EPiC Series in Computing

Volume 56, 2018, Pages 73–83

Proceedings of the 5th International OMNeT++ Community Summit

Towards evaluating Named Data Networking for the IoT: A framework for OMNeT++

Amar Abane[12], Paul Muhlethaler[3], Samia Bouzefrane[1], Mehammed Daoui[2], and Abdella Battou[4]

[1] Conservatoire National des Arts et Metiers, Paris, France
[2] University Mouloud Mammeri of Tizi-Ouzou, Algeria
[3] Inria, Paris, France
[4] National Institute of Standards and Technology, MD, USA

Abstract

Named Data Networking is a promising architecture for emerging Internet applications such as the Internet of Things (IoT). Many studies have already investigated how NDN can be an alternative for IP in future IoT deployments. However, NDN-IoT propositions need accurate evaluation at network level and system level as well. This paper introduces an NDN framework for OMNeT++. Designed for low-end devices and gateways of the IoT, the framework is capable of simulating NDN scenarios at the boundary of the network and the system. The framework implementation is presented and used to study a typical aspect of NDN integration in IoT devices.

1 Introduction

Current IoT systems are deployed on the IP protocol suite that has been used to build the worldwide Internet decades ago. However, by involving constrained physical devices, the IoT puts the IP model to the test and highlights its limitations. For example, security is still focused on communication channels when the data itself needs to be secured. Moreover, IoT systems need efficient support for resource naming and discovery, which is not easy to deploy with IP in constrained infrastructures. Although dedicated adaptations such as CoAP and 6LoWPAN have been developed, under the hood, matching the IoT vision is still a challenge for IP [12].

While adapting IP for the IoT might be seen as *cutting corners*, alternative architectures based on the Information Centric Networking (ICN) paradigm promise to natively satisfy emerging Internet applications. One of these architectures is Named Data Networking (NDN) [15]. The NDN project was funded by NSF[1] under the Future Internet Architecture (FIA) program. The main entity in NDN is the content. Networking operations are performed on names, and hosts (without logical addresses) request named-content directly from the network. Native features come along with this principle, such as: communication without establishing end-to-end

[1] National Science Foundation

A. Förster, A. Udugama, A. Virdis and G. Nardini (eds.), OMNeT 2018 (EPiC Series in Computing, vol. 56), pp. 73–83

connections and name-to-address resolution. Moreover, no consumer-provider path or session needs to be maintained, which provides a native support of connection disruption resulting from mobility.

In recent years, many studies investigated the suitability of NDN for the IoT, making NDN more and more powerful. However, important challenges need to be addressed to build a strong NDN-IoT duo. Specifically, integrating NDN in low-end devices, over low-power wireless standards is mandatory due to the degree of freedom allowed by NDN in terms of unbounded name length, packet structure and semantics. NDN-based solutions for the IoT need accurate evaluation, and relative results should be provided further to convince industrials, investors and IP-enthusiasts. Examples of scenarios in which such evaluation can be useful are: NDN-based networking for robots [5], NDN for public safety and tactical networks, NDN for WSN [2] and NDN over IEEE802.15.4 [1].

Although some of these scenarios can be evaluated in testbeds, implementations are commonly based on realistic needs, and sometimes do not support certain features or "outlandish" scenarios. However, a pure network simulator does not allow to model internal interactions of a system. Therefore, experimental NDN-IoT design requires the best of both worlds: a tool capable to simulate network protocols and system-level interactions. We introduce in this paper our implementation of an NDN framework for the OMNeT++ [8] simulator[2]. OMNeT++ can be used to simulate wired and wireless networks as well as on-chip networks, and so on. We designed the NDN framework with the following objectives in mind: (i) Evaluating how a design affects NDN entities in a given topology/scenario. (ii) Evaluating packet processing approaches (e.g. compression). (iii) Providing a good visualization of NDN communications at network and system levels for testing and teaching purposes. (iv) Providing an easy-to-use framework to quickly experiment initial NDN parameters without additional implementation, particularly for wireless networks and constrained devices.

The rest of this paper is organized as follows: The first part presents an overview of NDN (Section 2) and our framework design (Section 3). In the second part, the NDN framework is used to study a key aspect of NDN in IoT (Section 4). Section 5 gives a summary with future directions.

2 Named Data Networking

In NDN, the content is named independently from the producer location. Names are hierarchical, URI-like and formed by a sequence of components separated by slashes. Applications are free to design their own naming scheme as routers do not interpret the whole name. A producer can append components to the initial name to provide more information about the content (e.g. timestamp, sequence number, etc.).

The NDN architecture uses two types of packets: Interest and Data. Each NDN node has three data structures to process packets: (i) a Forwarding Information Base (FIB): associates content names to Face(s) through which the content can be retrieved, (ii) a Pending Interest Table (PIT): stores each forwarded Interest associated to its incoming Face(s), while waiting to get the corresponding Data packet. The notion of Face represents a network device interface or a local application in an NDN node. (iii) A third optional structure, Content Store (CS) is used to store forwarded Data packets to satisfy similar future Interests; which provides native in-network caching in NDN.

Typically, a communication is initiated by the consumer, and operates according to the

[2]Source code available at https://github.com/amar-ox/NDNOMNeT

following steps: (1) The consumer requests a content by sending an Interest carrying the name of the content (e.g. *home/room1/temperature*). (2) A node that receives the Interest checks if matching Data already exists in its local CS: if the corresponding Data is found, it is sent back. Otherwise, the router checks if an Interest with the same name is already in the PIT; if so, the new Interest is not forwarded and only the incoming Face is added to the existing PIT entry. If no similar Interest is found in the PIT, the new Interest is forwarded according to the FIB information, and stored in the PIT. (3) When the producer receives the Interest, it sends back a Data packet containing the requested content, or a part of it. The Data packet follows the reverse path of the Interest following traces left in the PIT of each router. (4) When the Data packet reaches a router, it is forwarded to the Faces from which the corresponding Interests were received. After that, the router removes the entry from the PIT, and stores the recent Data packet in its CS.

Figure 1 summarizes the Interest and Data processing steps in a typical NDN node.

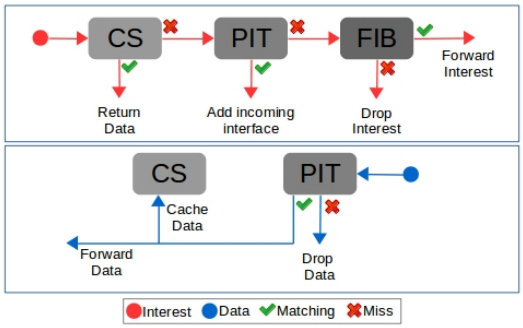

Figure 1: NDN forwarding process

NDN packets are encoded in the TLV (Type-Length-Value) format. TLV encoding represents an NDN packet as a collection of sub-TLVs. A TLV block is a sequence of bytes starting with a marker t, followed by its length l and a sequence of l bytes that represent the value.

3 Our framework design

OMNeT++ is not a native network simulator. However, ready-to-use domain-specific functionalities are provided by frameworks such as INET [6] which contains models for the IP protocol stack, link layer protocols, mobility, etc. Therefore, our framework is based on INET.

3.1 Overview

The NDN core module is designed as a network layer (*NdnL3*) that implements the network layer interface of INET (*INetworkLayer*). Within an NDN host, *NdnL3* is connected to the upper layer; expected to be the application layer, and the lower layer consisting on wired/wireless network interfaces. However, the NDN network layer can run on top of (or beside) IP with minimal adaptations. Following the modular approach of OMNeT++, we designed the NDN entities as independent modules included in the *NdnL3* module (Figure 2).

OMNeT++ modules use message passing through connected *Gates* to communicate. The notion of Gate provides a native abstraction of the Face concept; making the *NdnL3* module interact with upper (i.e. application) and lower (i.e. link) layers in a transparent way.

Due to space limitations, we report only the main components we implemented and a description of the packet representation we used.

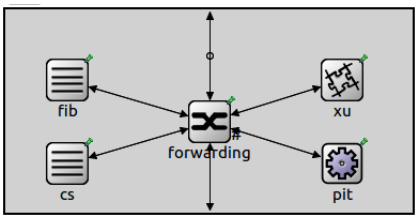

Figure 2: NDN network layer (NdnL3)

3.2 Host and Application Modules

NDN hosts: To represent NDN-based IoT devices, a base NDN wireless host (*NdnWirelessHostBase*) is implemented. It includes the typical wireless host components and the NDN layer (*NdnL3*). This module is used directly as a relay node since it does not include any application (Figure 3). By extending *NdnWirelessHostBase*, a typical IoT end-device is created (*NdnWirelessHost*) with consumer and/or producer applications.

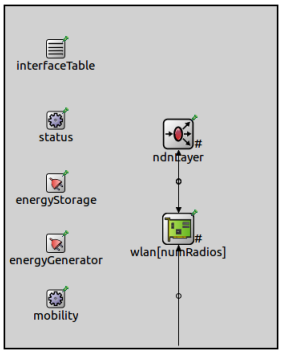

Figure 3: NDN relay node (NdnWirelessHostBase)

Applications: Two base applications are implemented. A producer (*ProducerAppBase*) with the following parameters: (i) *prefix*; under which the content is produced. (ii) *dataLength*; if not provided, the TLV length is used. (iii) *startTime*. (iv) *stopTime*. (v) *dataFreshness* [13]. A consumer (*ConsumerAppBase*) with the following parameters: (i) *startTime*. (ii) *stopTime*. (iii) *prefix*; for which Interests are issued, with additional name components (e.g. sequential number) (iv) *interestLength*; if not provided, the TLV length of the Interest is computed and used. (v) *sendInterval*; time to wait between two issued Interests. (vi) *numInterests*; number of Interest to issue. (vii) *interestReTx*; maximum number of Interest retransmissions. (viii) *interestLifetime*; value of the Interest lifetime field.

Figure 4 shows a typical NDN end-device with applications (*NdnWirelessHost*).

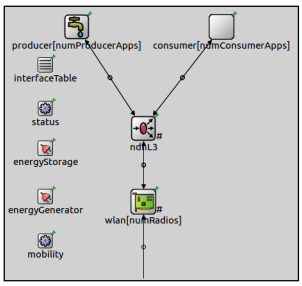

Figure 4: NDN end-device (NdnWirelessHost)

3.3 NDN layer Modules

Pending Interest Table: *IPit* is an abstraction of the PIT. It includes a typical entry which stores the following information: (i) A copy of the forwarded Interest, (ii) Incoming Face(s) of the Interest, (iii) Face(s) to which the Interest is forwarded, (iv) Expire time of the entry and, (v) Source MAC. NDN-to-MAC mapping is discussed in the next Section.

IPit provides the following functions: (i) Lookup, (ii) Create, (iii) Remove, (iv) Update, and (v) Print.

On Interest timeout, *IPit* implementations can emit a signal with a notification that includes the Interest, its incoming Face(s) and source MAC address(es).

A base implementation (*PitBase*) of *IPit* is provided with the following parameters: (i) *interestLifetime*. If no value is provided, the lifetime value of the Interest is used instead. (ii) *maxSize*. When the PIT is full, no Interest can be forwarded.

Forwarding Information Base: *IFib* is an abstraction of the FIB. It includes a typical entry which keeps the following information: (i) Name prefix, (ii) Face(s), (iii) Destination MAC address, (iv) Expire time of the entry (optional).

IFib provides the following functions: (i) Lookup, (ii) Create, (iii) Remove, (iv) Update, (v) Register prefix; used to create an entry to the local application and, (vi) Print.

When a prefix expires, *IFib* implementations can emit a signal with a notification that includes the prefix and its corresponding Face(s) and MAC address(es).

A base implementation of *IFib* (*FibBase*) is provided with the following user parameters: (i) *entryLifetime* and, (ii) *maxSize*.

Content Store: *ICs* is an abstraction of the CS. It includes a typical entry which stores the following information: (i) A copy of the Data packet and, (ii) A stale flag to manage the freshness of the Data [13].

ICs provides the following functions: (i) Lookup, (ii) Add, (iii) Remove, (iv) Update freshness and, (v) Print.

A base implementation of *ICs* (*CsBase*) is provided. It uses the FIFO replacing policy and the cache size can be defined (*maxSize*).

Experimental unit: Some internal NDN processes that can be imagined to improve efficiency cannot be assimilated to the forwarding strategy, and are not related to native NDN entities (i.e. PIT, FIB, CS). Packet compression and Fuzzy logic NDN forwarding [10] can be cited as examples of such processes. To provide a clean and flexible way to implement such

experimental design without disturbing the NDN base implementation, an eXperimental Unit module (XU) is included.

This module is designed to evaluate future AI-based operations such as semantics extraction from names, intelligent routing, etc.

IXu is an abstraction of the eXperimental Unit. The following signals can be emitted by *IXu* implementations to notify on processing progress: (i) Packet received, (ii) Packet processing begin, (iii) Packet processing end, (iv) Packet processing error and, (v) Packet processing success.

Forwarding: The forwarding module is the main component of our design and it is connected to PIT, FIB, CS and XU.

The forwarding process is abstracted in the *IForwarding* interface which provides the following functions: (i) *processLLInterest*; process Interest coming from lower layer. (ii) *processLLData*; process Data coming from lower layer. (iii) *processHLInterest*; process Interest coming from higher layer (i.e. application layer). (iv) *processHLData*; process Data coming from higher layer. (v) *forwardInterestToRemote*; forward Interest to remote Face (e.g. radio). (vi) *forwardDataToRemote*; forward Data to remote Face. (vii) *forwardInterestToLocal*; forward Interest to local application. (viii) *forwardDataToLocal*; forward Data to local application. (ix) *mapToMAC*; encapsulate Interest or Data considering the given unicast or broadcast MAC address. (x) *onInterestTimeout*; when receiving an Interest timeout signal. (xi) *onPrefixExpired*; when receiving a prefix expired signal.

The communication with PIT, FIB, CS and XU can be either through message passing or direct module access as provided by OMNeT++. However, the message passing communication allows to model time processing of each module separately and independently. The forwarding module implementation is intended to subscribe to PIT, FIB and XU signals in order to handle *NdnL3* events.

The current framework implementation includes a base forwarding strategy (*ForwardingBase*). It supports the following parameters: (i) *ndnMacMapping*; code of the NDN-to-MAC mapping to use. (ii) *cacheUnsolicited*; whether to cache unsolicited Data packets or not. (iii) *forwarding*; whether to forward Interests (router) or not (end-device).

3.4 Messages and packets

To represent NDN packets, we use the OMNeT++ message representation in order to provide an easy way to create packets and access their fields. However, for packet-related evaluation, a set of tool functions are provided to generate the TLV representation and to compute the actual size of a packet from the OMNeT++ packet representation.

For evaluation purposes, Interest and Data have a common superclass (*NdnPacket*) which includes non-NDN fields. These fields are of two types: (i) inherited from OMNeT++ *Packet* class [9] and, (ii) additional fields that include a sequence number, a type (i.e. Interest or Data), a hop count and other fields that help for statistic or experimental implementations.

4 Use case: NDN over low-rate wireless technology

In this section, we use our framework to study an aspect of integrating NDN over a low-power low-rate wireless technology. Considering the IEEE802.15.4 standard, one of the enablers of the IoT, fundamental questions need to be explored and revisited for NDN as described in the following.

4.1 NDN wireless forwarding strategies

In one hand, wireless forwarding strategies are generally based on a *flood-and-learn* mechanism [3, 11, 14]. Although lightweight mechanisms for constrained devices are rare in literature, a simple one has been proposed in [4]: Reactive Optimistic Name-based Routing (RONR) and detailed further.

On the other hand, different approaches may be used to map learned information, which are in form of NDN names, to MAC addresses. This question has been investigated in [7] where different mappings are discussed: (i) broadcast for Interest and Data (IBDB), (ii) broadcast for Interest and unicast for Data (IBDU), (iii) unicast for Interest and broadcast for Data (IUDB), (iv) unicast for Interest and Data (IUDU). The study is particularly focused on CPU usage and system wakeup comparison.

To show a typical use of our framework, we simulate a wireless forwarding mechanism to enhance the NDN-to-MAC exploration cited above, by studying both network performance and device state in static and mobile scenarios, under different parameters. Therefore, we combined RONR with a related solution for wireless NDN forwarding [3]. The implemented mechanism (*ForwardingBase*) works as follows:

1. When a consumer issues an Interest for unknown content, an Interest flooding is performed by the relay-nodes until reaching the producer. Interest retransmissions are randomly delayed to minimize the risk of collisions.

2. After the first Data is sent back by the producer, each relay-node that solicited the Data creates a temporary FIB entry with the corresponding prefix.

3. Subsequent Interests are forwarded only by the nodes that have the corresponding entry in the FIB.

4. The Interest flooding phase is performed again when an entry is expired or after an Interest timeout (a flood flag is used by the consumer).

4.2 Simulation and results

We configured a local wireless network composed of *9* fixed routers, *5* mobile producers and a mobile consumer, in an area of *100m* x *100m*. Each producer has a unique content of *30* pieces that are randomly requested by the consumer. Figure 5 shows the OMNeT++ visualization of the simulated network. Caching is disabled in order to study the behavior in a complete consumer-producer path. The rest of the simulation parameters are reported in Table 1.

Parameter	Value
MAC layer	CSMA with *250kbps* data rate
Communication range	*30m*
Interests number	*100*
Max Interest retransmissions	*3*
Interest send interval	*1s*
Interest size	*30 kb*
Data size	*100 kb*
Mobility model	Random Waypoint

Table 1: Simulation parameters

Each simulation is executed with 7 random seeds and the average results are reported. The following metrics are measured: (i) number of duplicated Interests received by relay-nodes, (ii) mean/max PIT size and PIT lookup in relay-nodes, (ii) number of collisions observed by all nodes, (iv) Interest satisfaction rate and Interest-Data RTT measured by the consumer.

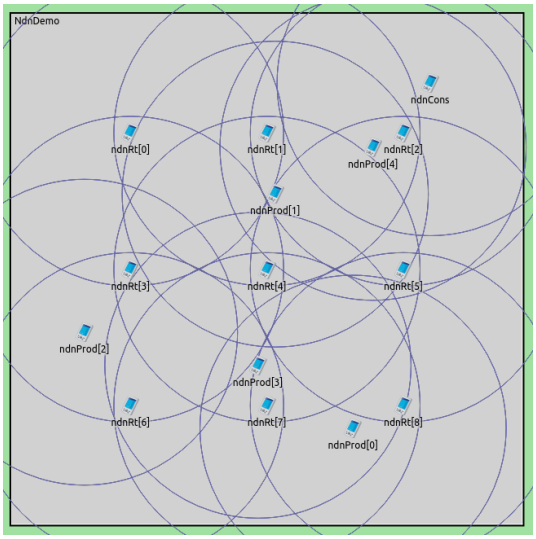

Figure 5: Simulated scenario

As expected, the two mappings that use Interest broadcast (i.e. IBDB and IBDU) always achieve the highest Interest duplication value, as well as PIT size, PIT lookups and collisions (Figure 6 to 9). Indeed, the broadcast mechanism generates more Interests in the network, which leads to more PIT lookups, more retransmissions due to flooding, and thus more collisions. For the same reason, Interest and Data unicast (IUDU) causes the lowest number of Interest duplications and PIT lookups.

However, contrary to what we might expect, IUDU does not cause the lowest number of collisions, for both mobile and static scenarios as shown in Figure 9. The lowest number of collisions is achieved by the IUDB mapping, followed by IUDU. This can be explained by the fact that Data broadcast prevents some nodes from unnecessary Interest retransmission. During the flooding phase, nodes can overhear Data broadcast, while Interest unicast phase reduces the number of potential forwarders in consumer-producer direction. IUDB can be considered as a good compromise when flood-and-learn techniques are used, especially in mobile scenarios.

We observe that changing from one mapping to another has the same effect in mobile and static scenarios in terms of Interest duplication, PIT size, PIT lookups and collision number. However, performances are slightly higher in the static case, as we can expect (Figure 10 and 11).

IUDU achieves the best performance in the static scenario (Figure 10 and 11). This indicates that the Interest broadcast only causes extra computation without improving the performance. The study in [7] also reports that IUDU mapping provides the highest Interest satisfaction rate in static scenario. However, in the mobility scenario, IUDU only achieves the second better performance, while the best one is achieved by the IBDU mapping. This is certainly due to the exploratory nature of broadcasting Interests. Although this feature is not so helpful in static configurations, it helps to find the content producer when the hosts are mobile.

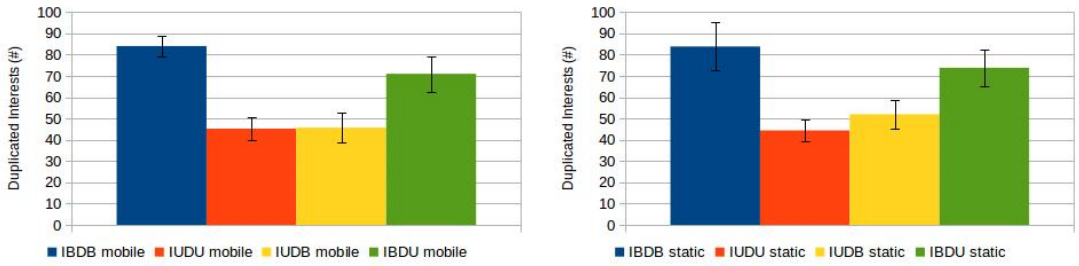

Figure 6: Interest duplication

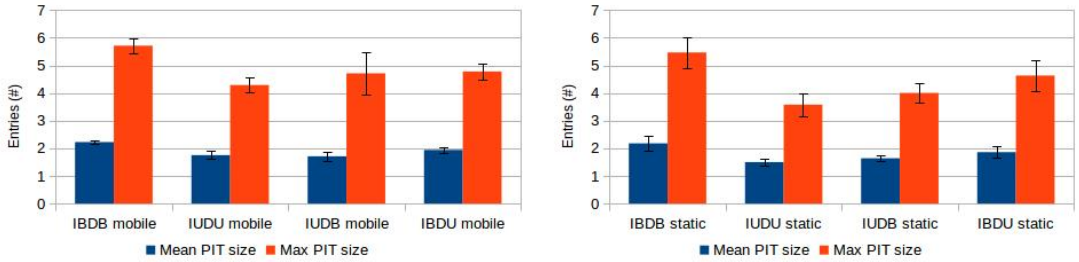

Figure 7: PIT size

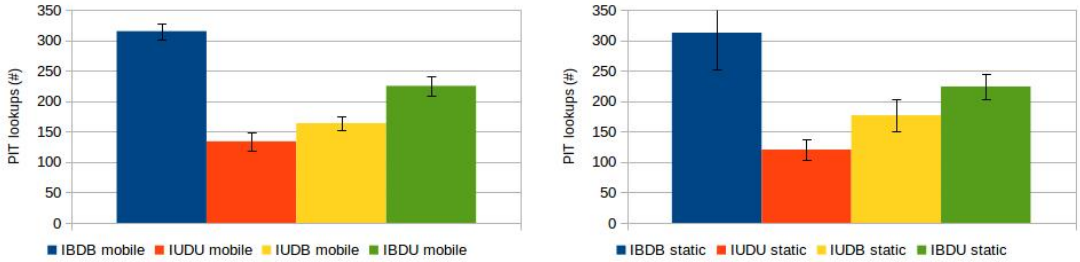

Figure 8: PIT lookups

5 Summary and future work

The proposed NDN framework is intended to be used to evaluate NDN for low-end IoT solutions, by focusing on both local wireless networks and system state of constrained devices. Although for some evaluations, a simulation cannot serve as a suitable substitute for a real-world testbed, the proposed framework can quickly provide accurate results for different aspects of the solution. For example, it can be used to evaluate the following aspects: (i) packet fragmentation, encoding/decoding and other NDN packet processing, (ii) secured and trust-based wireless forwarding, (iii) hybrid NDN-IP gateways and, (iv) impact of NDN names to MAC addresses mapping approaches.

The current implementation does not support all NDN features; the Interest forwarding

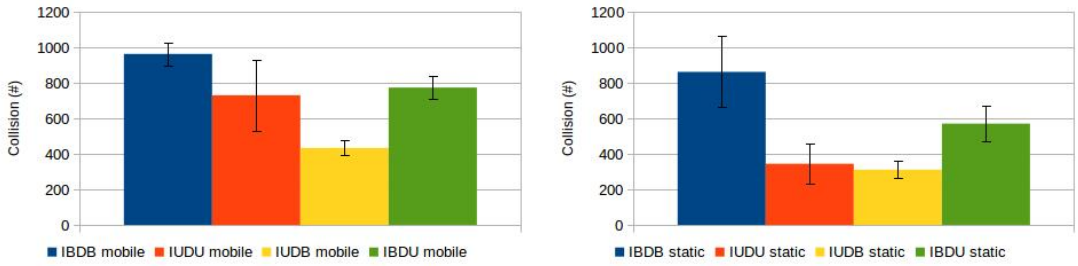

Figure 9: Collision number

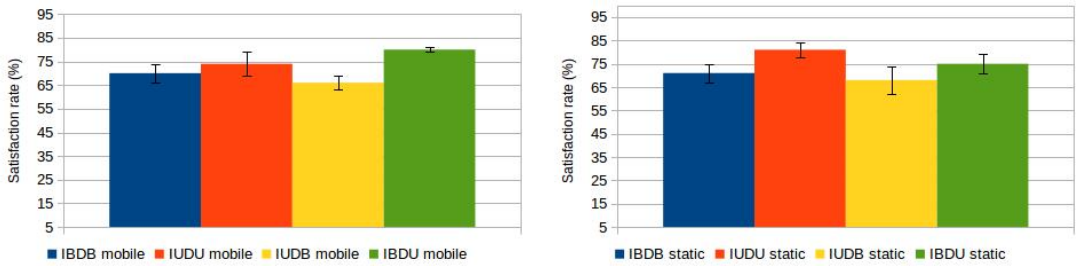

Figure 10: Consumer Interest satisfaction rate

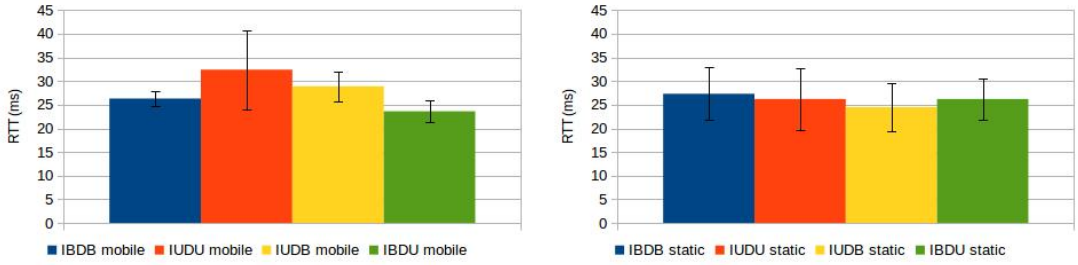

Figure 11: Consumer Interest-Data RTT

implemented is based on names only, while hop limit and lifetime should be considered also. As a future work, we plan to improve the framework with two main features: (i) A fully-customizable forwarding strategy that take the most of OMNeT++ module parameters. (ii) A memory usage model for NDN entities, as well as accurate packet processing time measurement, in addition to the mobility and energy consumption models provided by INET.

References

[1] Amar Abane, Mehammed Daoui, Samia Bouzefrane, and Paul Muhlethaler. Ndn-over-zigbee: A zigbee support for named data networking. *Future Generation Computer Systems*, 2017.

[2] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton. Named data networking: A natural design for data collection in wireless sensor networks. In *2013 IFIP Wireless Days (WD)*, pages 1–6, Nov 2013.

[3] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Forwarding strategies in named data wireless ad hoc networks: Design and evaluation. *Journal of Network and Computer Applications*, 50(Supplement C):148 – 158, 2015.

[4] Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C. Schmidt, and Matthias Wählisch. Information centric networking in the iot: Experiments with ndn in the wild. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ACM-ICN '14, pages 77–86, New York, NY, USA, 2014. ACM.

[5] Loïc Dauphin, Emmanuel Baccelli, Cedric Adjih, and Hauke Petersen. NDN-based IoT Robotics. ACM ICN'17 - 4th ACM Conference on Information-Centric Networking, September 2017. Poster.

[6] INET. "INET Framework". [Online]; https://inet.omnetpp.org/.

[7] Peter Kietzmann, Cenk Gündogan, Thomas C. Schmidt, Oliver Hahm, and Matthias Wählisch. The Need for a Name to MAC Address Mapping in NDN: Towards Quantifying the Resource Gain. In *ACM ICN 2017 - 4th ACM Conference on Information-Centric Networking* , Berlin, Germany, September 2017.

[8] OpenSim Ltd. "OMNeT++". [Online]; https://www.omnetpp.org/.

[9] OpenSim Ltd. "OMNeT++ Simulation Library". [Online]; https://www.omnetpp.org/doc/omnetpp/api/classomnetpp_1_1cPacket.html.

[10] Spyridon Mastorakis, Kevin Chan, Bongjun Ko, Alexander Afanasyev, and Lixia Zhang. Experimentation with fuzzy interest forwarding in named data networking. *CoRR*, abs/1802.03072, 2018.

[11] Meisel Michael, Pappas Vasileios, and Zhang Lixia. Listen first, broadcast later: Topology-agnostic forwarding under high dynamics. In *Annual conference of international technology alliance in network and information science*, 2010.

[12] Wentao Shang, Yingdi Yu, Ralph Droms, and Lixia Zhang. Challenges in IoT networking via TCP/IP architecture. Technical Report NDN-0038, NDN, February 2016.

[13] NDN Team. "NDN Packet Format Specification". [Online]; http://named-data.net/doc/NDN-packet-spec/current/.

[14] Lucas Wang, Alexander Afanasyev, Romain Kuntz, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. Rapid traffic information dissemination using named data. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 7–12, New York, NY, USA, 2012. ACM.

[15] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named Data Networking. *ACM SIG-COMM Computer Communication Review*, 44(3):66–77, July 2014.