backbone infrastructure to handle the traffic.

However, for the IoT, with highly constrained devices, storage level optimizations or router placement strategies might not be effective. Clustering based approaches have been proven to improve node level energy consumption in Mobile Ad-hoc Networks (MANETs), with the Low-Energy Adaptive Clustering Hierarchy (LEACH) being one among the notable protocols in this category [5]. A CCN based architecture for MANETs was proposed in [1] and demonstrated an improvement in energy efficiency compared to IP based clustering, with reduction in hop count. In another CCN based routing protocol for MANETs proposed in [10], improvement in energy efficiency is achieved by distributing the network traffic across multiple nodes to reduce stress on a particular set of nodes transmitting popular content.

All these techniques make use of ICN architectural features to save on power consumption. None of the techniques make use of the additional information about the content that is communicated to improve the system. The strategy proposed in this paper takes advantage of this additional information about the content and the application requesting the content to make intelligent forwarding decisions, thereby saving energy. The novelty in our work is the extension of the idea to make forwarding decisions to multiple faces at the strategy layer described in [6]. Our forwarding strategy makes use of the context under which data is requested by a consumer to reduce network transactions and aid embedded system level optimizations.

5. CONCLUSION

We proposed a novel context based NDN forwarding strategy for energy efficient scheduling of data requests in IoT based systems. In this approach, the application (name) generating the request is appended to the name in the interest packets. Using this information, the gateway classifies the interest packets into pre-defined domains based on the nature of the application. The categorized interests are resolved into appropriate scopes in the context of users and domains. At the scope level, forwarding decisions on interests are made based on tolerance to inaccuracies in data determined for the domain, reusing cached data or data from already active devices wherever possible. This reduces the number of data requests forwarded to the embedded system compared to an IP based implementation.

We demonstrated an improvement in energy efficiency of about 24% with this technique for a simple scenario compared to the IP implementation. In addition to energy savings, this technique facilitates changes in system topology, QoS support and provides improved security using application-level access control of interests. It also simplifies naming as consumers need not maintain granular level information about producers to request data. With 5G promising to reduce the latency in communication across IoT devices and to maintain battery life of things up to ten years, the proposed strategy can take advantage of the improved bandwidth utilization to perform context based forwarding even in cases of time critical IoT applications, thereby expanding its scope of usage.

However, the proposed technique generates increased processing load at the gateway, which could limit the scalability of the system. Future work could consider a distributed hashing approach to decentralize the forwarding strategy and address these scalability concerns. Also, improved load balancing of the requests could be sought to avoid all requests being handled by a small set of devices.

6. REFERENCES

[1] Amadeo, M., and Molinaro, A. CHANET: A content-centric architecture for IEEE 802.11 MANETs. In *NoF* (2011).

[2] Burke, J., et al. Authenticated lighting control using Named Data Networking. Tech. rep., NDN-11, 2012.

[3] Choi, N., et al. In-network caching effect on optimal energy consumption in Content-Centric Networking. In *IEEE ICC* (2012).

[4] Fang, C., et al. A survey of green information-centric networking: Research issues and challenges. *IEEE Communications Surveys and Tutorials 17*, 3 (2015).

[5] Heinzelman, W., et al. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS* (2000).

[6] Jacobson, V., et al. Networking Named Content. In *ACM CoNEXT* (2009).

[7] Katsaros, K., et al. Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications. *IEEE Network 28*, 3 (2014).

[8] Krintz, C., et al. SmartFarm: Improving agriculture sustainability using modern information technology. In *ACM SIGKDD DSFEW* (2016).

[9] Landau, D. How 5G will Power the Future Internet of Things. *TECH INNOVATION, August* (2015).

[10] Lee, G., et al. An energy efficient routing protocol for CCN-based MANETs. *Int. J. Smart Home 7*, 1 (2013).

[11] Llorca, J., et al. Dynamic in-network caching for energy efficient content delivery. In *IEEE INFOCOM* (2013).

[12] Osseiran, A., et al. Scenarios for 5G mobile and wireless communications: the vision of the METIS project. *IEEE Communications Magazine 52*, 5 (2014).

[13] Shang, W., et al. Securing building management systems using Named Data Networking. *IEEE Network 28*, 3 (2014).

[14] Song, Y., et al. Power-aware traffic engineering with Named Data Networking. In *IEEE MSN* (2011).

[15] Texas Instruments. *MSP430FR572x Mixed-Signal Microcontrollers*. SLASE35B datasheet, 2016.

[16] Tschudin, C., and Sifalakis, M. Named Functions and Cached Computations. In *IEEE CCNC* (2014).