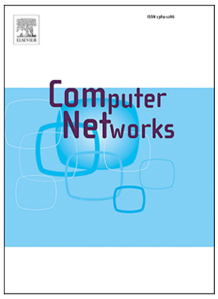
Journal Pre-proof



Cache management for large data transfers and multipath forwarding strategies in Named Data Networking

Mohammad Alhowaidi, Deepak Nadig, Boyang Hu,
Byrav Ramamurthy, Brian Bockelman

PII: S1389-1286(21)00397-2
DOI: https://doi.org/10.1016/j.comnet.2021.108437
Reference: COMPNW 108437

To appear in: *Computer Networks*

Received date : 30 September 2020
Revised date : 26 July 2021
Accepted date : 23 August 2021

Cache Management for Large Data Transfers and Multipath Forwarding Strategies in Named Data Networking*

Mohammad Alhowaidi[a], Deepak Nadig[b,*], Boyang Hu[a], Byrav Ramamurthy[a], Brian Bockelman[c]

[a] *Dept. of Computer Science and Engineering, University of Nebraska-Lincoln, 256 Avery Hall, Lincoln, NE 68588, USA*
[b] *Dept. of Computer and Information Technology, Purdue University, Knoy Hall of Technology, 401 N Grant St, West Lafayette, IN 47907, USA*
[c] *Morgridge Institute for Research, University of Wisconsin-Madison, 330 N Orchard Street, Madison, WI 53715, USA*

Abstract

Named Data Networking (NDN) is a promising approach to provide fast in-network access to compact muon solenoid (CMS) datasets. It proposes a content-centric rather than a host-centric approach to data retrieval. *Data* packets with unique and immutable names are retrieved from a content store (CS) using *Interest* packets. The current NDN architecture relies on forwarding strategies that are only dependent upon *on-path* caching. Such a design does not take advantage of the cached content available on the adjacent *off-path* routers in the network, thus reducing data transfer efficiency. In this work, we propose a software-defined storage-aware routing mechanism that leverages NDN router cache-states, software defined networking (SDN) and multipath forwarding strategies to improve the efficiency of very large data transfers. First, we propose a novel *distributed multipath* (D-MP) forwarding strategy and enhance-

*An earlier version of this work was presented at the 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) [1] and the 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) [2]

*Corresponding author
Email address: nadig@purdue.edu (Deepak Nadig)

[1]The authors acknowledge the valuable contributions of Dr. David R. Swanson (Deceased), Holland Computing Center, UNL, to this work.

[2]This work was conducted when the second author was at the University of Nebraska-Lincoln.

ments to the NDN Interest forwarding pipeline. In addition, we develop a *centralized* SDN-enabled control for the multipath forwarding strategy (S-MP) which leverages the global knowledge of NDN network states that distributes *Interests* efficiently. We perform extensive evaluations of our proposed methods on an at-scale wide area network (WAN) testbed spanning six geographically separated sites. Our proposed solutions easily outperform the existing NDN forwarding strategies. The D-MP strategy results in performance gains ranging between 10.4x to 12.5x over the default NDN implementation without *in-network* caching, and 12.2x to 18.4x with *in-network* caching enabled. For S-MP strategy, we demonstrate a performance improvement of 10.6x to 12.6x, and 12.9x to 18.5x, with *in-network* caching disabled and enabled, respectively. Further, we also present a comprehensive analysis of NDN cache management for large data transfers and propose a novel prefetching mechanism to improve data transfer performance. Due to their inherent capacity limitations of the NDN router caches, we use SDN to provide an intelligent and efficient solution for data distribution and routing across multiple NDN router caches. We demonstrate how software-defined control can be used to partition and distribute large CMS files based on NDN router cache state knowledge. Further, SDN control will also configure the router forwarding strategy to retrieve CMS data from the network. Our proposed solution demonstrates that the CMS datasets can be retrieved 28% − 38% faster from the NDN routers' caches than existing NDN approaches. Lastly, we develop a prefetching mechanism to improve the transfer performance of files that are not available in the router's cache.

Keywords: NDN, SDN, Compact Muon Solenoid, Forwarding Strategy, Cache Management

---

1. Introduction

Data management in high energy physics (HEP) is challenging due to its complexity and volume. These datasets are immutable once generated by the experiments, scientists repeatedly read and process these datasets. A critical challenge in the CMS workflow is how to deliver large volumes of data to researchers efficiently. An experiment data file has an average size of 2 Gigabytes, with file sizes ranging between 100 Megabytes and 20 Gigabytes. A complete dataset comprises multiple files, with the dataset sizes ranging from 2 − 100 Terabytes. Thus, providing speedy access to such datasets becomes a key enabler for data-intensive science research. The CMS experiment on the Large Hadron Collider (LHC) manages a large volume of data that currently exceeds 100PB across multiple sites. The experiment manages approximately 35PB of data (a combination of detector readouts and simulated readouts across various physics-related formats); this data is write-once and read-many. All CMS managed data is immutable once written to permanent storage[3]. Through a combination of caching and pre-placement, CMS moves its data across 50 data centers throughout the Worldwide LHC Computing Grid [4]. Data delivery for CMS experimental workflows is challenging due to the large size of the datasets. Further, exchanging the data between different sites – streaming to use laptops or offsite batch jobs – has historically required a burdensome set of middleware and dedicated computing infrastructure. Therefore, a better solution is needed to provide fail-over services, multiple repositories, and assist in the synchronization across multiple data repositories, reducing the overheads on the original dataset source and, consequently, the data transfer latencies. In this work, we study how to leverage Information-Centric Networking (ICN) to provide faster in-network CMS data access for end-users. Contrary to IP-based, host-centric Internet architectures, ICN emphasizes content by making it directly addressable and routable. The users request the data based on its name instead of IP addresses.

Named Data Networking (NDN) [5] is one such architecture that proposes the use of names for fetching data instead of relying on addresses for identifying data locality. The end-user sends an *Interest* packet with the data name and the network is responsible for both forwarding and caching the requested data. One of the main characteristics of NDN is *in-network* caching, where the router keeps a copy of the data to satisfy a future request. This reduces the latency arising from fetching the data from the source for all subsequent requests. The software defined networking [6] paradigm has generated significant interest in the information centric networking (ICN) community. SDN has been used to address the name-based routing and forwarding [7, 8] by decoupling the ICN data plane from its control plane [9].

Under the NDN paradigm, a content store (CS) acts as a cache management data structure. The CS is an *in-network* cache and performs data lookups for incoming Interests and serves the consumers without the need for forwarding the Interests to the NDN producers. In the current NDN implementation, it is only beneficial to cache the data in the CS when the cached contents are available *on the path* to the content producer. This a serious limitation as it reduces the data transfer efficiency by ignoring the (requested) cached content available on adjacent/off-path routers in the network. The adjacent/off-path routers are generally closer (in terms of the number of hops or routing cost) to the consumer when compared to the NDN producer. Therefore, fetching the data from the producer and caching it only in the on-path router instead of also utilizing the adjacent/off-path routers is inefficient. Thus, fetching the data from both the on/off path routers would greatly improve the data retrieval performance.

In this paper, we propose a multipath forwarding strategy to address the above problem. Our first approach proposes enhancements to the existing NDN Forwarding Daemon (NFD) implementation. Specifically, we propose a forwarding strategy that retrieves non-overlapping data packets from multiple routers simultaneously. Further, the strategy provides additional flexibility in the per-router choice of the Interest pipeline depth configuration. Next, we propose a centralized approach using a SDN controller for managing/mapping the current contents of the CS. This approach allows us to make intelligent Interest pipeline forwarding decisions by analyzing the global view of the NDN network. The SDN controller is effectively used to analyze the network state and redirects the incoming Interests to the off-path routers that have cached the requested content. In our work, we enhance the data retrieval process for both cases by allowing both the NDN consumer and the NDN routers to fetch the content from multiple off-path locations based on the network states. Our proposed approaches, while improving data transfer performance on the one hand, also ensures congestion avoidance on a specific path by distributing Interests across multiple available paths.

To employ NDN for CMS workflows, we must also address the critical challenge of how to store large files in the network efficiently. In the NDN paradigm, a content store (CS) is an *in-network cache* that performs data lookups for incoming Interests and serves the consumers without the need to forward the Interests to the NDN producers. Due to the limitation of the cache capacity on each NDN router, novel approaches for efficient cache management are necessary. One approach is to deploy the NDN routers with large memory. However, this will increase the deployment costs and is therefore inefficient. Another approach is to use solid-state drives (SSD) for caching the data; the use of SSDs for caching not only increase the overall cost of the deployment but also add additional data retrieval latencies. In the Information-Centric Networking (ICN) community, software-defined approaches for NDN routing intelligence and caching management is an active area of research. To address the above problems, we propose a solution that employs an SDN controller to manage cache-aware NDN routers. Our proposed approach works in two phases. First, during the file retrieval process, if the file is not cached in the network (and resides on the producer storage), the interest packet will be forwarded to the centralized controller for the best retrieval strategy. Small files are retrieved using the default NDN approach. However, for large files that cannot be cached on a single router, distributed retrieval approach using multiple router content stores will be used. Second, depending on whether the requested file is already cached on multiple routers or not, the controller will provide a strategy for distributed file retrieval (See Section 3). Further, our proposed system architecture also enables the *prefetch feature*, where parts of the file can be prefetched and cached on different routers simultaneously.

Specifically, our solution, in comparison to the original NDN data repository and synchronization implementations, exploits multiple paths and off-path routers (not possible in the default NDN implementation) to optimize end-to-end data transfers. Further, our solution provides a better data management solution by offloading key decision-making tasks to an SDN controller.

The main contributions of our work are listed below:

1. We propose a *distributed* multipath (D-MP) forwarding strategy for NDN Interest pipeline processing and data retrieval. This approach demonstrates simultaneous data retrieval from a set of n routers with pre-configured Interest pipeline depths. In comparison to the default NDN implementation, our D-MP strategy performs over 10x better than the alternative.
2. We propose a *centralized*, SDN-enabled control for our multipath forwarding strategy (S-MP). We show that the centralized control (S-MP), unlike the D-MP case, provides additional benefits due to the knowledge of the global NDN network and cache state.
3. For both D-MP and S-MP approaches, we present NFD configuration algorithms detailing the consumer, Interest and routing pipelines, interfaces and the Interest distribution strategy.
4. We present cache management strategies for large data transfers using NDN. Using software-defined control, we present strategies for partitioning and distributing large CMS files based on NDN routers' cache-state knowledge.
5. We also develop a prefetching mechanism to reduce the data retrieval latency specifically for large file transfers. Our proposed approach further improves the data transfer performance by optimizing the file retrieval time while reducing the path latency.
6. Lastly, we evaluate the performance of our multipath forwarding and cache management solutions for large data transfers on an at-scale, geographically distributed wide area network (WAN) research testbed and provide valuable WAN performance insights.

The paper is organized as follows: Section 2 presents background on named data networking (NDN), software defined networking (SDN) for NDN and the related works; In Section 3, we describe our proposed system architecture for NDN multipath forwarding strategies and SDN control of NDN; Section 4 outlines our solution approach for NDN Interest pipeline management for both D-MP and S-MP usecases. In Section 5, we describe our proposed approach for NDN cache management for large file transfers. We describe our evaluation framework, network testbed setup and experimental design for multipath forwarding strategies in Section 7. In Section 8, we present extensive results and discussions for our proposed multipath forwarding strategies, SDN control for NDN and large data transfer cache management approaches. Finally, we conclude our work in Section 9.

2. Background and Related Work

2.1. Named Data Networking

The traditional IP-networking has problems such as IP mobility, network address translation (NAT) traversal, and address space limits. Named Data Networking (NDN) [5] is an excellent solution to mitigate such problems. NDN is a Future Internet Architecture (FIA) [10] project that proposes re-designing the current host-centric Internet architecture. It is developed on a name-based packet forwarding and routing scheme, using a hierarchical and unbounded namespace. These ensure the communication continuity as the data is no longer bound to the host address, provide data mobility, and eliminate address-space management in the network. NDN requests data using its name instead of the IP address. There are two types of packets in NDN: (i) Interest packet, which contains the data name, and (ii) Data packet, which contains the data to be sent back to the consumer. NDN names are hierarchically structured. For instance, the name */ndn/repository/file*, is carried by the Interest and is used to forward the data to the content custodian. NDN routing is similar to its IP network counterpart but with longest-prefix matches performed on the data *names* instead of the IP addresses. Each NDN router maintains a forwarding information base (FIB) populated with name prefixes. A name-based routing protocol is used to populate each router with the *name* prefix and the associated interface on which the data can be retrieved.

NDN Forwarding Daemon (NFD) is responsible for routing the Interest and caching the data. NFD manages three data structures: Pending Interest Table (PIT), Content Store (CS), and Forwarding Interest Base (FIB). The NDN consumer generates and sends the Interest packet in the NDN network. When the router receives the Interest packet, it uses the data name to forward the packets to the NDN producer (i.e., data custodian). The router will store the Interest in its PIT along with the incoming interfaces. Suppose the same Interest packet, from another consumer, reach the router. In that case, the PIT will return the Data packet to the consumer upon receiving it. The NDN router will reply directly to the consumer if the data is already stored in the cache. Otherwise, the Interest packet will be forwarded to the producer. The caching mechanism is an efficient way to reduce the latency of retrieving data and reduce overheads on the producer. The FIB acts as the routing table for the router.

NDN routers have different forwarding strategies that define how and where to forward Interest packets. The forwarding strategy chooses the next hop for forwarding until the Interest reaches the destination. It can also select different paths to retrieve the data packets. For instance, the forwarding strategy can forward the packets based on the shortest (number of hops) path, lowest latency path, or least congested path.

2.2. Software Defined Networking

Numerous research efforts have focused on ensuring network programmability and software defined networking (SDN)[11] is a critical part of this effort. The popularity of SDN has increased rapidly over the last few years and it has become a well investigated area of research. The main idea behind the success of the SDN is that it provides a separation between the control plane and the data plane. This separation led to many benefits such as reducing overheads for the network operators in managing the different parts of the network and connecting different types of networks. Further, it provides an automation process to control and forward traffic through the network. SDN relies on network configuration based on a programmable policy. Further, SDN reacts to different network conditions by selecting the fastest route to avoid congestion.

There are numerous advantages to implementing NDN over SDN, including routing [12], forwarding [8], security monitoring [13] and orchestration [14]. The authors in [15] used SDN and OpenFlow to optimize the TCP congestion control performance in NDN. Controller-based routing schemes were developed by the authors in [16] to support mobility in NDN. SDN, virtualization technologies and network functions virtualization (NFV), and their benefits to network management, service provisioning and quality of service (QoS) have also been explored in [14, 17, 18]. In this work, we focus on using SDN to manage NDN routers' cache and apply multipath forwarding strategies to improve large data transfer efficiency.

2.3. Related Work

Several other works focus on SDN-NDN integration, improving content caching and placement, and routing/forwarding mechanisms. The authors in [19] proposed using a controller to perform content selection and placement on specific off-path routers. Other approaches to optimizing NDN caches include joint-path and off-path cooperative caching policies [20], content popularity based multi-path forwarding and caching strategies [21], and the use of network coding and cache content placement to achieve better bandwidth and cache cost performance [22]. Interest routing and forwarding strategies in [23] rely on the discovery of temporary copies of the content not available in on-path caches to forward data requests on each hop. However, the above works only focus on caching optimization to improve existing forwarding strategy performance. The authors in [24] proposed SDCCN, to program content-centric networking (CCN) forwarding strategies and caching policies using an SDN approach. However, this approach focuses on cache replacement algorithms for improving strategy performance. Unlike the above works, our work focuses on developing novel forwarding strategies for Interest pipeline management.

Numerous recent works explore the use of centralized control for managing information centric networks (ICN). Works such as [19], [20], [21], and [23] explore cache placement, caching policies, and content selection strategies on both on- and off-path routers. SDN-based control of ICN is also proposed for distributed data transfers in data-intensive science [25, 26, 27]. The authors in [1] discuss multi-path interest distribution strategies for both distributed and centralized control of NDN and how it can improve data transfer performance. However, the interest distribution strategies presented in [1] do not consider network layer properties such as NDN on-path congestion and the route bandwidth availability. Unlike the above works, we propose a centralized cache management framework that considers the limitation of the NDN router cache for large data transfer. Previous works deal either with data retrieval performance and cache placement or develop techniques for off-path data retrieval. However, the missing piece in those works, which we address in this work, is the cache management framework and the collaboration between different routers' CS to cache large files and return it to the user efficiently.

3. System Architecture

NFD employs a per-namespace forwarding strategy to forward Interests. The strategy choice would affect packet forwarding decisions and play an important role in fetching the data from a given NDN router. Several Interest forwarding strategies are available for use by the NFD, including best routes, multicast, client control, NCC (implemented from CCNx, i.e., CCN backward), access router, and adaptive SRTT (smoothed round-trip time) -based Forwarding (ASF) [28] strategy.

Although efficient for many existing network environments, the strategies described above do not cater to the necessary performance requirements of large-scale distributed datasets. We develop strategies suitable for large-volume distributed data transfers over high-bandwidth, high-delay wide area networks (WANs). The targeted use of the developed strategies and forwarding pipelines are complex and distributed file systems such as CernVM File System (CVMFS) [29]. High-energy physics (HEP) workflows (e.g., Compact Muon Solenoid (CMS) [30]) are evaluating the use of CVMFS for the distribution of experimental datasets [25] using NDN. Next, we outline two approaches to Interest distribution: i) a distributed multipath (D-MP) strategy and enhancements to the existing NDN Interest pipeline, and ii) A centralized SDN-enabled control for the multipath strategy (S-MP).

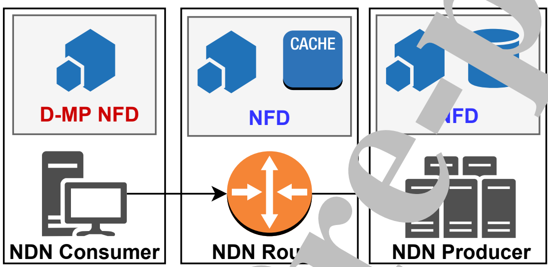


Figure 1: D-MP: Distributed Multipath Network Architecture.

3.1. D-MP: Distributed Multipath Forwarding Strategy

NFD data transfer decisions use a combination of strategy choice and the forwarding pipeline depth. Together, they form the NFDs' intelligence and packet processing logic. The strategy choice influences the packet forwarding decisions. The forwarding pipeline specifies the number of simultaneous Interests that are forwarded per request. We propose NFD strategy enhancements to optimize the NFD forwarding pipeline. The architecture is as shown in Figure 1. Our NFD enhancements enable parallel data (or namespace) retrieval from multiple routers using a per-router Interest pipeline depth. Our proposed distributed multipath (D-MP) strategy benefits from multipath gains and/or *off-path caching* to reduce the latency of data-delivery to the consumer. Unlike the multicast strategy, our optimizations focus on parallel data retrieval from a set of NDN routers. We also consider the effects of caching at the content store (CS) of each router on data retrieval times. Using the D-MP strategy, the data consumer can simultaneously request non-overlapping data segments from multiple routers. Further, our approach enables the consumer to specify a per-router forwarding pipeline depth. Thus, D-MP can optimize parallel data transfers from multiple NDN routers based on the exchanged information between the NDN consumer and NDN routers. Details of the D-MP approach are presented in Section 4.1.

3.2. S-MP: Centralized SDN control for the Multipath Forwarding Strategy

The D-MP approach described in the previous section relies on *priori* information about the available router forwarding paths for forwarding decisions. Although the D-MP approach benefits from multipath data retrieval and larger optimized Interest pipelines per path, it is vulnerable to dynamic network state changes due to its dependence on a priori information. To facilitate the use of real-time NDN network state information in Interest forwarding decisions, we propose a software-defined control architecture for the multipath forwarding strategy. The architecture is shown in Figure 2. The S-MP architecture uses representational state transfer (REST) application programming interfaces (APIs) for information exchange between the NDN and the SDN infrastructures. The centralized SDN controller manages the NDN network state information, including router states, available forwarding paths, and cached contents. It also asynchronously communicates with the NDN routers and the content producers to create a data map of the CSs of the NDN routers. They are also representative of the data cached in the memory buffer of each NDN router. The S-MP strategy involves the following:

3.2.1. Consumer NFD Configuration

First, we choose the set of routers that already cache the requested content either partially or fully. Based on the caching information, we formulate a multi-router Interest distribution strategy and communicate it to the data consumer NFD. The consumer's NFD configures the associated faces, routes, and Interest pipeline depths. It initiates the parallel non-overlapping data retrieval from multiple NDN routers. Suppose the requested data caching is unavailable at the routers. In that case, the SDN controller sends a list of the best candidate routers and the corresponding Interest distribution strategy to the consumer NFD. The consumer uses this information as before to set up the connections.

3.2.2. Data Retrieval

The consumer NFD establishes parallel connections with the specified list of routers and configures each connection with an associated Interest pipeline depth. Parallel connections retrieve the requested data and assemble it at the consumer. If no data is cached at the routers, the consumer NFD sends the Interests for non-overlapping data segments to the routers. The routers in-turn fetch and cache the requested data from the NDN producers and deliver it to the consumer.

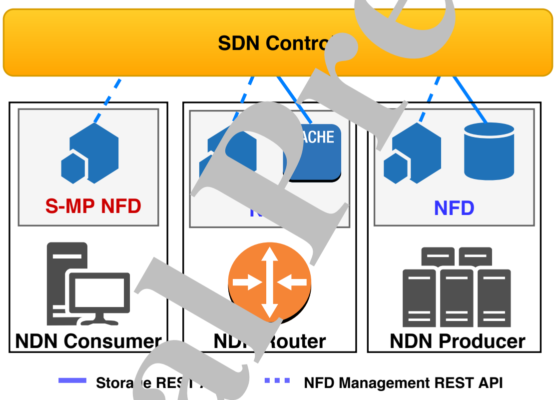


Figure 2: S-MP SDN-enabled control for multipath forwarding.

Two types of messages are exchanged between NDN routers and the SDN controller. Namely, 1) *Forward* message, used when the requested data is unavailable at the router, and 2) *Update* message, used when the new data is received and cached at the router. When the Interest arrives at the NDN router, the NFD checks the content store (CS) for the requested data. If the data is available, the router will return a copy to the consumer. Otherwise, the router's NFD forwards the Interest along the path to the NDN producer. The SDN controller uses the *Update* message to update its NDN state information data. An *Update* message is sent to the SDN controller on a per data packet (segment) basis. The *Update* packets are sent to the controller for both data addition or data removal from the NDN repositories; Thus, these messages are used in two scenarios, namely (i) Addition: when the segment is received by the NDN router and cached, (ii) Removal: when an existing segment is removed by the cache replacement policy. Table 1 shows the message types exchanged between the SDN controller and NDN. Details of the S-MP strategy are presented in Section 4.2. Next, we present our cache management architecture for large data transfers.

Table 1: SDN-NDN Interaction Message Types

Message Type	**Purpose**	**Size**
Forward	Indicates that the data is unavailable at the router	Variable (TLV)
Update	Indicates that new data is available and cached at the router	Variable (TLV)

3.3. Cache Management Architecture

NFD forwarding strategy is responsible for choosing the interest forwarding interface. Designing and selecting the correct strategy will affect the data retrieval process and performance. Our cache management architecture can be combined with our proposed multipath forwarding strategies for improved performance.

The default NFD forwarding strategies can be used in different network environments, however, they are unsuitable for large-scale datasets comprising of large files (i.e., files larger that the routers' CS size). Since NDN relies mainly on cached data to reduce retrieval latency, the above strategies ignore the environments that deal with large datasets. In this paper, we propose a solution for cache management and develop a forwarding strategy that deals with large datasets. We develop an architecture and associated strategies to deal with large-volume distributed data transfers over high-bandwidth, high-delay wide area networks (WANs). The targeted use of our architecture is the complex and distributed filesystems such as CernVM File System (CVMFS) [29]. High-energy physics (HEP) workflows (e.g., Compact Muon Solenoid (CMS) [30]) are evaluating the use of CVMFS for the distribution of experimental datasets [25] using NDN.

3.3.1. Explanatory Example

For large data transfers that exceed the Router CS size, a cache miss will always occur, and the data will be fetched from the producer. The following example illustrates the need for our proposed solution:

Suppose the NDN content store size is 500MB (default size in the current NDN implementation), and we have a 1GB file with name *"large"*. The file will be segmented into several chunks and fetch by several interest packets. The interest name will be, for instance, */ndn/large/segNum=1* for the file's first chunk. For simplicity, let us assume that the chunk size is 1MB. Then, we will have 1000 interests to fetch the file *"large"*. Since the CS size is 500MB, chunks 1-500 will first fill up the CS then chunks 501-1000 will start replacing the chunks in the CS if a regular replacement strategy is employed. Example strategies include the first in first out (FIFO) strategy or the least recently used (LRU) strategy. Now, suppose the file is requested again by the same consumer or another consumer. In that case, the interests will start by fetching chunk 1. Since the CS currently contains chunks (501-1000), then a cache miss will occur. The interest will be forwarded to the producer. The chunks (1-500) will again fill the CS and result in a cache miss. The same scenario will repeat when fetching the chunks (501-1000).

3.3.2. Architecture

Figure 3 shows an overview of our cache management architecture. All routers and the producer communicates with a centralized controller for cache

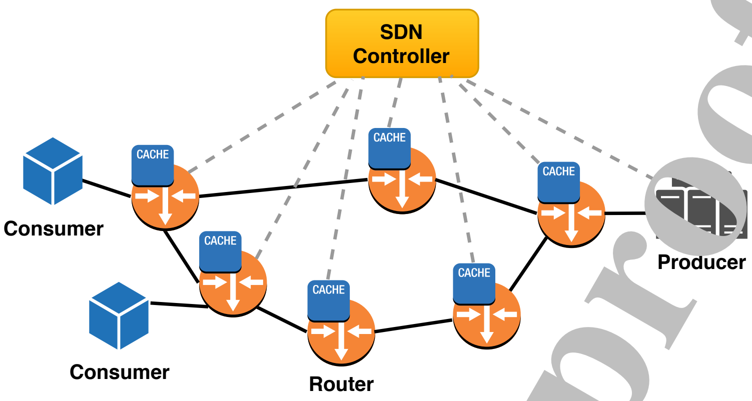


Figure 3: Cache management architecture.

management and data retrieval. The router/producer to SDN controller communication employs representational state transfer (REST) application programming interfaces (API) to interact with an SDN application over an out-of-band link. The centralized controller manages all routers' CS and the forwarding strategies that need to be installed on the routers. The system architecture is described below:

1. *NDN Consumer*: The NDN consumer represents the user requesting the data. The Consumer in our architecture is unaware of the underlying architecture and requests the data directly by sending Interest to the network.
2. *NDN Router*: The NDN routers are responsible for caching the data in their own CS. Also, the NDN routers will use the forwarding strategy to forward the interest to the next-hop router. The NDN routers collaborate with the controller while storing the data. If the file size is small, then one NDN router is used to store the file. However, for large files, the file will be stored on multiple NDN routers' CS. Splitting the file among multiple NDN routers will avoid premature CS cache replacement. It will also ensure faster data retrievals due to the file transferred from in-network caching.

3. *NDN Producer*: The NDN producer represents the content custodian or the storage where all data resides and caters to the consumer interests/requests. The producer can store the actual files or store them as NDN packets. Storing the data as NDN packets in advance will avoid the overhead from converting the regular files into NDN signed packet. In this work, we convert and segment the files into NDN packets. For segmenting the file, we retain the default NDN chunk size (i.e., 8000 bytes). The NDN producer will update the centralized controller about the files that it has in its repository and the size (number of chunks) of each file. The controller will store this information in its database for cache management purposes.
4. *Centralized Controller*: The controller in this work represents the primary entity in managing the interest routing and caching. It receives Interest from routers and sends different forwarding strategies to the corresponding routers for caching the whole file or partially, based on each router's CS and the file size. The centralized controller uses REST APIs for communication with the NDN routers and the NDN producer. The controller manages the NDN network state information, including routers CS states (i.e., the available space on CS), forwarding paths, and data information on the NDN producer. We run a program alongside the NFD on all NDN routers and the NDN producer. This program is responsible for internal communication with the NFD and external communication with the SDN controller. The controller asynchronously communicates with this program to create a data map of all routers' CS.

Figure 4 shows an example of the use of our architecture. When consumer C wants to retrieve a large file from the NDN producer, it will create an interest and send it to router R1. If it is a new interest, then router R1 will send the interest to the controller. The controller will read the file information and the Routers CS from its database. In our example, the controller finds that routers R2, R3, and R7 have enough space in their CS to store the large file. The controller will send the configuration to router R1 to split the interest for the file segments into three paths. If the file is split into n segments $(S_0 ... S_k .. S_r .. S_{n-1})$, these segments will be retrieved as follows:

- $S_0 - S_{(k-1)}$ through path $R1 \rightarrow R2 \rightarrow R5 \rightarrow P$
- $S_{(k)} - S_{(r-1)}$ through path $R1 \rightarrow R3 \rightarrow R6 \rightarrow P$
- $S_{(r)} - S_{(n-1)}$ through path $R1 \rightarrow R4 \rightarrow R7 \rightarrow P$

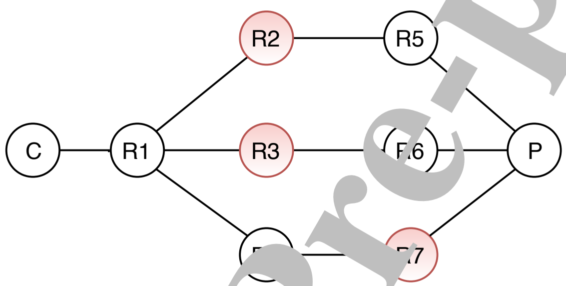


Figure 4: Distributed Interest example.

The controller will configure the routers R2, R3, and R7 to cache these segments. Simultaneously, the controller configures the routers R4, R5, and R6 not to cache those segments. This cache management strategy will help in reducing the number of cache misses on other routers. Further, it also benefits from other routers in caching other files.

3.3.3. Prefetching File Segments

We developed a prefetch mechanism to reduce the latency in retrieving files. In our previous example (Figure 4), the controller will ask routers R3 and R7 to start a parallel data segment retrieval. Since R1 starts retrieving segments $(S_0 - S_{(k-1)})$ first. The controller will tells router R3 and router R7 to retrieve segments $S_{(k)} - S_{(r-1)})$ and segments $S_{(r)} - S_{(n-1)})$, respectively and store them in their CS. When R1 finishes retrieving the first set of segments from the producer, it will start retrieving the second and third set of segments from routers R3 and R7 instead of the producer. Thus, it will reduce the path latency and optimize the overall data retrieval time.

4. Multipath Forwarding Strategies for NDN

The default NDN implementation relies on data caching only *on-the-path* to the content producer. This limitation reduces the data transfer efficiency as it ignores the (requested) cached content that may be available on adjacent/off-path routers in the network. These *off-path* routers are generally closer to the consumer when compared to the content producer. Therefore, data retrieval from both *on-* and *off-path* routers can greatly improve data retrieval performance. In this section, we outline the different implementation approaches for multipath Interest pipeline distribution. We discuss our forwarding strategies for both cases, i.e., D-MP and S-MP.

4.1. Interest Pipeline Distribution Approaches for D-MP

The D-MP-based NFD configuration and data transfers are outlined in Algorithm 1. For each incoming Interest packet, the consumer NFD computes the optimal forwarding strategy, a list of routers, and their corresponding pipeline depths. This router set is created based on a discovery phase. During this phase, the consumer sends a message to all routers to check if the data is available in the routers' caches. The consumer processes the replies from the routers and builds a forwarding strategy configuration. This configuration contains information about the NFD Face to use with each router and a per-router Interest pipeline depth.

Different approaches can be used for Interest pipeline distribution for the D-MP case. We implement a round-robin scheme for Interest distribution among a set of n routers. In this approach, we distribute $i, \forall i \in \{1, .., p\}$ *Interests* to the processing pipeline of n routers. This approach ensures that the Interest processing pipeline of each router is always saturated with the defined pipeline depth (i.e., p) for optimal performance. Another approach is to use a ratio-based

Interest distribution scheme. In this approach, Interests are distributed to a set of n routers based on a defined ratio partitioning scheme. For example, for a set of $n = 3$ routers and a pipeline depth, $p = 20$, an Interest distribution ratio of $40\%, 40\%$, and 20%, results in the Algorithm 1 assigning 8 Interests to routers 1 and 2, and 4 Interests to router 3 respectively in a non-overlapping fashion. This ratio can be calculated based on the network state information obtained during the discovery phase and changed during the data transfer. Thus, the ratio partitioning approach provides additional flexibility for adjusting the Interest processing pipeline depths for each router to Thetotal computing delay for the entire data transfer (both datasets outlined in Table 2) is on averageabout 900ms (as opposed to data transfer times that vary between 110 to 625 seconds for the100MB file transfers. Therefore, we focus on evaluating the end-to-end data transfer performance forlarge files as the SDN controller performance does not adversely affect data transfer performance.better balance routers' loads and/or processing capacities. In this work, we only present the performance results for the round-robin scheme.

Algorithm 1 D-MP(i)

Input: NDN Interest (i).

Output: D-MP NFD configuration for data transfer.

Consumer NFD Configuration Update:

for all $r \in R$ **do**
  Consumer sends discovery message to router r
  Consumer processes reply from router r
end for
Consumer computes optimal *namespace* configuration
Consumer update $C : i_{faces}, C : i_{pipeline}, C : i_{distribution}$

Consumer NFD Data Retrieval:

for all Routers $r \in C : i_{faces}$ **do**
  Configure router pipeline $r_p \leftarrow C : i_{pipeline}(r)$
  Forward *Interests* based on $C : i_{distribution}$
end for

4.2. SDN-enabled Centralized Interest Pipeline (S-MP)

The SDN-enabled centralized Interest pipeline distribution approach forwards consumer Interests for all requested names to the SDN controller. Algorithm 2 describes the consumer NFD processes and Algorithm 3 describes the SDN controller functionality.

Algorithm 2 S-MP(i)

Input: NDN Interest (i).

Output: S-MP NFD configuration for data transfer.

SDN-enabled Consumer NFD Configuration:

SD-NFDConfig(i)
$C : i_{faces} \leftarrow router\_list$
$C : i_{pipeline} \leftarrow p$
$C : i_{distribution} \leftarrow distribution\_map$

Consumer NFD Data Retrieval:

for all Routers $r \in C : i_{faces}$ **do**
    Configure router pipeline $r_p \leftarrow C : i_{pipeline}(r)$
    Forward *Interests* based on $C : i_{distribution}$
end for

The SDN controller is responsible for computing the forwarding strategy, multi-router configuration, and specifying the per-router pipeline depth for a given data transfer request. The SDN controller manages a map of the state of the content store (CS) compiled from all the NDN routers in the network. It also maps the off-path routers that host the requested data. Further, it computes decision statistics based on the routers' status and their network state information. It will then communicate the appropriate strategy configuration to the NDN consumer to in turn set up the necessary connections. It is to be noted that the communication between the SDN controller and the NDN routers/NDN producers are independent of the consumers' data requests. For every CS state change, the router sends a REST POST to the controller to notify the cache update. Optimum Interest pipeline distribution decisions are made based on the state of the CS of each router (either on- or off-path) in the network. The optimum decision specifies the forwarding strategy, the total number of routers, and the associated pipeline depths for configuring the consumer NFD for each router in the configuration.

Algorithm 3 SD-NFDConfig(i)

Input: NDN Interest (i).

Output: SD-NFD Configuration File.

Lookup data map for the *namespace* in Interest i
Compute optimal *namespace* configuration
return $\{router\_list, p, distribution\_map\}$

5. Cache Management for Large Data Transfers

In this section, we describe the implementation approach for cache management and data distribution.

5.1. Controller algorithm

When the router receives an Interest, it will forward it to the controller, asking for the best configuration to forward it. Algorithm 4 shows the process on the controller to build the configuration file. The controller will take the Interest as input, receive the file information, and the routers' CS status from its data map (database). It will sort the routers based on the available space in each router's CS. The routers with the largest CS available space will be used first. Based on each router's CS size and the available CS space, the controller will decide the number of routers needed to cache the file. Sorting the routers based on the CS space will avoid splitting the file among too many routers and avoid additional delays in retrieving the file later. The controller will send the configuration instructions to the routers to forward Interests for the file retrieval.

Algorithm 4 ControllerConfig(i)

Input: NDN Interest (i).

Output: SD-NFD Configuration File.

Lookup data map for the *namespace* in Interest i
Sort the routers based on the available space of the routers CS
Compute the number of routers that is needed to cache the file
return SD-NFD-Config(i)

5.2. Router Forwarding Strategy Algorithm

Algorithm 5 describes the procedure on the router. We develop a procedure for communication between the NFD on the router and the SDN controller. When an interest reaches the router, the router will check if it is requesting a chunk from the file with an installed configuration, or if this Interest is a new one. If the Interest is new, then the router will send the Interest information to the controller. The controller will reply with the instructions on how to forward the Interest, carried in the NFD-Config file. The router will use this file to forward the Interests to retrieve the specific data accordingly. The NFD-Config file carries the information on how to divide the Interests by requesting the file from several interfaces. Since the file is segmented into several chunks, each Interest will retrieve a specific chunk. The NFD-Config file will tell the router to forward a set of Interests requesting $(segment_0, segment_{i-1})$ on one interface, and another set of Interests requesting $(segment_i, segment_{j-1})$ on a different interface.

Algorithm 5 Router configuration(i)

Input: NDN Interest (i).

Output: Router NFD configuration for data transfer.

Configuration request:

if Interest not configured **then**
  Send Interest information to the controller
else
  Forward Interest based on the existing configuration
end if

Configuration arrival:

Read NFD-Config(i)
for all faces in NFD-Config **do**
  Forward Interests of $(segment_i, segment_k)$
end for

On the other hand, the controller will send instructions to the corresponding routers that will store the file segments in their CS to enable the caching process for these file segments. If the router does not receive this message, it will just pass the data without caching it.

Algorithm 6 Prefetch procedure

Input: prefetch_MSG.

Output: Request and cache file segments.

```
READ prefetch_MSG
for Seg = startSeg; Seg < endSeg; Seg++ do
    Interest = /ndn/fileName/Seg
    Send Interest
end for
```

Further, in the prefetch scenario, the controller will inform the NDN routers that will cache the file to start fetching parts of the file. The first router will retrieve the file segments in the usual way (Interests coming from the consumer). In contrast, all other routers that cache the other parts of the file will start issuing Interests to store the corresponding segments in their CS. Algorithm 6 shows our prefetch process. Once the router receives prefetch_MSG from the controller, the router will read the file name, prefix, and the range of the segments that are needed to be fetched. Then, the router will issue these interests and cache the segments in its CS even before the actual Interests sent from the NDN consumer.

6. NDN Multipath Strategy Analysis

In order to understand the benefits of the multipath strategies detailed in this work, we present the analysis and comparison of our proposed multipath forwarding strategy with the default NDN strategy. The default NDN strategy relies on sending a single Interest packet from the NDN consumer to the NDN producer through a single pre-configured path. In contrast, our proposed multipath forwarding strategies employ multiple paths to retrieve data simultaneously from nearby or adjacent off-path routers. The parameters used in our analysis are presented in Table 2.

Table 2: Parameters for NDN-Multipath Analysis.

Parameter	Description
NPS	The NDN packet size (in bytes) which carries the data from the producer to the consumer in along a given path.
DP	The pipeline depth representing the total number of simultaneous Interests sent simultaneously from a given consumer.
MP	Multipath parameter which represent the total number of paths used to forward the Interest packets.
B_i	The achievable throughput B for link i.
s_{num}	The total number of NDN segments representing the entire NDN data file that is stored on the NDN producer or cached at the router.
δ_i	The latency of the link i in milliseconds.
δ_{total}	The latency to retrieve the complete file in milliseconds.

Using the parameters defined in Table 2, we compare the latency performance for the following scenarios: (i) the default NDN implementation where one interest packet is sent and one data packet is retrieved at a time; (ii) A multi-interest design where multiple interests are sent on one path simultaneously to retrieve multiple (unique) data packets on a given link, and (iii) A multi-interest, multipath design which is our proposed solution relies on to send multiple (unique) Interests on different paths to the producer.

For the default NDN implementation, the latency to transfer one segment will depend on the link throughput and the time required to send the packet on the specific path. Therefore, if the file/data consist of a total of s_{num} segments, the latency to successfully transmit the entire file is given by

$$\delta_{total} = s_{num} \times \delta_i$$

The latency can be reduced using methods such as the multi-interest techniques and multipath data retrieval. Unlike the default NDN implementation where instead of transferring one segment at a time, we can transfer multiple segments based on the throughput, B_i of the link i. As a result, the latency reduces to:

$$\delta_{total} = \frac{s_{num} \times \delta_i}{DP}$$

where the DP value is selected based on the total number of available paths, each with B_i for the link i.

Additionally, significant improvements can be achieved through our proposed solution by considering a multipath strategy for data retrieval. In this case, the latency will be distributed across multiple paths by sending simultaneous (non-overlapping) interest packets on several paths instead of a single path. Therefore, the total latency is given by:

$$\delta_{total} = \left(\frac{s_{1..j} \times \delta_1}{DP1} + \frac{s_{j+1..k} \times \delta_2}{DP2} + .. + \frac{s_{n..num} \times \delta_n}{DPn}\right)/MP$$

where $s_{1..j}$ is the first set of segments simultaneously transfer on $path_1$, and $s_{n..num}$ is the last set of segments transfer on the last path $path_n$.

Assuming that all paths have the same achievable throughput and propagation delay, the total latency will be reduced by a factor of MP as follows:

$$\delta_{total} = \frac{s_{num} \times \delta_i}{DP \times MP}$$

The worst case scenario, in this case, is that we constrain the pipeline depth, $DP = 1$, and the total number of paths, $MP = 1$, which corresponds to the default NDN implementation of sending a single interest packet over a single path. Therefore, our proposed solution provides improved performance over the default NDN implementation. The only additional overhead of our design in comparison to the default NDN design is the communication between the NDN routers and the SDN controller. The NDN-to-SDN communication overheads are negligible as the SDN controller is usually closer to the NDN consumers and employs a single control packer per file transfer.

As an example, we will consider the network topology shown in Figure 4 in Section 3.3.2. From Figure 4, we have three paths between the consumer and the producer, namely: (i) $Path_1$ (P_1): $R1 \rightarrow R2 \rightarrow R5 \rightarrow P$, (ii) $Path_2$ (P_2): $R1 \rightarrow R3 \rightarrow R6 \rightarrow P$, and (iii) $Path_3$ ($P3$): $R1 \rightarrow R4 \rightarrow R7 \rightarrow P$). For simplicity, we assume that the transmission time and propagation delay is same for all paths. Now, consider a file which needs to be split into 100 segments. Assuming that the latency to retrieve one segment is 10ms, the overall latency to finish retrieving the whole file will be:

- Default NDN implementation: The implementation will need to retrieve 100 segments, where the second segment will be retrieved after getting the first segment sequentially, until the whole file is fetched. Also, the packets will traverse one path only. This will give us an overall latency as $100 \times 10ms$.
- Multi-interest design: The latency value will be reduced by the value of chosen DP. The DP value is chosen based on the B_i of that link. Then, if we choose P_1 to transmit the packet, then multiple segments DP would be retrieved at same time.
- Multi-interest, Multipath design: In this scenario, all paths (P_1, P_2, and P_3) could be selected to retrieve the segments. Based on the B_i of each path, then a separate DP will be set for each path. Therefore, the total number of simultaneous segments can be $DP \times MP$.

7. Evaluation

In this section, we describe our network testbed setup, datasets used in the performance evaluation, associated parameters, and the experimental design.

7.1. Network Testbed

Our test network topology is shown in Figure 5. The test network is composed of two NDN consumers, three NDN routers, an NDN producer, and an

SDN controller node. Consumer $C1$ is connected to all three routers ($R1$, $R2$, and $R3$). $C1$ is the main data consumer for all our tests. It implements our NFD forwarding strategies and Interest pipeline distribution approaches. Consumer $C2$ is only connected to $R1$, and its path to the NDN producer is $C1 \rightarrow R1 \rightarrow R2 \rightarrow R3 \rightarrow P$. The consumer $C2$ is *only used* for populating all routers with the same dataset for the tests with *in-network* caching enabled.

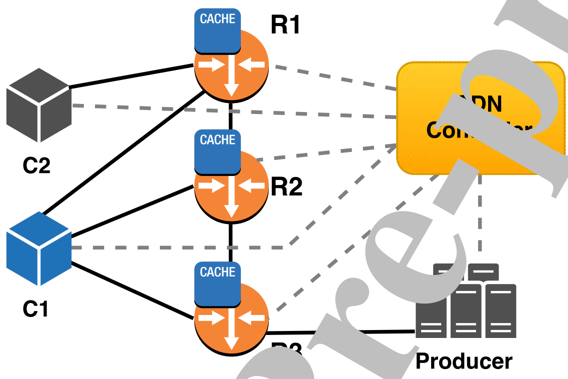


Figure 5: Test network for SDN enabled multipath forwarding.

All nodes in the test network are set up on the GENI (Global Environment for Network Innovations) [31] platform. GENI provides a platform for at-scale networking research, connecting compute resources over the Internet2 AL2S infrastructure. We use six GENI sites (with one NDN node per site) spread across InstaGENI infrastructure at Georgia Tech, Kansas, Rutgers, Stanford, UCLA, and UChicago. Therefore, this setup is representative of a real-world WAN NDN network.

Figure 6 shows our network topology for cache management experiments. This testbed consists of one NDN consumer, four NDN routers, an NDN producer, and a controller node. The NDN Consumer, all NDN Routers, and the NDN Producer are running *NDN-cxx* and *NFD*.

We used the GENI [31] platform as our network testbed. GENI provides a platform for at-scale networking research, connecting compute resources over the Internet2 AL2S infrastructure. We use seven GENI sites (with one node per site) spread across InstaGENI infrastructures at Kentucky MCV, Kentucky

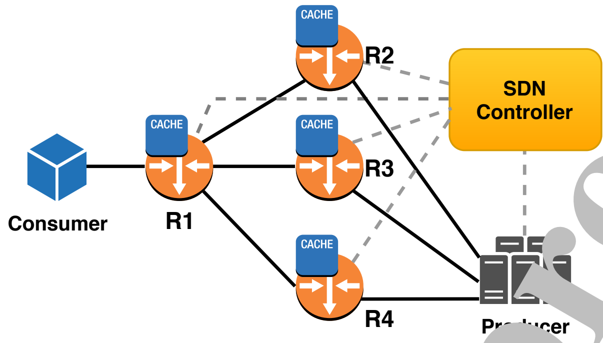


Figure 6: Network topology for cache management experiments.

PKS2, Clemson, Texas, Wisconsin, Vermont, and Hawaii. Therefore, this setup is also representative of a real-world WAN NDN network.

7.2. Experiments

In this section, we outline the various experiments to evaluate the performance of our multipath forwarding strategies. We evaluate NDN data transfer performance for different scenarios over our WAN test network as outlined in Table 3. The two datasets used were: i) 100MB file transfers, and ii) 1000 files of 8KB each. We design the following experiments (*E1* to *E5*) for our evaluations:

7.2.1. E1–Single Router, Single-Interest Pipeline

This is the default NDN strategy where the consumer retrieves all the requested data from a single router and/or a single producer.

7.2.2. E2–Single Router, Aggregate Interest Pipeline

This is a variation of the previous strategy *E1* with an increased Interest pipeline depth (P = 10 was used).

7.2.3. E3–Distributed Multipath (D-MP), Single-Interest Pipeline

In this case, we use three routers and with only one Interest per router. The list of routers is obtained using the communication between the consumer and the routers. The Interest distributed evenly between routers and based on Round-Robin (RR) technique.

7.2.4. E4–Distributed Multipath (D-MP) Pipeline

This is Similar to the previous case, but we use multiple Interests per router ($p = 10$). We evaluate two configurations for this pipeline: i) *E4a*: Round-robin with $p = 10$ per router, and ii) *E4b*: Ratio-partitioning based Interest pipeline distribution (a ratio of $50\%, 30\%, 20\%$ was used).

7.2.5. E5 & E6 –SDN-enabled Multipath (S-MP) Pipeline

This is Similar to E3 & E4, except the consumer will retrieve the list of routers which cache the data from the SDN controller. The optimum number (and list) of routers, and their associated pipeline depths are obtained from the SDN controller using Round-robin (RR) technique. We evaluate two case: i) *E5a*: S-MP Round-robin and ii) *E5b*: S-MP Ratio-partitioning similar to *E4*.

We evaluate the performance with/without *in-network* caching. For the *in-network* caching-enabled case, the CS at the routers will have cached the requested data. Therefore the Interest is not forwarded to the NDN producer. In the default NFD implementation, the Interest will be forwarded to the producer and only benefits from on-path caching. In our proposed architecture, the SDN will reconfigure the consumer to send Interest to off-path routers, which also host the requested data. Thus, we do not restrict data forwarding only to the on-path routers. This architecture reduces the latency for data retrieval, producer overheads, and avoids single-path congestion.

Table 5: Evaluation Parameters.

Test Datasets	Experiment Design		
	#Routers	Pipeline Size	Caching
100MB Files, and 8KB×1000 Files	1	1	w/ & w/o
	1	10	w/ & w/o
	3	1 per Router	w/ & w/o
	3	10 per Router	w/ & w/o
	3	5:3:2	w/ & w/o

7.2.6. Cache Management Experiments

We evaluate the cache management for large data transfers using large files with file sizes chose to exceed the default NDN content store capacity of 512 MB. We use three different file sizes namely, 600 MB, 800 MB and 1 GB to evaluate our proposed cache management architecture. We evaluate large file transfers for two use cases namely, with and without caching. Further, we also evaluate the data transfer performance when the prefetching feature is enabled for the above use cases.

8. Results and Discussion

In this section, we present the performance results of our proposed multipath forwarding strategies, cache management for large data transfer and demonstrate the benefits of our proposed prefetching feature.

8.1. Multipath Forwarding Strategies

The WAN data transfer performance of the proposed D-MP and the S-MP methods were evaluated on the GENI network testbed. The SDN controller and all NDN entities (i.e. consumers, routers, and producers) were placed on different InstaGENI sites and aggregated using layer-2 stitching over Internet2 AL2S. Two sets of WAN transfer performance results for two datasets are presented in Figure 7. For both datasets, we evaluate the transfer performance with i) *in-network* caching disabled, i.e. the requested data is not available in the routers' content store (CS), and the requested data is always fetched from the producer and then cached at the router(s); and ii) *in-network* caching enabled, i.e. the requested data is available on both *on-path* and *off-path* routers. All the results in this paper are computed with 95% confidence interval over five runs.

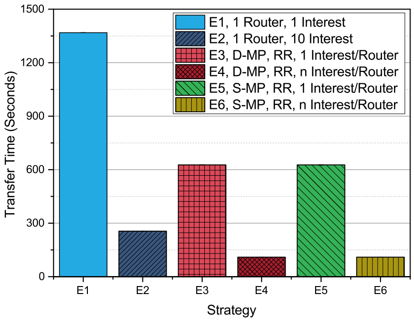
Figures 7a and 7b show the transfer performance results for the experiments listed in the Section 7.2 for the 100MB dataset with *in-network* caching disabled and enabled, respectively. Our D-MP approach performs 12.5x and 18.4x better than the default NDN implementation with *in-network* caching disabled and enabled respectively. In addition, the S-MP strategy shows performance gains of 12.6x and 18.5x with *in-network* caching disabled and enabled respectively.

Figure 7c shows the transfer performance for the second dataset (i.e., 1000×8KB files) with *in-network* caching disabled. We see that D-MP and S-MP approach perform 10.4x and 10.6x better than the default NDN implementation, respectively. In Figure 7d, with *in-network* caching enabled, we see further performance improvements, with D-MP and S-MP performing 12.5x and 12.6x better respectively.

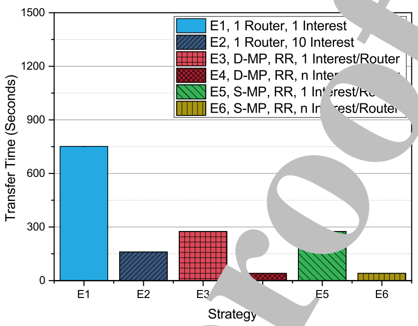
Comparing the two proposed approaches, we see that S-MP performs 0.8% and 0.54% better than the D-MP case for transferring the 10GB dataset. It performs 1.92% and 0.8% for transferring 1000×8KB files. The reason for that is that the S-MP only adds a small latency overhead to the transfer time. This is because the Interest packet is forwarded to the SDN controller, and the Consumer waits to receive the configuration update before initiating the connections with the appropriate routers. Furthermore, we note that this is a one-time cost and can be minimized by placing the SDN controller closer to the edge of the NDN networks. Thus, the S-MP approach scales predictably with an increasing number of Interests. While in the D-MP case, the Consumer needs to contact all routers in the network to build the configuration file, which will decrease the performance. The degradation in performance will increase for the D-MP case as the number of routers in the network increases, as shown in Figure 8.

To compare the D-MP strategy with the S-MP strategy, we increase the number of routers by adding another layer of routers to our testbed. The extra three routers are located on three different sites on the GENI testbed and are two hops away from the consumer. Figure 8 shows the communication overhead to build the configuration file for D-MP and S-MP strategies. We observe from the figure that the communication delay increases for the D-MP strategy as the number of routers increases. On the other hand, the S-MP overhead is consistent because it does not depend on the number of the routers but the SDN controller's location.

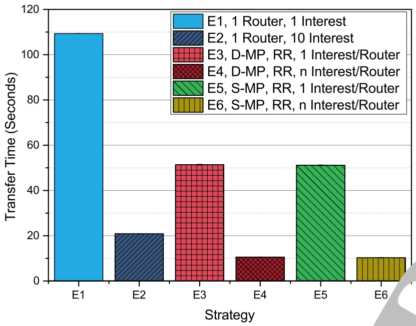
The WAN performance comparison with ratio partitioning strategies is shown



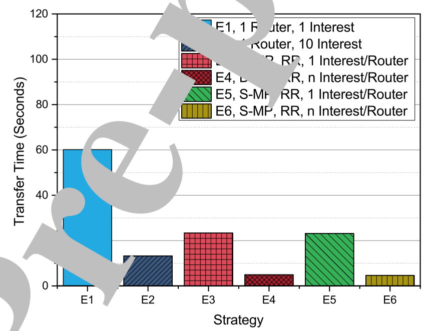
(a) 100MB files, No caching.



(b) 100MB files, Caching enabled.



(c) 8KB files, No caching.



(d) 8KB files, Caching enabled.

Figure 7: WAN Performance Evaluation of the D-MP and S-MP strategies for different datasets.

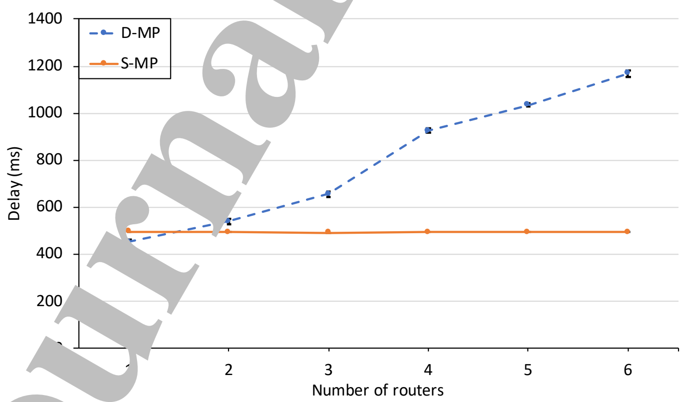
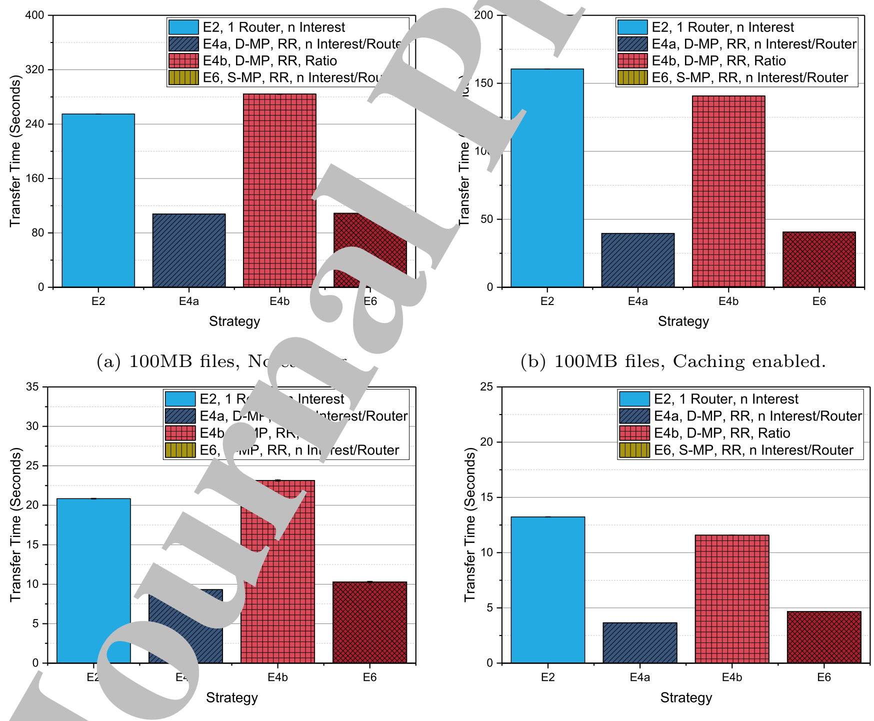


Figure 8: Comparison of D-MP and S-MP communication overheads.

in Figure 10. We evaluate the data transfer performance by comparing D-MP and S-MP strategies with the ratio partitioning technique described in Section 7.2. For our evaluation, we use the ratios of 50%, 30%, 20% for routers R1, R2, and R3, respectively. Figures 10a and 10b compares the data transfer performance for the multi-interest experiments for the 100MB dataset with in-network caching disabled and enabled, respectively. Further, the Figures 9c and 9d compares the data transfer performance for the multi-interest experiments for the 8k dataset with in-network caching disabled and enabled, respectively. From the figures, we observe that ratio partitioning exhibits performance that is similar to the single router n Interest pipelines. However, we note that with caching enabled, the ratio partitioning technique performs better than the single router, n Interest pipeline. Setting the ratios to 33% per router defaults to the round-robin case (E4a in the figure). Thus, with caching enabled, tuning the Interest pipeline ratios for each router is essential for optimal data transfer performance.



(a) 100MB files, No caching. (b) 100MB files, Caching enabled.

(c) 8KB files, No caching. (d) 8KB files, Caching enabled.

Figure 9: WAN Performance Comparison with Ratio Partitioning (Ratio: 50/30/20).

8.2. Cache Management for Large Data Transfers

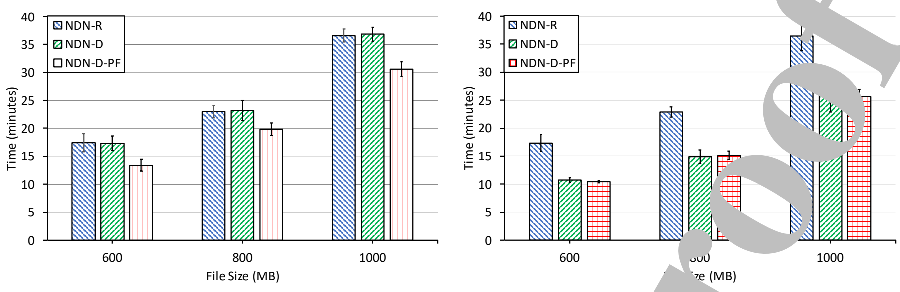
The WAN data transfer performance of the proposed architecture was tested on the GENI network platform. All nodes (i.e., controller, consumer, routers, and producer) were placed on different InstaGENI sites and aggregated using layer-2 stitching over Internet2 AL2S. Three different files (60MB, 500MB, 1GB) were used to compare the performance between the default NDN and our proposed architecture. For all files, we evaluate the transfer performance when i) The requested data is not available in the routers' CS and the requested data is always fetched from the producer and then cached at the router(s); and ii) The file request occurred after the previous step, i.e., the data might be available in the Router CS since a similar request has been executed earlier. We computed the results with 95% confidence interval.

In all experiments, we used the default NDN router CS (500MB). The replacement policy for the routers CS is least recently used (LRU). The files are segmented and converted into NDN packets. Each segment uses the default NDN segment size (8800 Byte). We set the interest pipeline depth to 50; the consumer will send 50 Interest simultaneously before receiving the corresponding data. We set a static value for the pipeline depth to increase the transfer performance. This value will be changed into a dynamic value based on the network/path conditions in the future work.

Table 4: Experiment type.

Experiment type	File location
NDN-R	Default NDN architecture
NDN-D	Proposed system architecture
NDN-D-PF	Proposed system architecture with Prefetch

Table 4 shows the different types of experiments that we used for the performance comparison. NDN-R represents the default NDN architecture with the default settings. NDN-D represents our proposed system architecture for cache management. NDN-D-PF represents the cache management with the prefetch



(a) File Not cached.

(b) File cached.

Figure 10: WAN Performance Evaluation of the distributed cache with/without prefetch. (a) when the file is not cached, and (b) when the file is cached.

feature enabled as explained in Section 3.3.3.

Figure 10a shows the transfer performance when the file is requested for the first time; in this scenario, the file is not cached in any router CS. Since in this work we are interested in caching the large file transfers, we run the experiments listed in Table 4 with three different file sizes; 600MB, 800MB, and 1GB. These file sizes are larger than the default router CS size of 500MB used in NDN. Figure 10a shows that the performance of NDN-R and NDN-D are similar since the file is not present in any routers' CS and the files are always requested from the NDN producer.

On the other hand, the NDN-D-PF shows a 13.5% – 23.6% performance improvement over other approaches. This performance gain is due to the file retrieval process in NDN-D-PF where the first part is requested normally through the path (Consumer -> R1 -> R2 -> producer) and simultaneously, the controller will direct routers R3 and R4 to prefetch the other parts of the file. Therefore, routers R3 and R4 satisfy the interests request for other file segments.

Figure 7b shows the transfer performance when the file is requested for the second time; in this case the file is already cached in the network (i.e., some router's CS). The NDN-R shows no benefit from the NDN router caching mechanism. This is due to the fact that file sizes are larger than the CS. Therefore, the cached file segments will be replaced with new file segments (similar to the example in Section 3. In the NDN-R case, all file segments will be requested from the NDN producer. On the other hand, NDN-D and NDN-PF show 28.1% – 38% performance gains. This is due to the NDN-D and NDN-PF approaches where the files are cached on multiple routers' CS and all file segments are served by the routers rather than the NDN producer.

Although our architecture focused on large dataset transfers, small file transfers will still follow the default NDN route (single path only). Small files are not split among several routers unless they are larger than the routers' CS. Our approach adds a small additional delay due to the communication with the controller. However, the delay is negligible as the controllers are typically one-hop away from the routers. In our testbed, the SDN controllers are one hop away from the NDN nodes. The total delay overhead for the end-to-end data transfer (for both datasets outlined in Table 3) is less than 1% of the total transfer time. For example, considering the 100MB dataset, the overhead is 1.9 seconds (comparing E4 and E6 in Figure 7a) which contributes an overhead of about 0.3% to the total data transfer time. Therefore, we focus evaluating the end-to-end performance for large files as the SDN controller performance does not adversely affect data transfer performance.

9. Conclusions

In this paper, we presented an architecture that uses centralized control with NDN to provide faster in-network access to large datasets. We use SDN to provide an intelligent and efficient solution for data distribution and retrieval across multiple NDN routers' caches. The SDN controller splits and distributes large data files to multiple NDN routers' content stores. Our proposed system architecture results in a performance gain of 28.1% - 38% compared to the current NDN architecture. Moreover, we developed a prefetch mechanism for the files that are already cached in the network, which further reduces the file transfer time.

We also proposed two novel approaches for Interest pipeline distribution to improve the performance of NDN data transfers. Our D-MP strategy provides up to 18.4x improvement in performance over the current NDN strategies. The S-MP forwarding strategy provides better flexibility in Interest distribution by creating a map of the current state of the NDN routers' content stores. It also provides an 18.5x performance improvement over existing NDN approaches. We evaluated our solutions on an at-scale research testbed to provide valuable insights into the WAN transfer performance of an NDN network. Extensive evaluations with both *in-network* caching enabled and disabled, show that the proposed solutions remarkably outperform the current alternatives. Finally, the S-MP solution provides a generalized framework for software-defined control of an NDN network. This solution can be easily extended to incorporate adaptive and intelligent decision-making strategies for Interest pipeline management.

In future work, we will focus on the cache placement problem. In the present scenario, we need to split and cache large files among multiple routers. Thus, choosing the best locality of the routers will surely improve file transfer performance. We will also study the effect of link bandwidths and how to avoid congested links in the NDN network.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Numbers OAC-1541442 and CNS-1817105. This work was completed using the Holland Computing Center of the University of Nebraska, which receives support from the Nebraska Research Initiative. The authors would like to thank Garhan Attebury, Holland Computing Center at UNL for his valuable support.

References

[1] M. Alhowaidi, D. Nadig, B. Ramamurthy, B. Bockelman, D. Swanson, Multipath Forwarding Strategies and SDN Control for Named Data Net-

working, in: 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2018, pp. 1–6.

[2] M. Alhowaidi, D. Nadig, B. Ramamurthy, Cache Management for Large Data Transfers in Named Data Networking using SDN, in: 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), 2019, pp. 1–6, iSSN: 2153-1684. doi:10.1109/ANTS47819.2019.9118137.

[3] C. Grandi, B. Bockelman, D. Bonacorsi, et al., CMS Distributed Computing Integration in the LHC sustained operations era, in: Journal of Physics: Conference Series, Vol. 331, IOP Publishing, 2011, p. 062032.

[4] I. Bird, Computing for the Large Hadron Collider, Annual Review of Nuclear and Particle Science 61 (2011) 99–118.

[5] L. Zhang, A. Afanasyev, et al., Named data networking, ACM SIGCOMM CCR 44 (3) (2014) 66–73.

[6] B. A. A. Nunes, M. Mendonca, et al., A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, IEEE Comm. Surveys Tutorials 16 (3) (2014) 1617–1634. doi:10.1109/SURV.2014.012214.00180.

[7] S. Gao, Y. Zeng, et al., Scalable area-based hierarchical control plane for software defined information centric networking, in: Intl. Conf. on Computer Communication and Networks (ICCCN), 2014, pp. 1–7. doi:10.1109/ICCCN.2014.6911839.

[8] E. Aubry, T. Silverston, I. Chrismen, Implementation and Evaluation of a Controller-Based Forwarding Scheme for NDN, in: Advanced Information Networking and Applications, 2017, pp. 144–151. doi:10.1109/AINA.2017.[illegible]

[9] M. Vahlenkamp, F. Schneider, D. Kutscher, J. Seedorf, Enabling Information Centric Networking in IP Networks Using SDN, in: SDN for Future

Networks and Services (SDN4FNS), 2013, pp. 1–6. doi:10.1109/SDN4FNS.2013.6702539.

[10] J. Rexford, C. Dovrolis, Future Internet architecture: clean-slate versus evolutionary research, Communications of the ACM 53 (9) (2010) 36–40.

[11] N. Feamster, J. Rexford, E. Zegura, The road to SDN: an intellectual history of programmable networks, ACM SIGCOMM Computer Communication Review 44 (2) (2014) 87–98.

[12] J. Li, R.-c. Xie, T. Huang, L. Sun, A novel forwarding and routing mechanism design in SDN-based NDN architecture, Frontiers of Information Technology & Electronic Engineering 19 (9) (2018) 1135–1150.

[13] T. Combe, W. Mallouli, T. Cholez, G. Doyen, B. Mathieu, E. Montes de Oca, An SDN and NFV Use Case: NDN Implementation and Security Monitoring, Springer International Publishing, Cham, 2017, pp. 299–321. doi:10.1007/978-3-319-64653-4_12.
URL https://doi.org/10.1007/978-3-319-64653-4_12

[14] H. L. Mai, M. Aouadj, G. Doyen, W. Mallouli, E. M. de Oca, O. Festor, Toward Content-Oriented Orchestration: SDN and NFV as Enabling Technologies for NDN, in: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, pp. 594–598.

[15] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, L. Veltri, Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed, Computer Networks 57 (16) (2013) 3207–3221.

[16] J. Torres, L. Ferraz, O. Duarte, Controller-based routing scheme for Named Data Network, Electrical Engineering Program, COPPE/UFRJ, Tech: Rep (201[illegible]).

[17] M. Amadeo, C. Campolo, G. Ruggeri, A. Molinaro, A. Iera, SDN-Managed Provisioning of Named Computing Services in Edge Infrastructures, IEEE

Transactions on Network and Service Management 16 (4) (2019) 146[illegible]–1478. doi:10.1109/TNSM.2019.2945497.

[18] N. El Houda BenYoussef, Y. Barouni, S. Khalfallah, J. B. H. Slama, K. Ben Driss, Mixing SDN and CCN for content-centric Qos aware smart grid architecture, in: 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), 2017, pp. 1–5. doi:10.1109/IWQoS.2017.7969139.

[19] H. Salah, T. Strufe, Comon: An architecture for coordinated caching and cache-aware routing in CCN, in: Consumer Communications and Networking Conference (CCNC), IEEE, 2015, pp. 663–670.

[20] H. K. Rath, B. Panigrahi, A. Simha, On Cooperative On-Path and Off-Path Caching Policy for Information Centric Networks (ICN), in: Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on, IEEE, 2016, pp. 842–849.

[21] Y. Xin, Y. Li, et al., Content aware multi-path forwarding strategy in Information Centric Networking, in: Computers and Communication (ISCC), 2016 IEEE Symposium on, IEEE, 2016, pp. 816–823.

[22] J. Wang, J. Ren, et al., A minimum cost cache management framework for information-centric networks with network coding, Computer Networks 110 (2016) 1–17.

[23] R. Chiocchetti, D. Perino, et al., Inform: a dynamic interest forwarding mechanism for information centric networking, in: Proc. 3rd ACM SIGCOMM workshop on ICN, ACM, 2013, pp. 9–14.

[24] S. Charpinel, C. A. S. Santos, A. B. Vieira, et al., SDCCN: A novel software defined content-centric networking approach, in: Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on, IEEE, 2016, pp. 87–94.

[25] M. Alhowaidi, B. Ramamurthy, et al., The Case for Using Content-Centric Networking for Distributing High-Energy Physics Software, in: ICDCS, 2017, pp. 2571–2572. doi:10.1109/ICDCS.2017.295.

[26] H. Lim, A. Ni, D. Kim, et al., NDN Construction for Big Science: Lessons Learned from Establishing a Testbed, IEEE Network 32 (6) (2018) 124–136. doi:10.1109/MNET.2018.1800088.

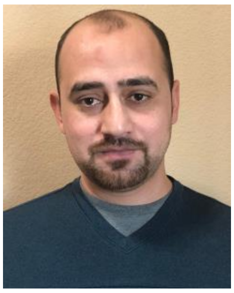
[27] H. Newman, A. Mughal, D. Kcira, et al., High Speed Scientific Data Transfers Using Software Defined Networking, in: Proceedings of the Second Workshop on Innovating the Network for Data-Intensive Science, INDIS '15, ACM, New York, NY, USA, 2015, pp. 2:1–2:9, event-place: Austin, Texas. doi:10.1145/2830318.2830320.

[28] V. Lehman, A. Gawande, et al., An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN, in: Intl. Sym. on Quality of Service (IWQoS), 2016, pp. 1–10. doi:10.1109/IWQoS.2016.7590394.

[29] J. Blomer, P. Buncic, R. Meusel, The CernVM file system, Tech. rep., Technical Report (2013).

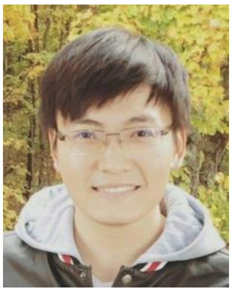
[30] S. Chatrchyan, et al., The CMS Experiment at the CERN LHC, JINST 3 (2008) S08004. doi:10.1088/1748-0221/3/08/S08004.

[31] M. Berman, J. S. Chase, L. Landweber, et al., GENI: A federated testbed for innovative network experiments, Computer Networks 61 (2014) 5 – 23, sI on Future Internet Testbeds - Part I. doi:http://dx.doi.org/10.1016/j.bjp.2013.12.037.



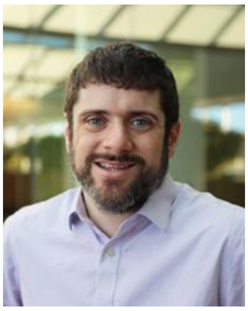
Mohammad Alhowaidi received his Ph.D. degree in Computer Engineering from the University of Nebraska-Lincoln. He has a master's degree in Computer Science and Engineering from Linkoping University, Sweden. He received his bachelor's degree in Computer Engineering from Jordan University of Science and Technology, Jordan. His research interests are in future internet architectures, software defined networking, optical networks, and resource allocation in cloud networks.

Deepak Nadig is currently a Ph.D. Student in the Department of Computer Science and Engineering at the University of Nebraska-Lincoln (UNL). He received the B.E. and M.Tech degrees from Visvesvaraya Technological University, India, in 2004 and 2007, respectively. He is also an IEEE certified Wireless Communications Professional (IEEE WCP). His research interests are in the areas of Software Defined Networks (SDN), Network Functions Virtualization (NFV) and Cybersecurity.



Boyang Hu is currently a Ph.D. student in Computer Science and Engineering at the University of Nebraska-Lincoln (UNL). He received the B.S. degrees from Beijing Jiaotong University, China, in 2011 and M.S. from the University of Maryland, College Park, US, in 2013. His research interests include Network Security, Software Defined Networks (SDN) and Network Functions Virtualization (NFV).

Byrav Ramamurthy is currently a Professor and former Graduate Chair in the Department of Computer Science and Engineering at the University of Nebraska-Lincoln (UNL). He is the author of the book "Design of Optical WDM Networks - LAN, MAN and WAN Architectures" and a co-author of the book "Secure Group Communications over Data Networks" published by Kluwer Academic Publishers/Springer in 2000 and 2004 respectively. He served as the Chair of the IEEE Communication Society's Optical Networking Technical Committee (ONTC) during 2009-2011. He served as the IEEE INFOCOM 2011 TPC Co-Chair. He is currently the Editor-in-Chief for the Springer Photonic Network Communications (PNET) journal. His research areas include optical and wireless networks, peer-to-peer networks for multimedia streaming, network security and telecommunications. His research work is supported by the U.S. National Science Foundation, U.S. Department of Energy, U.S. Department of Agriculture, NASA, AT&T Corporation, Agilent Tech., Ciena, HP and OPNET Inc.

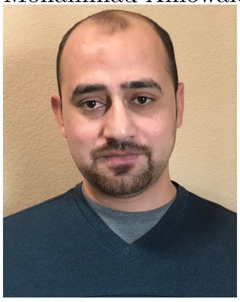


Brian Bockelman is currently an Associate Scientist at the Morgridge Institute for Research, University of Wisconsin-Madison. His research interests are in Research Computing and Distributed High-Throughput Computing (DHTC). For over a decade, he has worked with the Open Science Grid on issues in distributed high-throughput computing and now serves as the Technology Area Coordinator, leading the evolution of the technologies used by the OSG. Within Nebraska, Dr. Bockelman served as a key staff member of the Holland Computing Center (2008 – 2019) and as an Associate Research Professor in the Computer Science and Engineering department and worked on the CMS project, which hosts significant computing resources at the Holland Computing Center.

Brian Bockelman

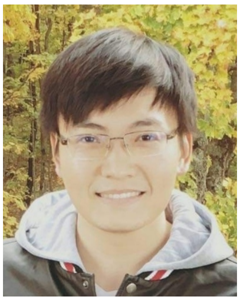
Deepak Nadig

Mohammad Alhowaidi



Byrav Ramamurthy

Boyang Hu



We do not have any conflicts of interest to disclose.