Performance Evaluation of the NDN Data Plane Using Statistical Model Checking

Siham Khoussi[1,2], Ayoub Nouri[1], Junxiao Shi[2]
James Filliben[2], Lotfi Benmohamed[2], Abdella Battou[2], and Saddek Bensalem[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble Institute of Engineering Univ. Grenoble Alpes, VERIMAG, 38000 Grenoble, France
[2] National Institute of Standards and Technology, Gaithersburg, MD, 20899, USA

Abstract. Named Data Networking (NDN) is an emerging internet architecture that addresses weaknesses of the Internet Protocol (IP). Since Internet users and applications have demonstrated an ever-increasing need for high speed packet forwarding, research groups have investigated different designs and implementations for fast NDN data plane forwarders and claimed they were capable of achieving high throughput rates. However, the correctness of these statements is not supported by any verification technique or formal proof. In this paper, we propose using a formal model-based approach to overcome this issue. We consider the NDN-DPDK prototype implementation of a forwarder developed at the National Institute of Standards and Technology (NIST), which leverages concurrency to enhance overall quality of service. We use our approach to improve its design and to formally demonstrate that it can achieve high throughput rates.

Keywords: NDN · SMC · Model-based design · Networking

1 Introduction

With the ever growing number of communicating devices, their intensive information usage and the increasingly critical security issues, research groups have recognized the limitations of the current Internet architecture based on the internet protocol (IP) [12]. Information-Centric Networking (ICN) is a new paradigm that transforms the Internet from a host-centric paradigm, as we know it today, to an end-to-end paradigm focusing on the content, hence more appropriate to our modern communication practices. It promises better security, mobility and scalability.

Several research projects grew out of ICN. Examples include content-centric architecture, Data Oriented Network Architecture and many others [17], but one project stood out the most and was sponsored by the National Science

---

The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Foundation (NSF) called Named Data Networking (NDN) [19]. NDN is gaining rapidly in popularity and has even started being advertised by major networking players [1].

IP was designed to answer a different challenge, that is of creating a communication network, where packets named only communication endpoints. The NDN project proposes to generalizes this setting, such that packets can name other objects, i.e. *"NDN changes the semantics of network services from delivering the packet to a given destination address to fetching data identified by a given name. The name in an NDN packet can name anything - an endpoint, a data chunk in a movie or a book, a command to turn on some lights, etc. "* [19]. This simple change has deep implications in term of routers forwarding performance since data needs to be fetched from an initially unknown location.

Being a new concept, NDN (Section 2) has not undergone any formal verification work yet. The initial phase of the project was meant to come up with proof-of-concept prototypes for the proposed architecture. This has lead to a plethora of less performing implementations in terms of packets' forwarding rates (throughput). A lot of effort was then directed to optimizing NDN forwarders' performances by trying different data structures (Hash maps) and targeting different hardware (GP-GPU). Unfortunately, validation was mainly carried using pure simulation and testing techniques.

In this work, we take a step back and try to tackle the performance problem differently. We consider a model-based approach that allows for rigorous reasoning and formal verification (Section 3). In particular, we rely on the $\mathcal{S}$BIP framework [11, 16] offering a stochastic component-based modeling formalism and Statistical Model Checking (SMC) engine. $\mathcal{S}$BIP is used along an iterative and systematic design process which consists of four phases (1) building a parameterized functional system model, which does not include performance (2) run a corresponding implementation in order to collect context information and performance measurements, characterized as probability distribution functions, (3) use these distributions to create a stochastic timed performance model and (4) use SMC to verify that the obtained model satisfies requirements of interest.

This approach is applied to verify that the NDN Data Plane Development Kit (NDN-DPDK) (an effort to develop a high performance forwarder for NDN networks at the National Institute of Standards and Technology (NIST)) can perform at high packet forwarding rates (Section 4). We investigate different design alternatives regarding concurrency (number of threads), system dimensioning (queues sizes) and deployment (mapping threads to multi-core). Using our approach, we were able to figure out what are the best design parameters to achieve higher performances (Section 5). These were taken into account by the NDN developers at NIST to enhance the ongoing design and implementation. To the best of our knowledge, this is the first work using formal methods in the context of the NDN project.

2   Named Data Networking

This section describes the NDN protocol and introduces the NDN-DPDK forwarder being designed and implemented at NIST.

2.1   Overview

NDN is a new Internet architecture different from IP. Its core design is exclusively based on naming contents rather than end points (IP addresses in the case of IP) and its routing is based on name prefix lookups [9].

The protocol supports three types of packets, namely *Interest, Data* and *Nack*. Interests are consumer requests sent to a network and Data packets are content producers replies. The Nack lets the forwarder know of the network's inability to forward Interests further. One of NDN's advantages is its ability to cache content (Data) everywhere the Data packet propagates, making the NDN router stateful. Thus, future Interests are no longer required to fetch the content from the source, instead Data could be retrieved directly from a closer node that has a cached copy.

Packets in NDN travel throughout a network as follow: first a client application sends an Interest with a name prefix that represents the requested content. Names in NDN are hierarchical (e.g., /YouTube/Alex/video1.mpg denotes a YouTube video called Video1.mpg by a Youtuber Alex). Then, this packet is forwarded by the network nodes based on its name prefix. Finally, this Interest is satisfied with Data by the original source that produced this content or by intermediate routers that cached it due to previous requests. It is also crucial to note that consecutive transmissions of Interest packets with similar name prefix might not lead to the same path each time, but could rather be forwarded along different paths each time a request is made, depending on the forwarding strategy in place. This means that the same Data could originate from different sources (producers or caches).

The NDN forwarding daemon (NFD) [3], has three different data structures: *Pending Interest Table (PIT), Content Store (CS)* and *Forwarding Interest Base (FIB).* The packet processing, according to the NDN protocol, is as follows:

1 – For Interests, the forwarder, upon receiving an Interest, starts off by querying the CS for possible copies of the Data, if a CS match is found during this operation, the cached Data is returned downstream towards the client. Otherwise, an entry is created in the PIT with its source and destination faces (communication channels that the forwarder uses for packet forwarding) for record keeping. Using the PIT, the forwarder determines whether the Interest is looped in the network by checking a global unique number called Nonce in the Interest against existing previous PIT entries. If a duplicate nonce is found the Interest is dropped and a Nack of reason *Duplicate* is sent towards the requester. Otherwise, the FIB is queried for a possible next hop to forward the Interest towards an upstream node; if there is no FIB match, the Interest is immediately dropped and replied with a Nack of reason *No Route*.

2 – For Data, the forwarder starts off by querying the PIT. If a PIT entry is found, the Data is sent to downstream nodes listed in the PIT entry, then the PIT arms a timer to signal the deletion of this entry and a copy of the Data is immediately stored in the CS for future queries. If no record is found in the PIT, the Data is considered malicious and discarded.

2.2   The NDN-DPDK Forwarder

NDN-DPDK is a forwarder developed at NIST to follow the NDN protocol and to leverage concurrency. In this paper, we evaluate its capacity to achieve high throughput using Statistical Model Checking (SMC).

The NDN-DPDK forwarder's data plane has three stages: input, forwarding, and output (Fig. 1). Each stage is implemented as one or more threads pinned to CPU cores, allocated during initialization. **Input** threads receive packets from a Network Interface Card (NIC) through faces, decode them, and dispatch them to forwarding threads. The **forwarding** thread processes Interest, Data, or Nack packets according to the NDN protocol. **Output** threads send packets via faces then queue them for transmission on their respective NIC.
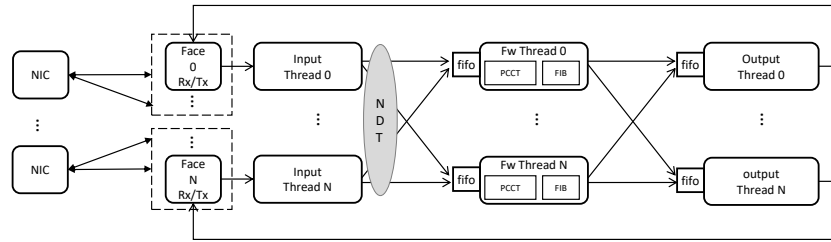

Fig. 1: Diagram of the NDN-DPDK forwarder

During forwarder initialization, each hardware NIC is provided with a large memory pool to place incoming packets. The input thread continuously polls the NIC to obtain bursts of 64 received packets. Then decodes, reassembles fragmented packets, and drops malformed ones. Then, it dispatches each packet to the responsible forwarding thread which is determined as follows: (a) For an Interest, the input thread computes SipHash of its first two name components and queries the last 16 bits of the hash value in the Name Dispatch Table (NDT), a 65,536 entry lookup table configured by the operator, to select the forwarding thread. (b) Data and Nack carry a 1-byte field in the packet header which indicates the forwarding thread that handled the corresponding Interest. Once identified, Data (or Nack) will be dispatched to the same one.

The forwarding thread receives packets dispatched by input threads through a queue. It processes each packet according to the NDN protocol, using two data structures both implemented as hash tables: (a) The FIB records where the content might be available and which forwarding strategy is responsible for the name prefix. (b) The PIT-CS Composite Table (PCCT) records which downstream node requested a piece of content, and also serves as a content cache; it combines the PIT and CS found in a traditional NDN forwarder.

The output thread retrieves outgoing packets from forwarding threads through a queue. Packets are fragmented if necessary and queued for transmission on a NIC. The NIC driver automatically frees the memory used by packets after their transmission, making it available for newly arrived packets.

3 Formal Model-based Approach

In this section, we describe the methodology used in this study which includes the underlying modeling formalism as well as the associated analysis technique.

3.1 Overview

Our methodology (Fig. 2) is based on a formal model. In order to evaluate a system's performance, its model must be faithful, i.e. it must reflect the real characteristics and behavior of this system. Moreover, to allow for exhaustive analyses, this model needs to be formally defined and the technique used for analysis needs to be trustworthy and scalable. Our approach adheres to these principles in two ways. First, by relying on the $\mathcal{S}$BIP formal framework (introduced below) that encompasses a stochastic component-based modeling formalism and an SMC engine for analysis [11]. Second, by providing a method for systematically building formal stochastic models for verification that combine accurate performance information with the functional behavior of the system.
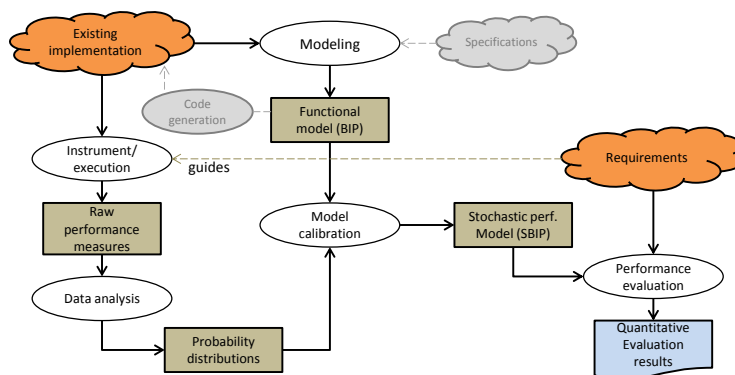

Fig. 2: Performance evaluation approach for NDN data plane.

This approach takes a functional system model and a set of requirements to verify. The functional model could be obtained from a high-level specification or an existing implementation (we use the latter in this paper). The system's implementation which could also be obtained by automatic code generation, is instrumented and used to collect performance measurements regarding the requirements of interest, e.g. throughput. These measurements are analyzed and characterized in the form of probability density functions with the help of statistical techniques such as sensitivity analysis and distribution fitting. The obtained probability density functions are then introduced in the functional model using a well defined calibration procedure [15]. The latter produces a stochastic timed model (when measurements concern time), which will be analyzed using the SMC engine.

Note that the considered models in this approach or workflow can be parameterized with respect to different aspects that we want to analyze and explore.

Basically, the defined components types are designed to be instantiated in different context, e.g. with different probability density functions thus showing different performance behaviors. While, the model considered for analysis using SMC is a specific instance for which all the parameters are fixed, some degree of parameterization is still allowed on the verified requirements.

3.2   Stochastic Component-based Modeling in BIP

BIP (Behavior, Interaction, Priority) is a highly expressive component based framework for rigorous system design [6]. It allows the construction of complex, hierarchically structured models from atomic components characterized by their behavior and their interfaces. Such components are transition systems enriched with variables. Transitions are used to move from a source to a destination location. Each time a transition is taken, component variables may be assigned new values, computed by user-defined C/C++ functions. Composition of BIP components is expressed by layered application of interactions and priorities. Interactions express synchronization constraints between actions of the composed components while priorities are used to filter among possible interactions e.g. to express scheduling policies.

The stochastic semantics of BIP were initially introduced in [14] and recently extended for real-time systems in [16]. They enable the definition of stochastic components encompassing probabilistic variables updated according to user-defined probability distributions. The underlying mathematical model behind this is a Discrete Time Markov Chain. These are modeled as classical BIP components augmented with probabilistic variables as shown in Fig. 3 and depicts a client behavior in a client-server setting where the client issues a request (**snd**) each p time units. The period p is set probabilistically by

Fig. 3: A stochastic BIP component; client behavior issuing requests each time unit p.

sampling a distribution function ($p \triangleright$) given as a parameter of the model. Time is introduced by explicit **tick** transitions and waiting is modeled by exclusive guards on the **tick** and **snd** transitions with respect to time (captured in this example by the variable t).
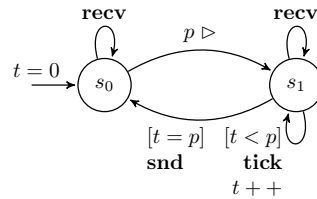
3.3   Statistical Model Checking in a Nutshell

Statistical model-checking (*SMC*) [8, 18] is a formal verification method that combines simulation with statistical reasoning to provide quantitative answers on whether a stochastic system satisfies some requirements. It was successfully used in various domains such as biology [7], communication [4] and avionics [5]. It has the advantage to be applicable to models and implementations (provided that they meet specific assumptions) in addition to capturing rare events. The

$\mathcal{S}$BIP SMC engine [11] implements well-known statistical algorithms for stochastic systems verification, namely, Hypothesis Testing [18], Probability Estimation [8] and Rare Events. In addition, it provides an automated parameters exploration procedure. The tools take as inputs a stochastic BIP model, a Linear-time/Metric Temporal Logic (LTL/MTL) property to check and a set of confidence parameters required by the statistical test.

4   NDN-DPDK Modeling

In this section we present the modeling process of the NDN-DPDK from a functional to a stochastic timed model for throughput evaluation.

4.1   A Parameterized Functional BIP Model

Fig. 4 depicts the BIP model of the NDN-DPDK forwarder introduced in Section 2 which shows its architecture in terms of interacting BIP components that can easily be matched to the ones in Fig. 1. The presented model is parameterized with respect to the number of components, their mapping into specific CPU cores, FIFOs sizes, etc. Due to space limitation, we present in [10] the behaviors of all the components of the NDN-DPDK forwarder in Fig. 4. It is worth mentioning that the model is initially purely functional and untimed. Time is introduced later through the calibration procedure.
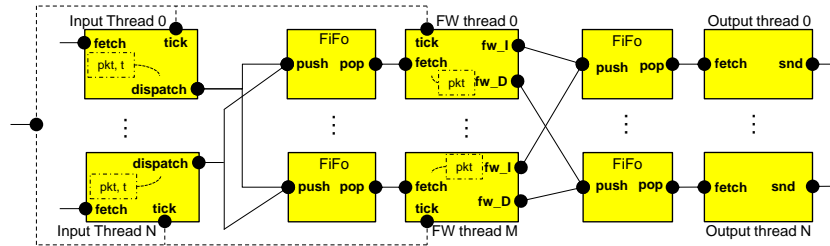

Fig. 4: A functional BIP model of the NDN-DPDK forwarder

4.2   Building the Performance Model

To build a performance model for our analysis, we consider the network topology in Fig. 5 which has a traffic generator client (consumer), a forwarder (NDN-DPDK) and a traffic generator server (producer), arranged linearly.

The green line shows the Interest packet path from the client to the producer through the forwarder and the red line indicates the Data path towards the
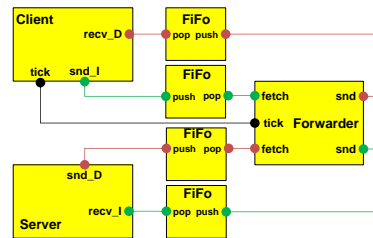

Fig. 5: Considered network topology

client. The structure of our model (Fig. 4) calls for four distribution functions to characterize performance: a) Interest dispatching latency in input threads. b) Data dispatching latency in input threads. c) Interest forwarding latency in forwarding threads. d) Data forwarding latency in forwarding threads. Notice that Nack packets are out of the scope of these experiments. We identified the following factors that can *potentially* affect the system's performance:

1. **Number of forwarding threads.** Having more forwarding threads distributes workload onto more CPU cores. The cores can compete for the shared L3 cache, and potentially increase forwarding latency of individual packets.
2. **Placement of forwarding threads onto Non Uniform Memory Access nodes (NUMA).** Input threads and their memory pools are always placed on the same NUMA node as the Ethernet adapter whereas the output threads and the forwarding threads can be moved across the two nodes. If a packet is dispatched to a forwarding thread on a different node, the forwarding latency is generally higher because memory access is crossing NUMA boundaries.
3. **Packet name length measured by the number of its components.** A longer name requires more iterations during table lookups, potentially increasing Interest forwarding latency.
4. **Data payload length.** Although the Data payloads are never copied, a higher payload length increases demand for memory bandwidth, thus potentially increasing latencies.
5. **Interest sending rate from the client.** Higher sending rate requires more memory bandwidth, thus potentially increasing latencies. It may also lead to packet loss if queues between input and forwarding threads overflow.
6. **Number of PIT entries.** Although the forwarder's PIT is a hash table that normally offers $O(1)$ lookup complexity, a large number of PIT entries inevitably leads to hash collisions, which could increase forwarding latency.
7. **Forwarding thread's queue capacity.** the queues are suspected to impact the overall throughput of the router through packet overflow and loss rates. However, it does not influence packets individual latencies.

After identifying the factors with potential influence on packet latency, we instrument the real forwarder to collect latency measurements. Then, perform statistical analysis to identify which factors are more significant. This narrows down the number of factors used and associated distribution functions.

Forwarder Instrumentation. Factors 1, 2, 3, 4, 5 and 7 can be controlled by adjusting the forwarder and traffic generator configuration, while factor 6 is a result of network traffic and is not in our control. To collect the measurement, we modified the forwarder to log packets latencies as well as the PIT size after each burst of packets. We minimized the extra work that input threads and forwarding threads have to perform to enable instrumentation, leaving the measurement collection to a separate logging thread or post-processing scripts. It is important to mention that this task does in fact introduce timing overhead. Therefore, the values obtained will have a bias (overestimate) that translates into additional latency but the trends observed remain valid.

We conducted the experiment on a Supermicro server equipped with two Intel E5-2680V2 processors, 512 GB DDR4 memory in two channels, and four Mellanox ConnectX-5 100 Gbit/s Ethernet adapters. The hardware resources are evenly divided into two NUMA nodes. To create the topology in Fig. 5, we connected two QSFP28 passive copper cables to connect the four Ethernet adapters and form two point-to-point links. All forwarders and traffic generator processes were allocated with separate hardware resources and could only communicate over Ethernet adapters.

In each experiment, the consumer transmitted either at sending intervals of one Interest per 700 ns or per 500 ns under 255 different name prefixes. There were 255 FIB entries registered in the NDN-DPDK forwarder at runtime (one for each name prefix used by the consumer), all of which pointed to the producer node. The producer would reply to every Interest with a Data packet of the same name. The forwarder's logging thread was configured to discard the first 67 108 864 samples (either latency trace or PIT size) during warm-up period, and then collect the next 16 777 216 samples and ignore the cool down session. Each experiment represents about 4 million Interest-Data exchanges. We repeated

Table 1: Factors used. NUMA mapping is described below.

Factors	forwarding threads	Name length	Payload length	Sending intervals
Values	{1, 2, 3, 4, 5, 6, 7, 8}	{3, 7, 13}	{0, 300, 600, 900, 1200}	{500 ns, 700 ns}

the experiment using different combinations of the factors in Table 1 and the following NUMA arrangements:

(P1) Client and server faces and forwarding threads are all on the same NUMA,
(P2) Client face and forwarding threads on one NUMA, server face on the other,
(P3) Client face on one NUMA, forwarding threads and server face on the other,
(P4) Client face and server face on one NUMA, forwarding threads on the other.

In P1, packet latency is expected to be the smallest because all processes are placed on the same NUMA therefore, no inter-socket communication and no overhead are introduced. In P4, both Interests and Data packets are crossing NUMA boundaries twice since the forwarding threads are pinned to one NUMA whereas the client and the server faces, connected to the Ethernet adapters, reside on another. This is suspected to increase packet latency tremendously as opposed to P1, P2 and P3. These suspicions predict that placement P1 is the best case scenario and placement P4 is obviously the worst. However, we aim at getting more insight and confidence through quantitative formal analysis. This will provide a recommendation as to which placement is better suited based on the remaining parameters combinations.

Model Fitting. Before calibrating our functional BIP model with multiple distinctive probability distributions representing each combination of the factors, we choose to reduce the number of used distributions by performing a sensitivity analysis. This analysis examines the impact of several factors on the response
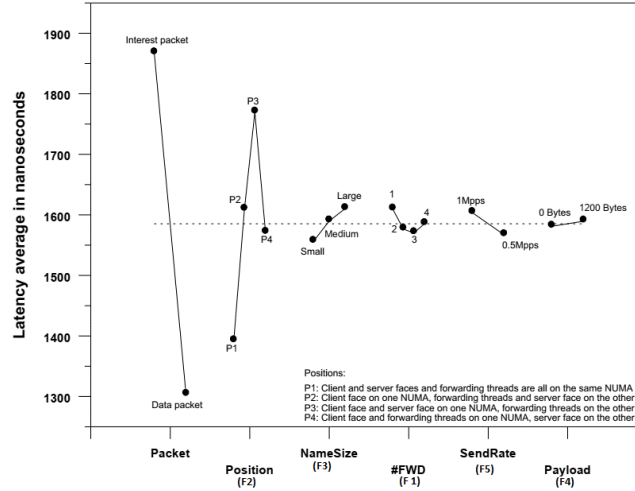
Fig. 6: Main Effects Plot for Interest and Data packets

(packet latency) and discovers the ones that are more important. In this paper, we use DataPlot [2] to produce the Main Effect Plot (Fig. 6) for factors 1 to 5.

The plot shows steeper line slopes for the packet type (packet type is not a factor. We intend to show how the NDN-DPDK forwarder processes both Interest and Data differently) as well as factors (1), (2), (3), and (5) which indicates a greater magnitude of the main effect on the latency. However, it shows almost a horizontal line for factor 4 inducing an insignificant impact on the latency. The latter is explained by the fact that the forwarder processes packet names (headers) only and doesn't read Data payloads. As for the PIT size (factor 6), it is expected to heavily increase packet latency when it is full. However, because this table's implementation is optimized for high performance and entries are continuously removed when Data packets arrive (PIT entries being satisfied), we confirmed through a correlation analysis that we can ignore this factor's impact.

Based on the analysis above, we build distribution functions for each of the factors that have greater impacts on packet latency in this study. These factors are: 1. (1) the number of forwarding threads, 2. (2) NUMA placement, 3. (3) packet name size (header), 4. (5) sending rate and, 5. (7) FIFO capacity (FIFO impacts the loss rates and not individual packet latency). We refer the reader to [10] to understand how we obtained the probability distributions for these factors.

Model Calibration. Calibration is a well defined model transformation that transforms functional components into stochastic timed ones [13]. In this section, we use the probability distributions obtained above to calibrate the functional BIP model of the NDN-DPDK forwarder shown in Fig. 4. Due to space limitations, we refer the reader to [10] where we describe the calibrated models of all the BIP components of the NDN-DPDK forwarder.

In the next section, we perform SMC on the calibrated model of the NDN-DPDK forwarder and explain the results.

5 Performance Analysis using SMC

5.1 Experimental Settings

We run the SMC tests using the probability estimation algorithm (PE) with a required confidence of $\alpha = 0.1$ and a precision of $\delta = 0.1$. Each test is configured with a different combination of values for the factors previously presented. And each execution of a test with a single set of parameters generates a single trace. The property evaluated with the SMC engine is: *Estimate the probability that all the issued Interests are satisfied, i.e. a Data is obtained in return for each Interest.* The SMC result is a probability estimation $\hat{p}$ which should be interpreted as being within the confidence interval $[\hat{p} - \delta, \hat{p} + \delta]$ with probability at least $(1 - \alpha)$. In the experiments below, the shown results corresponds to $\hat{p} = 1$.

5.2 Analyses Results

Queues Dimensioning. First, we explore the impact of sizing forwarding threads queues. Each forwarding thread has an input queue. Initially, we consider a model with a single forwarding thread and vary its queue capacity with 128, 1024 or 4096 (in packets). Then set the client's sending rate to: 10^5 packets per second (pps), 10^6 pps or 10^7 pps. The results are shown in Fig. 7a. The Y-axis represents the Interest satisfaction rate such that 100 % (resp. 0 %) indicates no loss (resp. 100 % loss) and the x axis represents the queue capacity under different sending rates.
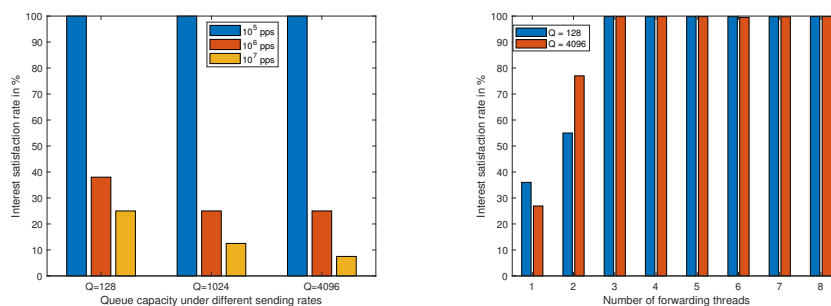
(a) One Forwarding thread with different sending rates.

(b) Many Forwarding threads with a sending rate set to 10^6 pps.

Fig. 7: Exploration results of the Forwarding threads queues sizes.

Fig. 7a indicates that at 10^5 pps (blue), the Interest satisfaction rate is 100%. This means that the forwarder (with one forwarding thread) is capable of handling all packets at this sending rate (10^5 pps of packet size 1500 bytes is equivalent to 1.2 Gbps), under any queue size. However, under a faster sender rate

(where a single forwarder shows signs of packet loss) we unexpectedly observed a better Interest satisfaction rate with a smaller queue (Q=128). After a thorough investigation of the real implementation, we found out that the queues don't have proper management in terms of insertion and eviction policies that would give priority to Data over Interest packets. In the absence of such policy, more Interests would be queued while Data packets would be dropped resulting in Interests not being satisfied, thus lower performance (Interest satisfaction rate). *It is thus advised for the final implementation of the NDN-DPDK forwarder, to use a queue capacity smaller than* 128 *packets when the forwarder has a single forwarding thread and packets are sent at a fast rate.*

Similarly, we explore whether this observation remains true with more forwarding threads. In order to do that, we run SMC again on eight different models each with a different number of forwarding threads (1 to 8) under a sending rate of 10^6 pps (1 Interest per 1 us) where a loss rate was observed in Fig. 7a. Then, we experimented with two queue capacities, namely 128 and 4096 packets. The results are reported in Fig. 7b. The x Axis represents the number of forwarding threads while the y axis depicts the Interest satisfaction rate.

We observe that the queue size matters mainly in the case of a model with one and two forwarding threads. In fact, for a two threads model, a bigger queue size is preferred to maximize the performance, unlike when a single thread is used. As for the other six models, both sizes achieve almost 100 % Interest satisfaction. This is due to the fact that three forwarding threads or more are capable of splitting the workload at 10^6 pps and can pull enough packets from each queue with a minimum loss rate of 0.02 % . *This result stresses that, to avoid being concerned about a proper queue size, more threads are needed for handling a faster sending rate with minimum Interest loss.*

NUMA placement, number of forwarding threads and packet name length. Another aspect to explore, is the impact of mapping the forwarding threads and/or NDN Faces to the two NUMA nodes (0, 1) under different sending rates and for multiple name lengths where Face 0 exchanges packets with the client and Face 1 with the server. To do that, we consider the four NUMA arrangements (P1), (P2), (P3) and (P4) in section 4 as well as the factors in Table 1 in the SMC analysis.

In Figs. 8 to 13, each row represents experiments with similar packet name lengths {small=3, medium=7, large=13} and a queue capacity of 4096. The right-hand column indicates results for a faster sending rate of $2 * 10^6$ pps (500 ns interval) while the left-hand one shows results for a slower sending rate of $1.42 * 10^6$ pps (700 ns interval). The six figures includes four curves where each corresponds to the four NUMA arrangement options: P1 to P4.

The six Figs. 8 to 13 show that Interest satisfaction rates scale up with the increase of forwarding threads then reach a saturation plateau where adding more threads can no longer improve the performances. Furthermore, with fewer forwarding threads, the loss rate is unavoidable and exceeds 80 %. This is because the sending rate is faster than the forwarding threads processing capabilities
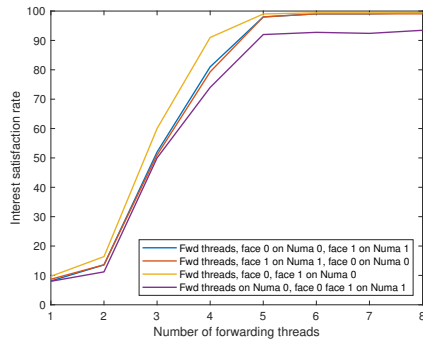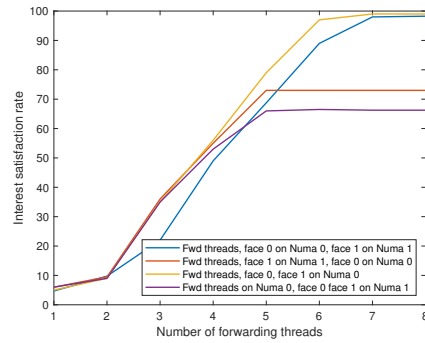
Fig. 8: small names, 700 ns
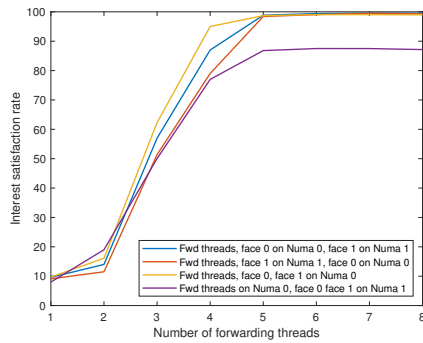

Fig. 9: small names, 500 ns
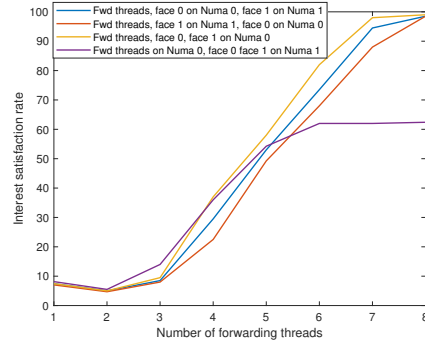

Fig. 10: medium names, 700 ns
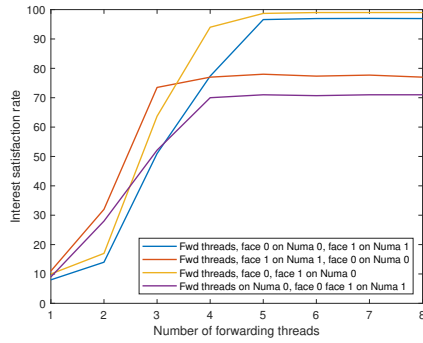

Fig. 11: medium names, 500 ns
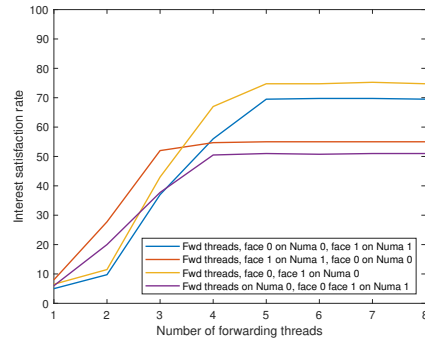

Fig. 12: large names, 700 ns

Fig. 13: large names, 500 ns

causing their FIFO queues to saturate and start dropping packets frequently. However, under a slower sending rate and packets with small, medium and large name lengths (3, 7, 13), Figs. 8, 10 and 12 show that a maximum satisfaction rate of over 90 % is achievable with only five forwarding threads. Whereas when the client is generating packets faster at 2 Mpps, a saturation plateau of over 90 % is reached at six threads or more for small and medium names (Figs. 9 and 11) and a plateau of slightly over 70 %, with five threads, for larger names (Fig. 13). Also, Figs. 8 and 10 demonstrate that placing all processes (threads and faces) on a single NUMA (placement P1) outperforms the other three options. This observation is explained by the absence of inter-socket communication thus less timing overhead added such as in the case of the purple plot where packets are crossing NUMA boundaries twice from Face 0 to the forwarding threads then through Face 1 and back (placement P4).

Figs. 9 and 11 show the impact of increasing the sending rate on packets with smaller names. In this case, it is preferred to also position all the processes on one NUMA such as the case of the yellow plot of the P1 series because NUMA boundary crossing usually downgrades the performance. In fact, the difference between no NUMA crossing and the double crossing (yellow and purple series respectively) is approximately 30 % loss rate with more than five threads. The second best option P2 which is placing the forwarding threads on the NUMA receiving Interest packets with Face 0 (NUMA hosting the Ethernet adapter that receives Interests from the Client). However, when the number of threads is not in the saturation zone and the threads get overworked and start to loose packets, it is recommended to opt for placement P3. *Based on these results, we recommend that for small to medium names, to use a maximum of eight threads but no less than five arranged as in placement P1 for optimum performances under a slower or a faster sending rate.*

With a larger name however, Fig. 12 depicts an unexpected behaviour when using three threads or less. In this case, placing the forwarding threads on the same NUMA as Face 1 (which is the Ethernet adapter connected to the server and receives Data packets), surpasses the other three options. Our explanation is that since forwarding threads take longer times to process incoming packets due to their longer name and timely lookup, particularly for Interests as they are searched by names inside the two tables (PCCT and FIB) rather than a token such as the case for Data packets. Placing the forwarding threads with the Data receiving Ethernet adapter connected to Face 1, has the potential to yield better results by quickly processing packets after a quick token search especially when the workload is bigger than the threads' processing capacity. When the sending rate is increased, the same results are observed in Fig. 13 for a similar name length but with a decrease in performance. *Thus, we recommend for larger names to use NUMA arrangement P3 only when the number of forwarding threads is less than three regardless of the sending rate (not advised due to high loss rate).*

6 Lessons learned and future work

This study shed light on a new networking technology called Named Data Networking (NDN) and its forwarding daemon. Ongoing NDN research includes the development of high-speed data plane forwarders that can operate at a hundred gigabits per second while using modern multi-processor server hardware and kernel bypass libraries. In this paper, we discussed the results of a performance evaluation effort we undertook to reach well-founded conclusions on how the NDN forwarder prototype developed by NIST (NDN-DPDK) behaves in a network in terms of achievable Interest satisfaction rate.

We conducted an extensive analysis under different factors such as the number of threads carrying tasks and function mapping to CPUs, using a model-based approach and statistical model checking. Given the wide array of design parameters involved, this effort contributes valuable insights into protocol operation and guides the choice of such parameters. The use of statistical model checking for performance analysis allowed us to discover potential sub-optimal operation and propose appropriate enhancement to the queue management solution. This has been taken into account in the ongoing NDN-DPDK forwarder implementation. Moreover, our extensive analysis provides a characterization of the achievable forwarding throughput for a given forwarder design and available hardware resources which would not have been possible to obtain, with such controllable accuracy, using traditional measurements and statistic methods. Furthermore, these results were communicated and shared with members of the NDN community in a conference throughout a poster interaction and gained attention from researchers who were interested in the methodology and its applications. In addition to that, the use of a BIP model refined at the right level of abstraction allows the generation of executable code that could be used instead of the real implementation.

It is important to note however, that our analysis depends largely on a stochastic model obtained using samples of data collected from the actual implementation of the forwarder which is suspected to have introduced timing overhead. Nevertheless, the trends observed throughout this study remain accurate and have provided valuable insight to the actual code. In the future, this analysis will be extended to answer the reverse question, namely **Given a desired throughput, what is the best hardware setup and the forwarder design to use?** Rather than the question **Given a hardware setup and a forwarder design, what is the maximum achievable throughput?** that we have investigated in this paper.

References

1. Brown, b. (2019). cisco, ucla & more launch named data networking consortium. [online] network world., `https://www.networkworld.com/article/2602109/ucla-cisco-more-join-forces-to-replace-tcpip.html`

2. Dataplot homepage, `https://www.itl.nist.gov/div898/software/dataplot/homepage.htm`
3. NFD Developer's Guide. Tech. rep., `http://named-data.net/techreports.html`
4. Basu, A., Bensalem, S., Bozga, M., Caillaud, B., Delahaye, B., Legay, A.: Statistical Abstraction and Model-Checking of Large Heterogeneous Systems. In: Forum for fundamental research on theory, FORTE'10. LNCS, vol. 6117, pp. 32–46. Springer
5. Basu, A., Bensalem, S., Bozga, M., Delahaye, B., Legay, A., Siffakis, E.: Verification of an AFDX infrastructure using simulations and probabilities. In: Runtime Verification, RV'10. LNCS, vol. 6418. Springer (2010)
6. Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in bip. In: Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods. pp. 3–12. SEFM'06, IEEE Computer Society, Washington, DC, USA (2006)
7. David, A., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B., Sedwards, S.: Statistical model checking for biological systems. Int. J. Softw. Tools Technol. Transf. **17**(3), 351–367 (Jun 2015)
8. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate Probabilistic Model Checking. In: International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI'04. pp. 73–84 (January 2004)
9. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking Named Content (2009), `https://named-data.net/wp-content/uploads/Jacob.pdf`
10. Khoussi, S., Nouri, A., Shi, J., Filliben, J., Benmohamed, L., Battou, A., Bensalem, S.: Performance evaluation of a NDN forwarder using statistical model checking. CoRR **abs/1905.01607** (2019), `http://arxiv.org/abs/1905.01607`
11. Mediouni, B.L., Nouri, A., Bozga, M., Dellabani, M., Legay, A., Bensalem, S.: $\mathcal{S}$BIP 2.0: Statistical model checking stochastic real-time systems. In: Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings. pp. 536–542 (2018)
12. Named data networking project. Tech. rep., USA (Oct 2010), `http://named-data.net/techreport/TR001ndn-proj.pdf`
13. Nouri, A.: Rigorous System-level Modeling and Performance Evaluation for Embedded System Design. Ph.D. thesis, Grenoble Alpes University, France (2015)
14. Nouri, A., Bensalem, S., Bozga, M., Delahaye, B., Jegourel, C., Legay, A.: Statistical model checking QoS properties of systems with SBIP. Int. J. Softw. Tools Technol. Transf. (STTT) **17**(2), 171–185 (April 2015)
15. Nouri, A., Bozga, M., Molnos, A., Legay, A., Bensalem, S.: *ASTROLABE*: A rigorous approach for system-level performance modeling and analysis. ACM Trans. Embedded Comput. Syst. **15**(2), 31:1–31:26 (2016)
16. Nouri, A., Mediouni, B.L., Bozga, M., Combaz, J., Bensalem, S., Legay, A.: Performance evaluation of stochastic real-time systems with the sbip framework. International Journal of Critical Computer-Based Systems **8**(3-4), 340–370 (2018)
17. Xylomenos, G., Ververidis, C.N., Siris, V.A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K.V., Polyzos, G.C.: A survey of information-centric networking research. IEEE Communications Surveys Tutorials **16**(2), 1024–1049 (2014)
18. Younes, H.L.S.: Verification and Planning for Stochastic Processes with Asynchronous Events. Ph.D. thesis, Carnegie Mellon (2005)
19. Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., claffy, k., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B.: Named data networking. SIGCOMM Comput. Commun. Rev. **44**(3), 66–73 (Jul 2014)