Fetching Popular Data from the Nearest Replica in NDN

Jianxun Cao*, Dan Pei*, Xiaoping Zhang*, Beichuan Zhang†, Youjian Zhao*
*Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University
Email: caojx10@mails.tsinghua.edu.cn, {peidan, zhxp, zhaoyoujian}@tsinghua.edu.cn
†Department of Computer Science, University of Arizona
Email: bzhang@cs.arizona.edu

***Abstract*—As a novel Internet architecture, Named Data Networking (NDN) shifts the communication model from address-centric to content-centric. An NDN router caches the data in its content store, greatly reducing network traffic. NDN adopts the *hierarchical naming schema*, which allows the name aggregation and enables high scalability. However, in a richly connected topology, the nearest data replica are often not on the path dictated by NDN's tree-like data fetching model. This might result in a lower data delivery efficiency compared with the *flat self-certifying naming schema* in other Information-Centric Networking (ICN) architectures.**

To address the low efficiency problem, we propose a CDN-like enhancement to the NDN design, called Fetching the Nearest Replica (FNR). In FNR, when a consumer sends an interest for a popular data, the data is fetched from the nearest replica in the network, regardless of whether it is on the best path from the producer to the consumer. We present the design details and theoretical overhead analysis for FNR. Our evaluation results using ndnSIM simulator show that on average FNR reduces the total (inter-domain, intra-domain) traffic by 25.6% (53.0%, 18.2%) on average, compared to the default NDN approach. In addition, the average latency is reduced by 37% and the average cost is reduced by 51.4%. To the best of our knowledge, this paper is the first NDN enhancement in the literature to support nearest replica fetching in NDN.

I. Introduction

Information-Centric Networking (ICN) is a new Internet architecture that focuses on *what* instead of *where*. The proposals of ICN includes decoupling names from locations, binding names to content, and providing a publish/subscribe model to discover and retrieve the content by name. In-network storage is introduced in ICN to help reduce the network traffic significantly, and data copies which are called **replicas** are dynamically created, cached and deleted in in-network storage. Among several proposed ICN architectures, different naming schemes result in the different data fetching models and routing principles.

Some ICN proposals, such as DONA [1], NetInf/SAIL [2], and MobilityFirst [3], adopt **flat self-certifying naming schema**. To publish content, Distributed Hash Table (DHT) is adopted to announce the content availability. To retrieve a content object, a name resolution and mapping process is proposed to resolve the mapping between the flat names to locations. In Flat self-certifying naming schema, the network can locate the nearest data replica for every data request. However, it faces the great **challenge of scalability** because the flat names of contents cannot be aggregated. With the rapid growth of content objects, a conservative estimate is that an ICN resolution and mapping system should be able to handle at least 10^{12} objects [4].

In contrast, NDN [5], [6] employs a URL-like **hierarchical naming schema** which enables prefix-based aggregation like in IP. Prefix-based aggregation, on-path storage, and tree-like routing model in NDN in general offer a better scalability than other ICN proposals. However, as the Internet topology becomes more and more richly connected in these years [8], the nearest data replica might not be on the best-path dictated by NDN's tree-like fetching data model. For example, in the forwarding plane, the router must forward the Interest packet along the default path according to the FIB, even if the nearest replicas might not be on the default forwarding path, resulting in the **low data delivery efficiency** problem in the current NDN design.

To address the low efficiency problem, in this paper we aim to design an NDN enhancement such that when a consumer sends an Interest packet for a popular data, the data is fetched from the nearest replica in the network, regardless of whether it is on the best path from the producer to the consumer. There can be two strawman approaches. First, in the *multi-path approach*, multiple copies of the same data are retrieved to the consumers. This approach has prohibitive overhead and defeats the NDN's initial purpose of reducing the traffic overhead. Second, each router announces the replica information in its Content Stores (CS) to other routers, so that the nearest replica can be known for each request. However, the announcement overhead is also prohibitive if all CS of all routers are sent. Apparently neither of the two strawman approaches works.

Our approach in this paper takes advantage of the fact that the distribution of data popularity in the current Internet is close to a Zipf distribution [9]. Therefore, instead of announcing the entire CS in the second strawman approach, in FNR a router just announces the information about the top-N data ranked by popularity in NDN routers' Content Store (CS). As such, according to the request popularity, we divide a router's CS into two parts: **Top-N Subset** and **Heavy-Tailed Subset**. With Zipf law, the Top-N Subset (with a very small subset size N) can already satisfy most data requests, while

[1]Xiaoping Zhang is the corresponding author.

the Heavy-Tailed Subset (with a huge subset size) just satisfies a small number of data requests. The Top-N subsets of routers are announced to *Tracker Servers*, and a consumer can consult its Tracker Server to locate the nearest replica for a popular data request, and then redirect the data request to the nearest replica.

Based on our above idea, we propose an NDN enhancement design, Fetching the Nearest Replica (FNR). The design of FNR shares the same spirits as the CDN for IP networks: one level of redirection to help locate the nearest replica for the requested data in the network. The top-N subsets maintained by Tracker Servers and routers are similar to the (dynamically updated) DNS system in the CDN network. The consumer looks up the tracker system to find the nearest replica in FNR, just like a host uses DNS to map the destination host name to a best IP address among multiple ones that serve the same content in the IP CDN system.

The contributions of this paper are summarized as follows.

1) To address current NDN design's problem of low data delivery efficiency, in this paper we propose FNR, a CDN-like enhancement to the NDN design. In FNR, when a consumer sends an Interest packet for a popular data, the data is fetched from the nearest replica in the network, regardless of whether it is on the best path from the producer to the consumer. To the best of our knowledge, this paper is the first NDN enhancement in the literature to support nearest replica fetching in NDN.
2) To make FNR scalable, practical, and light-weight, we addressed two major challenges. First, to reduce the overhead of replica announcement, we take advantage of Zipf law so that a router only announces the information of Top-N data in its CS and uses Bloom filter to further minimize the announcement traffic. Second, to make FNR scalable and practical, we employ an effective but conceptually simple design that in spirit is similar to CDN in IP network, and use one level of redirection to locate and retrieve the nearest replica for popular data.
3) Our evaluation results using ndnSIM simulator [11] show that our FNR enhancement to NDN greatly improves the performance, compared to the current NDN design. FNR on average reduces the total traffic by 25.6% on average, the inter-domain traffic by 52.0%, and the intra-domain traffic by 18.2%, compared to the current NDN design. Furthermore, the average latency is reduced by 37.0% and the average cost is reduced by 51.4%.

The remainder of the paper is organized as follows. The overall design goal of FNR and details of design are respectively presented in Section II and Section III. We conduct theoretical analysis for FNR overhead in Section IV. In Section V, we evaluate FNR using ndnSIM 2.0 simulator and analyze the results. Section VI briefly reviews related work. Finally, Section VII concludes the paper.

II. Design goals and core idea

Our proposed Fetching the Nearest Replica (FNR) in NDN consists of three steps. **Pre-lookup**: before sending out the Interest packet I_1 for the data D_1, the consumer needs to check whether there exists any replica of D_1. **Locating**: if the consumer ensures the existence of the replica, then the consumer will ask for the location of that replica. **Redirection**: with the location of the replica, the consumer will modify the destination of I_1 in the Interest packet and send it out.

A. Design goals

The functionality of FNR boils down to one major challenge: all the consumers and routers should synchronize the up-to-date information of replicas and rapidly return responses (such as synchronizing and locating). A more precise breakdown of FNR goals is as follows.

1) **High performance.** The design must ensure a high performance to locate and fetch the nearest replica, which impacts the inter-domain/intra-domain traffic and QoS of consumers.
2) **Low overhead.** The design must ensure a low overhead, including the extra storage overhead and transmission overhead.
3) **Practicality.** The design must ensure the practicality based on the NDN networking protocol, which means that the design should not be too complex to modify the fundamental design of NDN.
4) **Security.** The design must be robust to malicious consumers attempting to announce or hijack replica records.

B. Challenges

Based on the above design goals, there are several challenges as follows.

Challenge: How to deal with a large number of replicas in the router's CS before their announcements?

Solution: Given a huge number of contents in CS and the popularity of data requests due to Zipf distribution, we divide the contents of CS into two subsets: the **Top-N Subset** to be announced, and **Heavy-tailed Subset** not to be announced. *Our strategy of replica announcement is just to announce the replicas in Top-N Subset instead of all the replicas.* The amount of replicas in Top-N Subset are not so many, but can satisfy the most requests. In addition, because that popular contents are requested so frequently that they will exist in the cache for a relatively long time, replicas in Top-N Subset are relatively stable so that they are suitable for announcement.

Challenge: How to collect and synchronize the information of replicas?

Solution: In our design, we adopt the centralized model by deploying a **Track Server (TS)** in each ISP to collect and synchronize the information of replicas. A TS collects all the information of replicas to maintain a **Replica List** and sends it to each consumer in its domain. Once the replicas in Top-N Subset of any router change, the router will upload the latest information of replicas to its TS. Likewise, once the Replica List updates, TS will notify all the consumers to synchronize the local Replica List with the TS.

Challenge: How to locate the nearest replica?

Solution: In an ISP, a TS stores the information of internal network topology, including the mapping relation between replica and router, and network metrics. To locate the nearest replica, the consumer sends the name of the requested Interest packet to the TS, and the TS will return the location of the nearest replica according to its information of internal network topology.

Challenge: How to guarantee that fetching data from the replica performs better than fetching data from the producer directly?

Solution: **Local Hashing Table** (LHT) is proposed to adaptively make the decision between fetching data from the replica and fetching data from the producer directly. LHT records the Round-Trip Time (RTT) of the two methods of fetching data and updates the RTT dynamically.

Challenge: How to minimize the overhead?

Solution: The overhead, including storage overhead and transmission overhead, is mainly caused by the Replica List. To minimize the overhead, we adopt **Bloom Filter** [12], a simple space-efficient randomized data structure for representing a set in order to support membership queries, to represent all the replicas in each Top-N Subset so that the Replica List becomes a list of Bloom Filter, which causes little overhead during either storage or transmission.

C. Core idea

Our core design idea is that each router announces the popular data in routers' Top-N subset to all the network, and each consumer can fetch popular data directly from the CS of corresponding routers with redirection. As shown in Fig. 1, the core idea consists of three aspects:

1) **TS**. Deploy a specified Tracker Server (TS) in each ISP to collect all the content information from all the routers' Top-N Subsets in the network as well as the network status.
2) **Consumer**. A consumer synchronizes the local Replica List with TS's Replica List. Before a consumer sends out an Interest packet to fetch the data, the consumer will check the existence of data's replica by looking it up in the local Replica List. If so, the consumer will ask the TS for the location of the router whose Top-N Subset stores the data. Then, TS will tell the consumer about the most suitable router, and the consumer will try to fetch the data from the router's Top-N Subset.
3) **Router**. A router needs to timely generate the Top-N Subset from CS. When a router receives an Interest packet for the content in Top-N Subset, the router will return the corresponding data.

This process is similar to the one in IP CDN lookup. A TS resembles the DNS sytem and a router's Top-N Subset is like the CDN cache. The similar spirit is that DNS/TS tells the host/consumer where to request the nearest replica in CDN/Top-N Subset with address/name redirection.

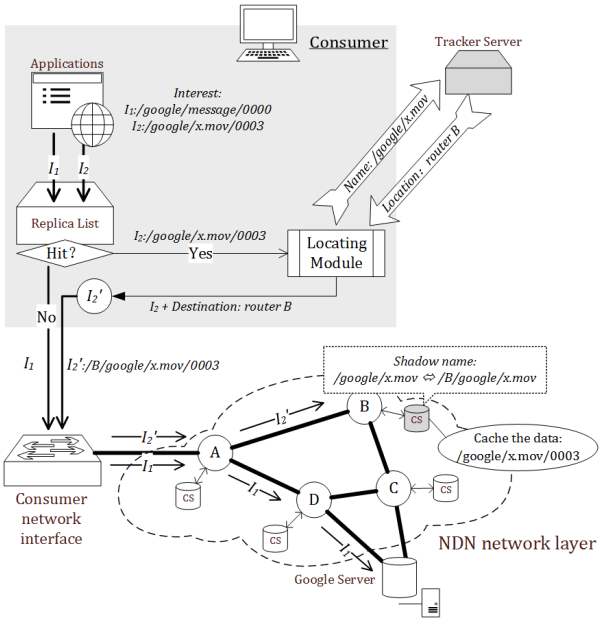


Fig. 1. The design.

D. Example

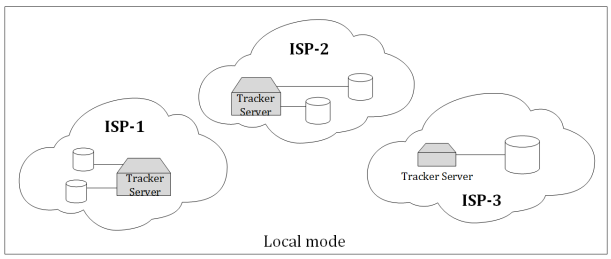
Here is an example to illustrate our design. In Fig. 1, I_1 and I_2 are the two Interest packets to fetch data. The name of I_1 is "/google/message/0000" and the name of I_2 is "/google/x.mov/0003". The corresponding Data packet of I_2 has a copy in the Top-N Subset of the router B. Before the consumer sends the two Interest packets to the network, it firstly looks up the Interest names in the local Replica List, which consists of all the Bloom Filters of the Top-N Subsets of routers' Content Store. There are different decisions for I_1 and I_2, depending on whether the look-up in Replica List hits or not.

Interest I_1: As soon as the look-up of I_1 in Replica List fails, there is no additional step before I_1 is sent out and the consumer just sends out I_1 to the network as usual.

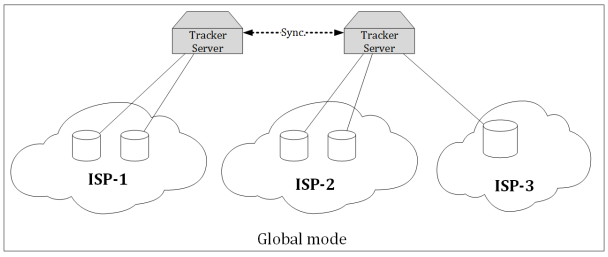
Interest I_2: When the look-up of I_2 in Replica List succeeds, before I_2 will be sent out, the processing of I_2 contains the following steps:

- In locating module, the name of I_2 ("/google/x.mov") will be sent to TS to locate the copies of the corresponding Data packet.
- Combining the network status and parameters, TS will return the current best destination (router B).
- A new Interest packet $I_2^{'}$ will be produced by changing the original name of I_2 ("/google/x.mov") to a shadow name ("/B/google/x.mov").

Then, the new Interest packet $I_2^{'}$ will be sent out to the network instead of I_2. According to the routing table, $I_2^{'}$ with the prefix "/B" will be forwarded to the router B. When the router B receives the Interest packet $I_2^{'}$, it firstly looks up the name of $I_2^{'}$ in a *mapping table* (implemented by pointers) to map the shadow name ("/B/google/x.mov") to the original name ("/google/x.mov"). Then, with the look-up of the original



(a) Local mode (Adopted)



(b) Global mode

Fig. 2. The comparison of deployment of TS between local mode and global mode.

name in its Content Store, the router B will return the Data with the shadow name ("/B/google/x.mov") to the consumer.

III. Design details

In this section, we will introduce our design in detail, which is divided into three parts: ***Tracker Server (TS)***, ***process in consumer*** and ***process in router***. In this paper, *domain* represents *ISP*.

A. Tracker Server (TS)

Tracker Server (TS) is proposed as a new network infrastructure component to manage information of replicas in network. In the view of a TS, there are several main challenges and objectives:

1) Deployment of TSs: There are two modes to deploy TSs: *local mode* and *global mode*, and the most important difference between the two modes is the scope of the routers and consumers served by TS. Local mode means that each ISP deploys just one TS, which just serves all the routers and consumers in this ISP. Correspondingly, with another deployment mode called global mode, there are some TSs that serve all the routers and consumers in the whole network. Each TS maintains global information and synchronizes the information with all the other TSs. Fig. 2 shows the comparison between local mode and global mode.

We adopt local mode by making a trade-off between the overhead and the performance. First, with local mode, because of the intra-domain lookup, the space complexity of Replica List and the time complexity of looking up the nearest replica will be greatly reduced. Second, fetching data from the local nearest replica (in ISP) performs equally well as fetching data from the global nearest replica (in the whole network) in most cases.

2) Maintain a network topology $G = (V, E)$, where V is a set of nodes and E is a set of links: To help consumers find the best replica location, TS needs to collect network status information for implementing the traditional traffic engineering objective: **to minimize the maximum link utilization (MLU)** [10].

$$\min_{\forall k: t^k \in T^k} \max_{e \in E} \left(b_e + \Sigma_k \Sigma_i \Sigma_{j \neq i} t_{ij}^k I_e(i,j) \right) \tag{1}$$

In the above equation, b_e denotes the amount of background traffic on link $e \in E$. We use $I_e(i,j)$ to present the indicator of edge e being on the route from router i to j in the topology G. T^k presents the set of acceptable traffic demand according to the requirements and properties of application session k, and correspondingly, t_e^k presents the amount of traffic on link $e \in E$ [10].

3) Merge, update and synchronize information: A TS provides the services (such as merging, updating and synchronizing) of all the popular content names and network information.

A TS maintains a Replica List from merging all the content names of each router's Top-N Subset. The Replica List consists of all the Bloom Filters of routers' Top-N subsets as shown in Fig. 3 and tells consumers which data is popular enough to be found in some routers' Top-N Subset.

Router ID	Bloom Filter											...				Update Time
0000	0	1	0	0	1	0	0	1	1	1	0	...	0	1	1	2016-01-01 12:00:00:000
0001	1	1	0	1	0	1	1	0	0	1	1	...	0	0	1	2016-01-01 12:00:07:600
0002	0	1	1	1	1	0	0	1	1	0	0	...	1	1	0	2016-01-01 12:04:04:500
.....	.....															.....

Fig. 3. An example of Replica List.

The information in a TS should be updated timely. Here are two cases. First, when any router needs to change its Bloom Filter because of the update of Top-N Subset, the router will notify TS to fetch the latest Bloom Filter. Then, TS will update its Replica List and synchronize with all the consumers. Second, for each router's Bloom Filter, TS will update the Bloom Filter from the router if the router doesn't update its Bloom Filter for a long time (*e.g.* one day).

4) Verify the identity of routers: To avoid fake announcements of attackers, each router must be authenticated as a real device in the network with the unique ID. A Router ID List of authenticated routers should be stored in TS. When TS receives any Interest packet which is claimed from a router, TS must match the ID of the router in the Router ID List and check the signature of the Interest packet to verify the identity of the router.

B. Consumer

For an Interest packet, before sending out the Interest packet, there are two more steps:

1) Check whether the Interest packet name is in the local Replica List. If not, the user will send out the Interest packet without any changes. Otherwise, the user will deal with the Interest packet in the *Locating Module* as follows.

2) Using Locating Module and LHT to get to the nearest replica location. LHT is introduced to cache the calculated information by TS to avoid repeated look-ups in TS as well as recording the states of outgoing Interest packets for replicas. Unlike the Replica List which records the existence of all the replicas, LHT just records the information of Interest packets which have requested for replicas recently. The time-life of an entry in LHT is limited. For a valid LHT entry, it serves three purposes:

- For the Subsequent Interest name, it avoids repeatedly locating the data.
- Provides the mapping from the name and the shadow name.
- Provides the decision about whether to fetch data from the replica location or to fetch the data from the producer.

The entry of LHT contains the following fields:

- **Timer.** If the timer expired, this entry is invalid.
- **Name.** The name of the Interest packet.
- **Shadow Name.** The name of the corresponding R-Interest (mentioned later) packet.
- **PRTT.** The dynamically updated RTT for fetching the data from the producer.
- **RLRTT.** The dynamically updated RTT for fetching the nearest replica.
- **Decision.** Dynamically deciding about fetching data from the producer or replica location according to their RTT value.

Locating Module will first check whether the Interest packet is the *New Interest packet* or the *Subsequent Interest packet* by looking it up in the LHT. If the look-up fails, which means the Interest packet is a New Interest packet, then the following steps will be performed:

- Send the Interest name to TS to get the nearest replica location which stores the corresponding backup data.
- Create a new entry in LHT.
- Add the nearest replica location as a new prefix to the Interest name as a new Interest packet. We define it as **R-Interest** packet and its name is called *shadow name*. Then, send out both the original Interest packet and R-Interest packet to the network to test the PRTT and RLRTT.

If the look-up succeeds, which means the Interest packet is the Subsequent Interest packet, then the timer of the entry for the Subsequent Interest packet will first be checked. If the timer is expired, this Interest packet will be still treated as a New Interest. Else, the Subsequent Interest will be queried in the LHT to make a decision about the type of outgoing Interest packet (R-Interest or original Interest) and be sent out.

Consumers should periodically obtain the latest version of Replica from TS to synchronize the Replica List with TS. Once the Replica List in TS changes, the consumer will download the changed part to update the local Replica List.

TABLE I
SOME DENOTATIONS

R	The number of routers in an ISP
R_e	The number of edge routers in an ISP
N	The number of entries in Top-N Subset
N_{cs}	The number of entries in CS
N_{lht}	The number of entries in LHT
N_{ts}	The number of TSs in an ISP
m	The length of array in Bloom Filter
f	False positive rate of Bloom Filter
k	The number of hashing fuctions

C. Router

A CS of a network router stores the most popular data. In our design, according to the popularity, a CS will be divided into two parts: announced **Top-N Subset**, and not announced **Heavy-Tailed Subset**. The two different subsets are just distinguished with different marks, *T* and *H*, in each entry of the CS. To ensure the announced contents exist in the CS, once an entry is marked with *T*, it cannot be replaced by any content until this mark is removed.

There is a control module to help the CS deal with the incoming R-Interest with two steps:

1) Maintain a mapping table to support shadow name lookup. The Interest packet needs to be checked whether it is an R-Interest packet. If so, before the lookup in CS, the shadow name of the R-Interest packet should be mapped to the original name. Thus, the CS needs to maintain a mapping table to deal with the name mapping process. This mapping table is just implemented with pointers.

2) Update the contents in Top-N Subset and upload its Bloom Filter to TS periodically. The contents in Top-N Subset are popular at present, while the popularity is reduced after a while. Thus, the contents in Top-N Subset should be updated with some content replacement policy, periodically. Once the popularity ranking of contents is changed greatly, or the contents in Top-N Subset are not updated for a long time, the Top-N Subset should also be changed. Then, after the contents in Top-N Subset are updated, the latest Bloom Filter of the Top-N Subset will be uploaded to the corresponding TS to make sure of the validity.

IV. ANALYSIS OF OVERHEAD

Some denotations are shown in Table I. The overhead of FNR consists of two parts: storage overhead SOV and transmission overhead TOV.

A. Storage overhead

The total storage overhead SOV is mainly composed of two parts: overhead in CS (SOV_r) and overhead in consumer (SOV_c).

For each router's CS, firstly, every CS entry must add a bit to record the mark of subset type, thus, the extra storage overhead is $N_{cs} \cdot 1$ bit $= N_{cs}$ bit. Secondly, each router's CS also needs to store an array of N pointers to all the contents in Top-N Subset. Assume that each pointer occupies 32 bits, then the extra storage overhead of N pointers equals $N \cdot 32$ bit $= 32N$ bit. Thirdly, each router has to store a Bloom Filter of Top-N Subset, which needs m bits. Thus, the total extra storage overhead of each router SOV_r equals:

$$SOV_r = N_{cs} + 32N + m \quad bit \tag{2}$$

For each consumer, the extra storage includes the storage of Replica List and LHT. On the one hand, as shown in Fig. 3, Replica List at least includes three fields: Route Number, Bloom Filter and Update Time. Assume that the bits of these three fields respectively require 16 bit, m bit and 64 bit, then the total extra storage overhead of Replica List equals $(16 + m + 64) \cdot R$ bit $\approx m \cdot R$ bit. On the other hand, as mentioned in Section III B, in LHT, each entry needs 6 fields to store the information, which approximately occupies 8 Kbit. Thus, the total storage space of LHT equals $8000N_{lht}$ bit. In conclusion, the total extra storage for each consumer SOV_c equals:

$$SOV_c \approx mR + 8000N_{lht} \quad bit \tag{3}$$

B. Transmission overhead

The total transmission overhead TOV consists of two parts: the transmission overhead between TS and routers (TOV_{tr}) and the transmission overhead between TS and consumers (TOV_{tc}).

To estimate the data transmission overhead between TS and routers, we assume that for the router r_i, the average update cycle of Top-N Subset is T_i. When a router needs to update its Bloom Filter array of Top-N Subset, it will upload the Bloom Filter array to TS. Thus, the transmission overhead of the router r_i equals m/T_i bit/s. Thus, the total transmission overhead between TS and routers TOV_{tr} equals:

$$TOV_{tr} = \sum_i \frac{m}{T_i} \quad bit/s \tag{4}$$

Once a router updates its Top-N Subset, every consumer needs to update the Replica List. However, because of the aggregation of the same data, the overhead of updating Replica List by all the consumers which are connected to the same edge router equals the overhead by just one consumer. Thus, the total transmission overhead between TS and consumers TOV_{tr} equals:

$$TOV_{tc} \approx R_e \cdot TOV_{tr} = R_e \sum_i \frac{m}{T_i} \quad bit/s \tag{5}$$

V. Performance evaluation

In this section, we evaluate the performance of FNR with modified ndnSIM 2.0 simulator [11]. The new version of ndnSIM integrates ndn-cxx library (NDN C++ library with eXperimental eXtensions) and the NDN Forwarding Daemon (NFD) to enable experiments with real code in a simulation environment.

A. Experimental setup

Topology: We set one domain with consumers to request the data provided by the producers outside the domain. To make our simulation as close as possible to reality, we use the topology of Tsinghua University Campus Network (THUNet) for intra-domain topology, which contains 20 routers, 49 consumers and 82 links with link metrics.

Dataset: Lots of studies investigate the request distribution and observe that the request distribution follows the heavy-tailed or Zipf-like distribution quite well [14]. Based on the above observation, the requesting dataset for consumers in our evaluation contains more than 1,000,000 objects whose request frequency distribution follows the Zipf law [9]:

$$f_i \sim \frac{1}{r_i^{\alpha}} \tag{6}$$

In Equation 6, f_i denotes the frequency request of the i^{th} popular object and r_i denotes the rank of f_i. α is the Zipf parameter which in our experiments equals 1.04 (The study [9] shows that the data request distribution in Asia follows the Zipf law with $\alpha = 1.04$).

Parameters: There are some reasonable assumptions of parameters:

- Consumers are set to send Interest packets to the network under the random frequency ranging from 100 times per second to 1500 times per second.
- The capacity of router's cache in the network is set as 10000 objects and the N of Top-N Subset of Cache is set as 1000, which means that we choose 10% of cache size to provide data replicas. We vary N at the end of this section.
- To simplify the complexity of implementation of minimizing the MLU, we use the following equation to measure the virtual distance between two nodes:

$$Vd_{ij} = \frac{cost_{ij} \cdot delay_{ij}}{bandwith_{ij}} \tag{7}$$

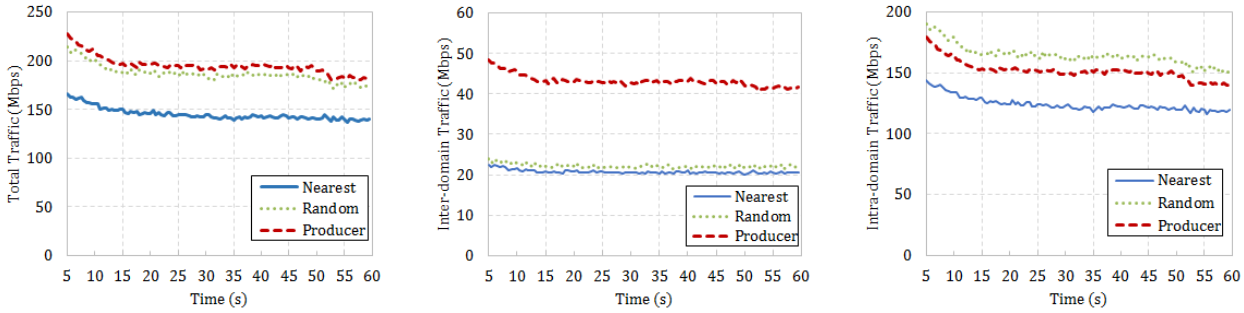
Bloom Filter: In our experiments, the parameters of Bloom Filter are set as follow:

- Elements: $n = N = 1000$
- Bloom Filter bits: $m = 8n = 800bit$
- Hash functions: $k = 6$

According to the above parameters, the false positive rate $f \approx 0.02$.

Comparison: We compare all the evaluation metrics in 60s among three data fetching models:

- *Nearest*. Fetching data from the nearest replica.



(a) Total Traffic (b) Inter-domain Traffic (c) Intra-domain Traffic

Fig. 4. The comparison of intro-domain/inter-domain/total traffic among *Nearest*, *Random* and *Producer*.

- *Random*. Fetching data from random replica.
- *Producer*. Fetching data from the producer directly.

Evaluation Metrics: We choose the following evaluation metrics in our experiments:

- *Total/inter-domain/intra-domain traffic*. Under the same request model of consumers, we compare the total/inter-domain/intra-domain/ traffic in 60s among three fetching data models to illustrate the best trade-off between the performance and the overhead of our design.
- *QoS(latency/cost)*. We evaluate QoS, which directly reflects the performance of our design, with two parts: *latency* (the average latency of requests which are hit in the cache) and *cost* (the average cost of all the requests).
- *Influence of N (size of Top-N subset)*. We simulate the influence of varying subset sizes by observing inter-domain traffic. As a result of this result, we can approximatively choose the most suitable value of subset size.

B. Total/inter-domain/intra-domain traffic

1) Total Traffic: Firstly, we survey the total traffic among the three data fetching models. From the Fig. 4(a), we notice that, with the *Producer*, the average of the total traffic equals 195.2Mbps. And with the *Random*, the average of the total traffic is still as high as 186.7Mbps, which has just declined by 4.3% compared with the *Producer*. However, with the *Nearest*, the average of the total traffic is down to 145.1Mbps, which is greatly reduced by 25.6% in comparison with the *Producer*.

2) Inter-domain Traffic: Secondly, we focus on the inter-domain traffic among the three data fetching models. As shown in Fig. 4(b), with the *Producer*, the average inter-domain traffic shows as high as 43.2Mbps, while with the *Nearest* the average inter-domain traffic drops to 20.7Mbps, which is reduced by 52.0% in comparison with the *Producer*. However, we find that with the *Random* the inter-domain traffic average is just 22.2Mbps, which has declined by 48.5% in comparison with the *Producer* and is close to the value of the *Nearest*.

From the figure, we can conclude that, fetching replica helps the network reduce the inter-domain traffic significantly, no matter whether the replica is the nearest or not.

3) Intra-domain Traffic: Now we investigate the intra-domain traffic among the three data fetching models. From the Fig. 4(c), we notice that, when fetching the data from the producer, the value of the intra-domain traffic fluctuates from the minimum 139.5Mbps to the maximum 179.2Mbps, the average of which is 152.0Mbps. And with *Random*, the average intra-domain is as high as 164.4Mbps, which has risen by 8.2% as compared to the fetching of the data from the producer. But with *Nearest*, the average intra-domain traffic just equals 124.3Mbps, which declined by 18.2% in comparison with fetching the data from the producer as well as declining by 24.4% in comparison with *Random*.

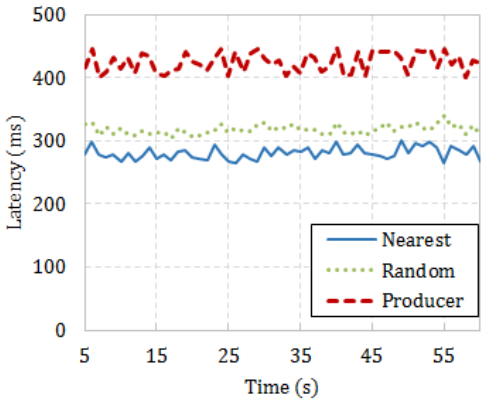
This is because that, although both the *Nearest* and the *Random* transform the part of inter-domain traffic to intra-domain traffic, the *Nearest* optimizes the choice of replicas to the lowest value by choosing the nearest replica, while the *Random* is not responsible for the choice, which may introduce higher traffic. Thus, compared to the *Random*, the significant advantage of *Nearest* is that, it reduces the intra-domain traffic greatly to fetch the nearest replica, instead of a random replica.

C. QoS

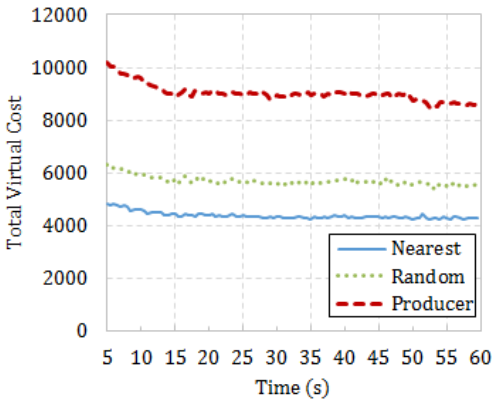
To evaluate QoS, we compare the average rate of latency and cost among the three data fetching models.

1) Latency: As shown in Fig. 5(a), first, with the *Producer*, the average latency of all the data requests per second fluctuates from about 400ms to about 450ms, the average of which is 424.8ms. Second, with the *Nearest*, this value fluctuates from about 270ms to about 300ms, the average of which is 280.5ms. Third, with the *Random*, this value fluctuates from about 300ms to about 330ms, the average of which equals 317.8ms. In other words, with the *Nearest* and the *Random*, the average percentage improvement in latency is 33.9% and 25.1% . We find that, with the *Nearest* or the *Random*, the average latency for consumers is reduced greatly, which means that consumers can obviously feel the improvement of fetching popular data with the *Nearest* or the *Random*.

2) Cost: We report another important parameter, cost, in terms of the total cost of the whole round trip during data transmission. Fig. 5(b) shows the ratio of cost among the three data fetching models. We observe that the cost with *Nearest*



(a) Average latency



(b) Average Cost

Fig. 5. The comparison of QoS (Latency and Cost) among FNR, FRR and fetching data from the producer.

is just 48.6 percent of the cost with fetching data from the producer on average, and the cost with *Random* is just 69.5 percent on average of the cost with fetching data from the producer on average.

Compared with the little improvement of latency, the great improvement of cost depends on the significant decline of inter-domain traffic, which brings considerably higher cost than intra-domain traffic. Much lower cost will enhance the User Experience (UE) and may help decrease the network charges for consumers to save money.

D. Influence of N

To check the influence of the different choices of Top-N Subset size N, we keep all the network parameters unchanged except for the size N which varies from 30 objects to 5000 objects, namely from 0.3% to 50% of cache size. Fig. 6 shows the average inter-domain traffic ratio and intra-domain traffic ratio between *Nearest* and *Producer* from 5s to 60s. There are two main observations. First, the larger N is, the lower the average inter-domain/intra-domain traffic ratio performs. Second, with the increasing N, the average inter-domain/intra-domain traffic ratio tends to approximatively decline at a exponential rate. When the size N achieves a certain value (about 5% of cache size in Fig. 6), the performance has hardly improved with the increasing of N and the average inter-domain traffic ratio reached a stable level. Thus, it's not necessary to increase the size of Top-N Subset N blindly. It is the best trade-off between performance and overhead that the ratio between the subset size and cache size remains between 5% to 10%.

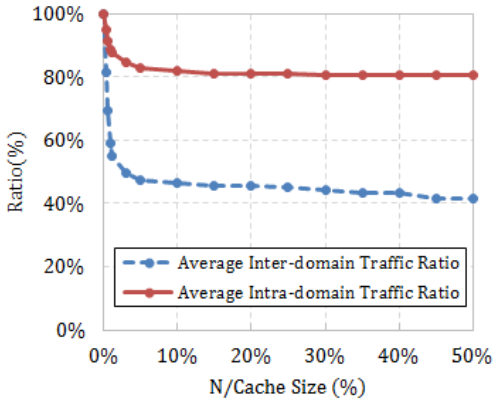


Fig. 6. Performance with different subset size n

VI. Related Work

Unlike flat self-certifying naming schema with low scalability, hierarchical naming schema in NDN provides high scalability, which faces three challenges: complex name structure, large forwarding table and high throughput [15]. Limited to the tree-like forwarding model, unlike DONA [1], NetInf/SAIL [2], and MobilityFirst [3], NDN cannot support the nearest replica routing and suffers from the trade-off between routing state reduction and information loss.

To combine both high scalability and high efficiency, most ICN studies pay attention to enhancing the scalability with flat self-certifying naming schema. Some developments of DHT schemes, such as Chord [16], Hierarchical Rings [17], and Canon [18], which are proposed for peer-to-peer overlays, are adopted for ICN architectures with flat self-certifying naming schema. MDHT [19], a hierarchical name resolution service for ICN, is introduced to provide name-based anycast routing and supports constant hop resolution. H. Liu et al. [7] propose SMVDHT, a new name resolution and routing framework that employs a combination of aggregation and multi-level virtual DHTs to improve ICN scalability. A. Sharma et al. [20] introduces a next-generation global name service that addresses the challenge of how to rapidly resolve identities to network locations under high mobility.

In contrast, there are few researches on introducing the nearest replica fetching to ICN architecture with hierarchical naming schema, such as NDN. Although NDN provides the adaptive forwarding strategy [21] to improve the forwarding performance greatly, there is still much information loss due to NDN's tree-like data fetching model. As a solution,

nCDN [22], which is similar to the CDN in IP network, to ensure the requests of popular data are routed to the best data copies straightforward. However, the nCDN still needs manual deployment, and the contents in nCDN cannot be dynamically updated.

VII. Conclusion

With the flatter and flatter Internet, the NDN's tree-like data fetching model and on-path storage may not take full advantage of the rich connections and large cache resources in NDN.

To address current NDN design's problem of **low data delivery efficiency**, in this paper we propose FNR, a CDN-like enhancement to the NDN design. In FNR, when a consumer sends an Interest packet for a popular data, the data is fetched from the nearest replica in the network. To make FNR scalable, practical, and light-weighted, first, we reduce the overhead of replica announcement by taking advantage of Zipf law so that a router only announces the information of Top-N data in the CS and uses Bloom filter to further minimize the announcement traffic. Second, we employ an effective but conceptually simple design that in spirit is similar to CDN in IP network, and use one level of redirection to locate and retrieve the nearest replica for popular data.

Our evaluation results using ndnSIM simulator show that our FNR enhancement to NDN greatly improves the performance, compared to the current NDN design. FNR on average reduces the total traffic by 25.6%, the inter-domain traffic by 52.0%, and the intra-domain traffic by 18.2%, as compared to the current NDN design. Furthermore, the average latency is reduced by 37.0% and the average cost is reduced by 51.4%. To the best of our knowledge, this paper is the first NDN enhancement in the literature to support nearest replica fetching in NDN.

Acknowledgement

We sincerely thank Zhelun Wu for helping the evaluation. We also thank Yousef Azzabi, Menghan Li for their suggestions and proofreading.

This work was partly supported by the State Key Program of National Science of China under grant 61233007, the National Natural Science Foundation of China (NSFC) under grant 61472214 & 61472210, the National High Technology Development Program of China (863 program) under grant 2013AA013302, the National Key Basic Research Program of China (973 program) under grant 2013CB329105, the Tsinghua National Laboratory for Information Science and Technology key projects, the Global Talent Recruitment (Youth) Program, and the Cross-disciplinary Collaborative Teams Program for Science & Technology & Innovation of Chinese Academy of Sciences-Network and system technologies for security monitoring and information interaction in smart grid.

References

[1] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 181–192.

[2] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf)–an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.

[3] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri, "Mobilityfirst future internet architecture project," in *Proceedings of the 7th Asian Internet Engineering Conference*. ACM, 2011, pp. 1–3.

[4] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011, p. 1.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," *Proceedings of ACM CoNEXT*, 2009.

[6] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[7] H. Liu, X. De Foy, and D. Zhang, "A multi-level dht routing framework with aggregation," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*. ACM, 2012, pp. 43–48.

[8] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse?" in *Passive and Active Network Measurement*. Springer, 2008, pp. 1–10.

[9] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 147–158, 2013.

[10] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: Provider portal for applications," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 351–362.

[11] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnsim 2.0: A new version of the ndn simulator for ns-3," *NDN Technical Report*, vol. NDN, no. 0028, January 2015.

[12] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[13] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 44–49.

[14] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 126–134.

[15] T. Song, H. Yuan, P. Crowley, and B. Zhang, "Scalable name-based packet forwarding: From millions to billions," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 19–28.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[17] A. Mislove and P. Druschel, "Providing administrative control and autonomy in structured peer-to-peer overlays," in *Peer-to-Peer Systems III*. Springer, 2004, pp. 162–172.

[18] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in g major: designing dhts with hierarchical structure," in *Distributed computing systems, 2004. proceedings. 24th international conference on*. IEEE, 2004, pp. 263–272.

[19] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "Mdht: a hierarchical name resolution service for information-centric networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM, 2011, pp. 7–12.

[20] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A global name service for a highly mobile internetwork," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 247–258.

[21] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

[22] X. Jiang and J. Bi, "ncdn: Cdn enhanced with ndn," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 2014, pp. 440–445.