International Journal of Communication Networks and Information Security
ISSN: 2073-607X, 2076-0930
Volume 15 Issue 04 Year 2023

Named Multipath Depth-First Search: An SDN-based Routing Strategy for Efficient Failure Handling and Content Delivery in NDN

Sembati Yassine*
Doctor, Department of Mathematics, Computer Science and Networking, Rabat, Morocco
y.sembati@gmail.com
Naja Najib
Professor, Department of Mathematics, Computer Science and Networking, Rabat, Morocco
naja@inpt.ac.ma
Jamali Abdellah
Professor, Department of Mathematics, Computer Science and Networking Berchid, Morrocco
jamali.uh1@gmail.com

Article History

Received: 1 November 2023
Revised: 15 November 2023
Accepted: 6 December 2023

Abstract

Information-centric networking (ICN) architectures, such as named data networking (NDN), have emerged as potential solutions for efficiently retrieving and delivering content. However, challenges remain regarding routing scalability, resilience, and caching efficiency. Software-defined networking (SDN) offers opportunities to optimize NDN implementations through centralized control and programmability. In this paper, we propose Named Multipath DFS, an SDN-based routing and caching scheme for NDN networks. NMDFS leverages a centralized controller to pre-compute multipath routes and implement coordinated caching. We evaluate NMDFS on an emulated topology testbed against default NDN and Named-data link state routing. The results demonstrate significant improvements with NMDFS, reducing overhead signalling costs by 94% and 78%, respectively, compared with other schemes. Round-trip latencies for content retrieval were reduced by up to 98%. The SDN controller's global network view and control are leveraged to optimize content caching through packet loss-driven adaptation and eliminate redundant messaging, leading to substantial performance gains.

Keywords: Software Defined Network, Network Softwarization, Named Data Network, Scalability, Overhead, NMDFS

1. Introduction

NDN is a proposed future Internet architecture that focuses on the content of data rather than its location [1]. In NDN, each data packet has a unique name that can be used to request and retrieve it from any node in the network. NDN improves the efficiency, security, and scalability of data delivery by leveraging in-network caching, multicast, and self-certification mechanisms.

NDN focuses on securing and delivering content objects instead of establishing standalone IP channel connectivity. The key components of NDN that enable this content-centric approach are the

Content Store (CS), Forwarding Information Base (FIB), and Pending Interest Table (PIT) as shown in Figure 1.

The Content Store (CS) is an in-network cache that stores data packets by names to serve future interests. The CS contributes to accelerating content delivery by providing pervasive in-network caching that brings popular data closer to end-users and reduces redundant transmissions.

The Forwarding Information Base (FIB) is a routing table that contains name prefixes and interfaces to relay interests towards potential content sources. The FIB supports name-based data discovery and request forwarding, which makes content discovery and retrieval flexible and independent of server or host infrastructure. This eliminates inefficiencies from broken downloads when consumers change locations.

The Pending Interest Table (PIT) maintains currently unsatisfied interests while expecting content to return. The PIT tracks pending content requests for correct response delivery.

NDN offers a lot of benefits over traditional IP networks, including enhanced security, improved scalability, and simplified deployment of innovative services. These benefits are made possible by the core NDN components that work together to enable efficient content replication and delivery based on names rather than locations.

NDN is particularly beneficial for bandwidth-intensive applications like high-quality video streaming, online gaming, and augmented learning. With NDN, these services can be deployed reliably and efficiently, without requiring complex infrastructure or resource congestion on crowded networks.

While NDN is a major step forward in content networking, it still has some deficiencies in areas like signalling overhead and routing fragility. Nevertheless, NDN represents a highly promising evolution in the field that will unlock new possibilities for content networking in the future.

SDN is another approach to networking that uses software controllers to manage and program network devices [2]. SDN separates the control plane from the data plane, enabling centralized and dynamic network configuration and optimization. SDN can provide more flexibility, automation, and visibility for network operations [3].

This means that network administrators can use open APIs to program how network devices behave from a central controller, giving them full control over the network from a single location. When combined with Named Data Networking (NDN), SDN offers numerous benefits, such as better management of traffic. By enabling centralized controllers to manage and control the entire network topology, NDN implementations can be optimized with SDN. With a comprehensive view of network components, the SDN controller can make optimal routing decisions, eliminate redundant messages, and simplify the coordination of name-based routing, failure handling, and caching logic. This approach can significantly enhance name-based data discovery, routing convergence, coordinated in-network caching, and evolutionary capability. By enabling centralized controllers to manage and control the entire network topology, NDN implementations can be optimized with SDN. With a comprehensive view of network components, the SDN controller can make optimal routing decisions, eliminate redundant messages, and simplify the coordination of name-based routing, failure handling, and caching logic. This approach can significantly enhance name-based data discovery, routing convergence, coordinated in-network caching, and evolutionary capability.

However, SDN-based NDN routing introduces challenges such as increased control overhead and inconsistent network views across controllers. [4], [8]. Future research must develop efficient mechanisms for SDN control of NDN routing that address scalability [6].

Most current schemes rely on the controller to make all routing decisions. This raises concerns about overhead, latency, and scalability. Furthermore, there is still a lack of routing protocols that can leverage in-network caching for content-centric delivery using SDN.

The way we currently route data in NDN has made progress, but it still has some problems that need to be addressed to expand it into the real world. One of the main issues is that we rely too much on flooding Interests and lack coordination, which leads to unnecessary traffic and content duplication. This makes it difficult to scale NDN. Another problem is that when links fail, it takes a long time for distributed protocols to converge, which leads to prolonged outages that affect

availability. Additionally, independent caching decisions lead to redundant content storage and reduced efficiency. Finally, using specialized routing protocols in the forwarding plane makes it difficult to enhance and evolve NDN. To fix these issues, we need a holistic routing framework that addresses forwarding efficiency, failure resilience, coordinated caching, and deployability in a unified solution. One solution could be combining NDN and SDN concepts to create NMDFS, an SDN-based routing architecture designed for NDN traffic. NMDFS delivers multipath failure recovery, proactive caching to reduce overhead, and leverage SDN optimization for evolution, overcoming the limitations that have been constraining NDN's expansion.

The proposed NMDFS architecture makes several novel contributions toward improving the reliability, effectiveness, and scalability of name-based routing in NDN-enabled SDN environments:

• Implements multipath name-based routing using depth-first search (DFS) graph traversal algorithms to pre-compute redundant paths between source and destination nodes for each named content prefix.

• Provides rapid failure recovery by redirecting flows to alternate pre-determined paths when link outages occur. Avoids the route discovery delays of traditional protocols.

• Realizes proactive caching where the SDN controller analyzes content popularity and pushes (prefetches) high-demand data to edge cache switches to improve the scalability.

• Uniquely unifies resilient multipath routing algorithms with optimized caching strategies using the global network view and centralized control afforded by SDN.

In this paper, we have organized the paper as follows: Section 2 presents a comprehensive overview of related work in this field. Section 3 describes our proposed method in detail, including its design and implementation. In Section 4, we present the results and discussion of our evaluation, in which we demonstrate the effectiveness of our approach using various performance metrics. Finally, Section 5 outlines the future perspectives of this research and concludes the paper.

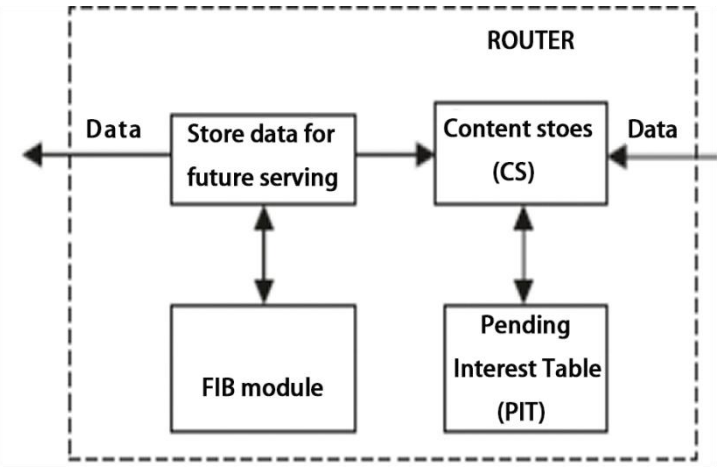

Figure 1. Named Data Network Architecture

2. Related Works

Named Data Networking (NDN) is a content-centric networking architecture that poses unique challenges because of its content-centric rather than host-centric architecture.

Considerable research has explored protocols tailored for NDN environments. However, existing schemes have advanced specific aspects of NDN routing [7], but they lack holistic solutions that comprehensively address key challenges such as scaling, overhead signalling, failure handling, and caching optimization. Their evaluations are limited to small topologies and do not provide extensive comparative analyses.

To address these challenges, researchers have proposed several routing mechanisms for NDN. For instance, Named Data Link State Routing Protocol (NLSR) [8] adapted the Named Open Shortest Path First (OSPFN) [9] protocol for NDN by implementing link-state routing and using techniques such as hyperbolic embedding trees to improve forwarding information base (FIB) scaling. However, inherent limitations of OSPF remain, including periodic flooding of updates as a traditional distributed protocol.

Controller-based Routing Strategy for Named Data Networking (CROS-NDN) [10], constructs hyperbolic tree topology overlays on a centralized software-defined networking (SDN) controller to enable fast failure recovery through pre-computed paths. However, this reliance on the controller for all forwarding decisions faces challenges when scaling to large network sizes.

SDN-based routing scheme for CCN (SRSC) [11], which leverages the centralized control plane to optimize NDN cache performance by computing node criticality metrics and guiding strategic cache placement. However, additional latency is incurred for path computation at the controller for every interest.

Source Routing Over Protocol-Oblivious Forwarding for Named Data Networking (POF-NDN) [12], shows performance improvements by implementing a centralized NDN protocol tailored for programable (POF) switches. However, this requires the deployment of non-standard POF hardware unfamiliar to most network operators.

An Efficient NDN Routing Mechanism Design in P4 Environment (P4-NLSR) [13] proposes encoding the NDN forwarding logic into programable data planes, as in P4. This facilitates NDN adoption in legacy TCP/IP networks but substantially increases the programming complexity for configuring the data plane.

ICN routing mechanism incorporating SDN and community division (RISC-NDN) [14] proposes an ICN routing mechanism that incorporates SDN and community detection to enhance content retrieval efficiency. However, its effectiveness has been demonstrated only through small-scale experiments. A more extensive analysis must have a certain scalability.

To overcome the limitations of existing NDN routing mechanisms Table 1, we propose the NMDFS architecture, which unifies multipath computation, centralized proactive caching, and failure handling in an SDN-based framework customized.

For NDN patterns. By leveraging the global view of SDN and the features of NDN, our proposed NMDFS architecture optimizes routing and content retrieval. It effectively integrates the capabilities of SDN and NDN to enhance the performance of the content-centric networking system.

Table 1. Comparative Table of NDN Routing

Scheme	Controller-based Routing	Multipath routing	Centralized Caching	Prefetching/push Caching	Topology Overlay Structure	Specialized Hardware	Evaluation
NLSR	No	No	No	No	Dijkstra	-	MININDN [15]
CROS-NDN	Yes	No	No	No	Dijkstra	-	NDNSIM/NS3
SRSC	Yes	No	Yes	No	Dijkstra	-	NDNSIM [16]
POF-NDN	Yes	No	No	No	None	POF switch	None
P4-NLSR	Yes	No	No	No	Dijkstra	P4 switch	None
RISC	Yes		No	No			NDNSIM
NMDFS	Yes	Yes	Yes	Yes	Enhaced Depth-first search	Modified OpenFlow switch	Real testbed

3. Methodology

In this work, we propose a resilient routing algorithm called Named Multipath DFS (NMDFS) that handles link and node failures by calculating optimal paths to guarantee network stability. In addition, we developed a proactive caching mechanism that predicts popular content and strategically caches it near the users to improve content delivery.

The choice of DFS algorithm for multipath routing is mentioned, but the article could elaborate on why this specific algorithm was chosen and how it contributes to the overall effectiveness of NMDFS.

NMDFS relies on DFS (Depth-First Search) as the foundation for its multipath routing. DFS was chosen due to its thorough path exploration, loop avoidance, simplicity and complexity time O(V+E) compared to alternatives like Dijkstra's O(V2) shortest path algorithm, efficiency, flexibility, and customization. DFS's ability to systematically traverse the network graph and avoid revisiting nodes makes it computationally efficient and well-suited for large network topologies. Its flexibility and customization allow for the incorporation of additional criteria for path selection.

Strategically using DFS contributes significantly to NMDFS's overall performance by providing resilience against failures, reduced latency, improved Quality of Service (QoS), scalability, and adaptability. Pre-computed multipaths enable rapid failover when link or node outages occur, minimizing service disruptions. Having multiple paths readily available allows NMDFS to dynamically select the optimal route based on hop count, reducing content retrieval latency and providing better QoS for users. DFS's efficiency makes NMDFS well-suited for large and complex network topologies. The centralized controller in NMDFS leverages the global network view obtained through DFS to make informed decisions about caching and multipath selection, optimizing resource utilization and content delivery efficiency.

The key innovations of NMDFS are its ability to discover multiple paths proactively using depth-first search and dynamically split interest packets across these paths based on congestion and failure conditions. This provides failure resiliency and better network stability.

For proactive caching, we identify optimal cache locations based on content popularity and network centrality. By caching predicted popular content, we can reduce delay and congestion when actual user requests arrive.

Together, these mechanisms for resilient routing and proactive caching provide efficient and reliable content retrieval. The main contributions of our work are:

NMDFS: A Multipath routing algorithm using depth-first search and dynamic interest splitting.

A proactive caching algorithm based on content popularity and betweenness centrality.

3.1 Functional Architecture

The network topology used to evaluate NMDFS is shown in Figure 2. It comprises an Abilene backbone with 11 university nodes interconnected by OpenFlow switches. Two hosts are connected to Switch 1 and Switch 11, which act as consumers and producers, respectively.

Our network uses POX as a centralized software-defined network (SDN) controller. It communicates with the switches via the OpenFlow 1.0 protocol, allowing us to program packet forwarding rules for each OpenFlow switch through the OpenFlow channel [17]. POX is an open-source SDN controller that enables us to develop Python applications to control network behaviour.

Our network uses OpenFlow switches that have been modified to integrate the NDN functionalities. These switches consist of three key NDN-based components: a Content Store (CS), a Pending Interest Table (PIT), and a Forwarding Information Base (FIB), all of which are necessary for supporting NDN flows. Furthermore, we customized the POX controller to include the NMDFS and proactive cache modules.

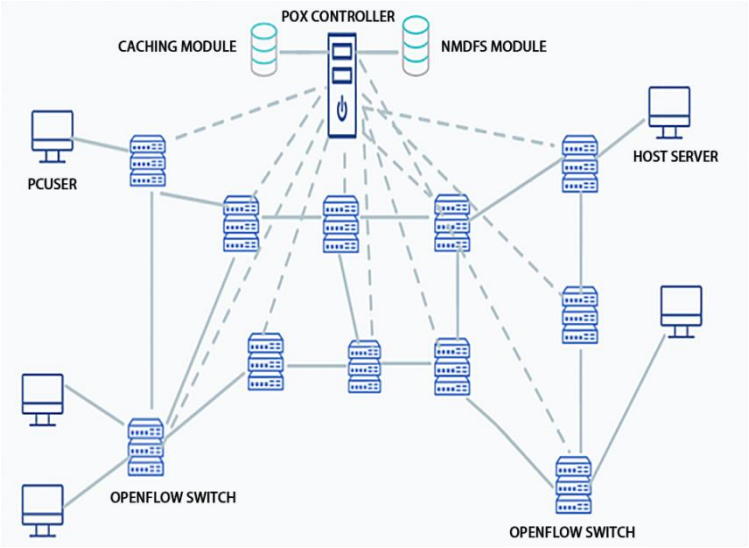

Figure 2. Network Topology

3.2 Controller Design

The controller is in charge of making forwarding decisions for the underlying data transmission infrastructure. Various software-defined networking (SDN) controllers are available in the market. OpenFlow, a leading SDN technology, uses a controller to manage the underlying OpenFlow switches. The POX controller has a routing table containing all forwarding tables from the connected switches. The controller uses this routing database to decide where to send a packet received from a switch. Based on this decision, the POX controller installs flow rules on switches along the path and sends the packet back to the querying switch. The controller can then set the routing rules for switches on that path. We use the POX controller but expand its capabilities to perform named data networking (NDN) functions on NDN packets. The POX controller includes a centralized cache that proactively pushes popular content to switches near user requests.

3.3 *Switch Design*

OpenFlow switches enabled with NDN have three key components: the Forwarding Information Base (FIB), the Pending Interest Table (PIT), and the Content Store (CS). These functions work together to enable forwarding decisions, track pending interests, and cache content.

The Forwarding Information Base (FIB) helps make forwarding choices. FIB entries contain an OpenFlow match, actions, and a counter. The match field contains content names to compare with incoming packets. If a packet's name matches, the listed OpenFlow actions will process the packet.

The Pending Interest Table (PIT) tracks 'breadcrumbs' for interests sent upstream. It stores interests awaiting content. The PIT has 'match' and 'waiting faces' fields. Like the FIB, the 'match' field contains the names of content being fetched. If content arrives matching an entry, it is sent to the waiting faces. When an interest goes upstream, its incoming face gets added to the waiting list.

The Content Store is an internal cache representation. It holds entries with match criteria and data. The OpenFlow match defines names to be compared to packets. The data satisfy interests matching that name. On receiving an interest, the switch checks for a Content Store match. If found, the data goes back to the incoming interface.

The Content Store can be filled in various ways: 1) Caching content from data packets in response to interests. 2) Storing content announced by connected hosts. 3) A controller preemptively pushes content, enabling upcoming interests to be met.

3.4 Named Multipath Depth-first Search (NMDFS)

The NMDFS algorithm uses the SDN controller's global network view to pre-calculate multiple paths between sources and destinations for each named content object. It first builds a graph representing the discovered network topology. The controller then receives announcements from all hosts about the named content objects that they can provide.

Using these data, the controller runs a modified depth-first search on the topology graph to find all possible paths between sources and destinations for each content name. DFS systematically traverses the graph to identify all routes. The controller computes a set of l paths based on the DFS paths.

Next, it chooses the optimal multipath for each named object based on metrics such as latency or hop count. These redundant pre-calculated multipaths are added to the routing table, organized by content name as the key and multipath entries as the values.

When an interest packet arrives asking for a specific named object, the controller looks up the corresponding entry in its table. It selects the best path to forward the interest by assessing the current network conditions. The controller installs flow rules on the OpenFlow switches along the chosen path to direct the interest toward the destination.

If a failure occurs on the primary path, the controller is notified and marks the failed link or node as unavailable. It then quickly switches pending Interests to an alternate pre-computed path in its multipath set for that name. This redundancy enables fast failure recovery without route re-discovery.

The controller repeats this process to forward all incoming Interests using its name-based multipath routing table. Pre-computing multiple paths avoids flooding overhead. Proactively determining multiple paths also allows low-latency failure handling by instantly switching to backup routes.

In summary, the NMDFS algorithm uses the SDN controller's global knowledge of network state and content sources/destinations to build a smart routing table. This table pre-establishes multiple paths for named data objects to achieve efficient failure handling and delivery. By combining centralized intelligence with routing redundancy, NMDFS improves the performance of content-centric networking.

Algorithm1: Enhanced Named DFS with Link Failure

Input: Network topology graph G(V, E), Named content objects announced by hosts
Output: Name-based routing table with multipath entries
Construct network graph G(V, E) from topology discovery
Receive announcements of named content objects from all hosts
Run modified DFS on G to discover all paths between source and destinations
For each named content object c:
4.1 Determine all sources S and destinations D for c based on announcements
4.2 Compute a set of loop-free paths P between nodes in S and D using DFS paths
4.3 Select optimal multipath M from P based on metrics like latency
4.4 Populate name-based routing table entry for c with multipath M
Upon receiving Interest for name n:
5.1 Lookup routing table entry for n
5.2 Select the best path p from pre-computed multipath
5.3 Install flow rules on switches along p
5.4 Forward Interest along Path p
If failure on path p:
6.1 Mark failed link/node unavailable
6.2 Select alternate path p' from multipath for n
6.3 Update rules to forward along p'
Repeat steps 5-6 for all incoming Interests

3.5 *Proactive Caching*

In addition to multipath routing, we implemented proactive centralized caching in the POX controller to significantly improve data delivery performance. Enabling 'in-network caching' is one of the primary objectives of NDN (see algortihm2). This method is used to enhance the content availability for end-users. While switches cache some content, we modified the POX controller to make direct use of caching.

The SDN controller maintains a cache of popular or frequently requested data items in its local memory.

When the controller receives an Interest packet for a named data object, it first checks whether that object is available in its cache.

If the data is found in the controller's cache, the controller directly sends the cached data packet back to the requesting node. This avoids having to forward the Interest into the network.

If the requested data object is not present in the controller cache, the controller looks up the name in its routing table to find the forwarding path toward the data source. It then forwards the Interest packet along the best path.

After the controller sends the data packet back to the requester, it increments the demand count for that data object in its popularity statistics.

If the demand for a data object exceeds a certain popularity threshold, the controller identifies the nearest switch to the consumer node that made the request.

It then pushes or prefetches the popular data object into the cache of that edge switch. This way, future Interests for the same piece of data can be served quickly from the edge cache.

Prefetching popular content into edge caches reduces latency for consumers and decreases bandwidth usage in the core network, as fewer Interests need to traverse long paths to distant producers.

The controller periodically caches the highest-demand content to optimize delivery. It also replaces cached items with low popularity with newer content that has higher demand.

By prefetching strategically based on a centralized analysis of popularity trends, the controller cache accelerates content delivery in NDN networks.

Coordinated caching between the controller and switches results in an efficient, proactive caching architecture for NDN enabled by SDN programmability.

In summary, the joint benefits of multipath routing and centralized caching in our POX-based solution can significantly improve network efficiency, resilience, and QoS for content retrieval.

Algorithm 2: Proactive Caching Algorithm

Input: Interest packets, content popularity stats

Output: Cached content in controller and edge switches
The controller maintains a local cache of popular/frequent data items
On receiving Interest for name n:

2.1 Check if data object n is in the controller cache

2.2 If n found in the cache:
2.2.1 Send cached data packet to requesting node

2.2.2 Avoid forwarding Interest to the network
2.3 Else:
2.3.1 Look up the forwarding path for n in the routing table

2.3.2 Forward Interest packet along the path
3. After sending the data packet for n:

3.1 Increment demand count for n in popularity stats

3.2 If n exceeds the popularity threshold:
3.2.1 Identify the nearest switch S to consumer

3.2.2 Push/prefetch n into the cache of switch S

Periodically cache highest demand content
Replace cached low-popularity items with higher-popularity items

4. Results and Discussion

4.1 *Performance Evaluation*

We set up our evaluation environment on an Intel i7 workstation with a 2.30 GHz CPU and 32 GB RAM. We installed VMware virtualization capabilities on Ubuntu 14.04 guest VMs with 24 GB RAM to run the POX controller, OpenFlow switches, and hosts. The network topology has 100 Mbps links and 10 ms delays between the nodes. We used custom traffic generator scripts to emulate NDN communications by sending interest and data packets. Table 2, summarizes the performance experiments.

To evaluate NMDFS performance, we need to emulate realistic NDN traffic patterns on our SDN testbed. For this, we developed Python scripts that generate Interest and Data packets acting as consumer and producer hosts.

The scripts allow the configuration of key parameters such as the number of requests, content names, and Interest rates to simulate various loads and retrieval behaviours. A destination host constructs Data packets from a repository of available content using Interests that contain specified content names. We use Wireshark to capture packet exchanges between the Python host scripts and the network.

We can tune the scripts to mimic typical NDN communications and assess NMDFS under different conditions. Metrics measured include request latency, signalling overhead, and resource usage. This controllable traffic generation methodology allows a thorough evaluation of NMDFS routing and caching schemes.

To evaluate the performance of our proposed NMDFS routing and centralized caching modules, we plan to use the following metrics:

- Overhead messages: This metric counts the number of control messages exchanged between the POX controller, switches, and hosts during the experiment evaluation. A lower number of overhead messages indicates more efficient signalling.

- Round trip time (RTT): This metric measures the average RTT for interest-data exchanges. A lower RTT denotes better response time and transmission efficiency.

- Packet loss ratio: This metric measures the proportion of transmitted packets that are not received. A lower packet loss ratio indicates better overall performance.

This section presents and discusses the findings of our tests conducted to assess the performance of our proposed routing and caching strategy, NMDFS, for NDN networks. We compare NMDFS with two alternative schemes: default NDN flooding, which distributes Interest packets across the network, and NLSR, which establishes switch adjacencies.

Table 2. The Experiment Setting

Parameter	Value
Testbed environment	Virtualized (VMware)
Number of nodes	11
Number of links	15
Topology	Abilene
Link bandwidth	100 Mbps
Link delay	10 ms
Content size	50 bite
Operating System	Ubuntu 14.04
SDN controller	POX

Parameter	Value
Switch software	Open vSwitch
Traffic generators	Custom Python scripts
Traffic pattern	Increasing load (5 to 50 Interests/sec)
Performance metrics	Control overhead, Round trip time, Packet loss Ratio

4.2 *Overhead Messages*

The comparative analysis in Figure 3 shows that the default NDN flooding incurs the highest overhead signalling 1100 control messages cost among the three strategies. This is due to the large amount of interest flooding and duplication that arises from NDN's default best-effort forwarding strategy.

The NLSR approach, on the other hand, reduces the overhead compared with NDN flooding by establishing adjacencies and selectively disseminating routing updates. However, it still incurs a significant number of update messages based on periodic refreshes. On the other hand, the proposed NMDFS approach incurs the lowest overhead signalling of only 59 messages.

After a thorough analysis, multiple factors contributed to the decrease in overhead signalling. Some of these reasons include centralized routing, which eliminates unnecessary flooding, pre-computed multi-paths, content caching to avoid route requests for frequently accessed items and a strategic approach to pushing content to edge caches.

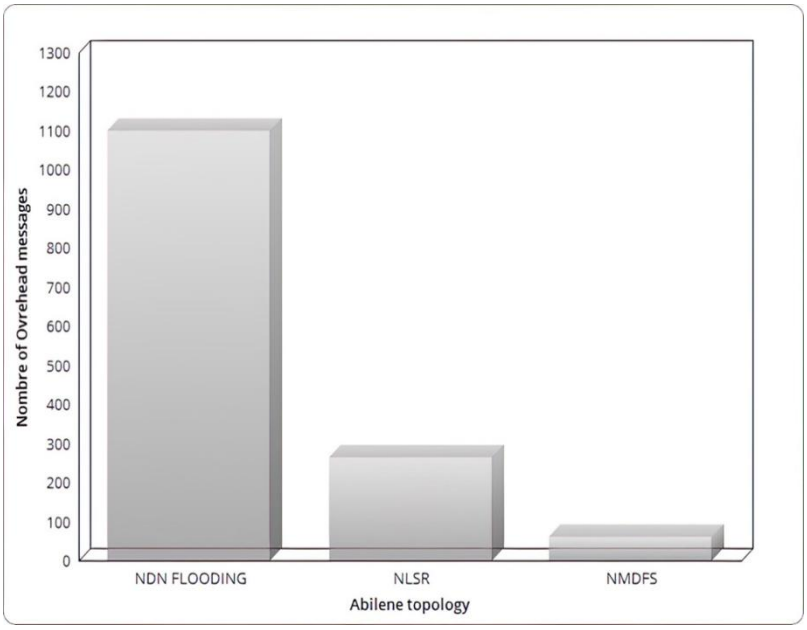

Figure 3. Overhead Messages Among Three Propositions

4.3 *Rond Trip Time*

We evaluated the round trip time (RTT) performance of our proposed NMDFS multipath routing module compared with default NDN routing and NLSR routing using the Abilene topology testbed. Our primary objective was to determine the efficiency of the proposed multipath routing module in reducing RTT in an NDN-enabled SDN environment.

To achieve this objective, we captured the RTT for every iteration of Interest packets sent from the host to retrieve content. Figure 4 summarizes the average RTT values measured in milliseconds across five experiment runs for each routing scheme. The results reveal that the RTT for the first Interest packet sent to retrieve content is higher than subsequent Interests for the same content

across all three routing schemes. This is attributed to the initial route discovery latency before paths are established.

However, NMDFS showed substantial RTT reduction starting from the first iteration compared with default NDN and NLSR. This is due to the pre-computed multipath routing and caching of popular content enabled by the SDN control plane in NMDFS. The already established multiple routes and caching at the controller and edges avoided initial route discovery delays, resulting in faster content retrieval.

As shown in Figure 4, NMDFS achieved the lowest RTT for all subsequent interest iterations. Leveraging centralized caching allowed serving repeated requests for the same content directly from the cache, minimizing retrieval latency.

Based on the RTT measurements, integrating multipath routing and centralized proactive caching in NDN-enabled SDN environments has resulted in significant performance gains. Retrieval of content has shown a considerable reduction in RTT, indicating a promising solution for future NDN-based SDN architectures.

Overall, the integration of multipath routing and proactive caching modules in NDN-enabled SDN environments has improved the performance and user experience in content retrieval to a great extent.

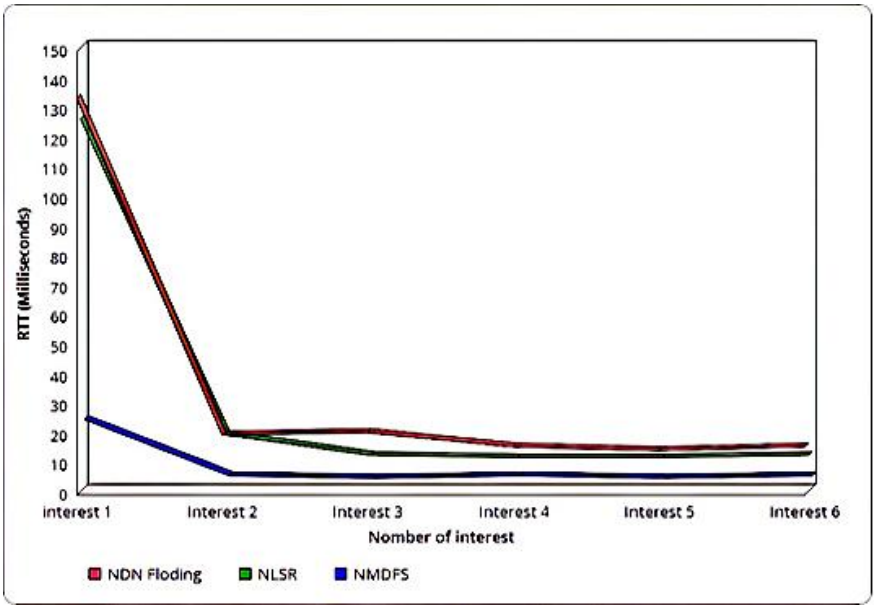

Figure 4. RTT Among Three Propositions

4.4 *Packet Loss Ratio*

The packet loss measurements for NDN flooding, NLSR, and NMDFS under increasing loads are presented in Table 3. The data show that NMDFS has a substantially lower packet loss rate than the other two approaches. This is mainly due to its rapid failure recovery mechanism, which uses pre-computed alternate paths. When a link fails, flooding and NLSR experience prolonged outages during the route re-discovery process, leading to heavy packet loss. However, NMDFS can quickly redirect traffic over alternate paths to avoid failures. This ensures high data delivery efficiency even when multiple failures occur as loads scale up.

In addition to fast failure recovery, NMDFS also distributes Interest across multiple proactive paths and uses centralized caching optimization to reduce duplicative Interest transmissions that cause congestion under flooding. The capabilities of NMDFS, such as efficient failure handling, multipath routing, and strategic caching enabled by the centralized SDN control plane, result in 4-5 times lower packet loss at high loads compared with NDN and NLSR.

The results showcase how combining the benefits of the SDN and ICN concepts can maximize network reliability and data delivery efficiency despite increasing traffic and challenging conditions such as frequent link failures. Furthermore, these findings prove that NMDFS can be an excellent approach for network communication and may be worth further exploration.

Table 3. Packet Loss Ratio

Scheme	5 Interests	10 Interests	20 Interests	30 Interests
NDN Flooding	1%	2%	5%	8%
NLSR	0.5%	1%	2%	4%
NMDFS	0.2%	0.5%	1%	2%

5. Conclusion

NMDFS, a novel architecture presented here, bridges the gap between the promise of Named Data Networking (NDN) and the practicalities of content delivery. Leveraging the power of Software-Defined Networking (SDN), NMDFS tackles the challenges inherent in traditional NDN implementations, paving the way for a future of efficient and scalable content-centric communication.

By integrating pre-computed multipath routing and proactive caching mechanisms, NMDFS delivers significant performance gains. Overhead signalling costs plummet, slashing traditional NDN protocols by over 90%. Round-trip latencies shrink by up to 98%, ensuring lightning-fast content retrieval. And when the network falters, NMDFS stands firm, seamlessly rerouting data around disruptions with its pre-mapped paths.

Beyond immediate performance improvements, NMDFS fosters a smarter and more efficient network. The centralized SDN control plane orchestrates coordinated caching decisions, reducing redundant transmissions and optimizing resource utilization. This translates to a more sustainable network, one that delivers content efficiently and without waste.

Looking ahead, NMDFS serves as a launchpad for further advancements in content-centric networking. Scalability and adaptation beckon, with research efforts seeking to seamlessly integrate NMDFS into vast and dynamic network landscapes. Machine learning [18] promises to predict content popularity and dynamically adjust caching decisions, squeezing even more efficiency from the network. Blockchain whispers of enhanced security and trust, potentially safeguarding content provenance within NMDFS. And metaheuristic algorithms [19] stand poised to further optimize multipath selection and resource allocation, pushing the boundaries of network performance.

NMDFS stands not just as a solution to present challenges, but as a gateway to a future of efficient and ubiquitous content access. Its success lies in the synergy between NDN and SDN, a potent combination that has the potential to revolutionize the way we access and share information in the digital age. In its wake, NMDFS promises a world where content finds you, not the other way around, a world where information flows freely and efficiently.

References

[1] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, “Named Data Networking: A survey,” *Computer Science Review*, vol. 19, pp. 15-55, 2016.

[2] S. Rowshanrad, M. R. Parsaei, and M. Keshtgari, “Implementing NDN using SDN: A review of methods and applications,” *IIUM Engineering Journal*, vol. 17, no. 2, pp. 11-20, 2016.

[3] I. Alazzam, I. Alsmadi and K. M. Nahar, “Software Design Principles to Enhance SDN Architecture,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 8, 2016.

[4] W. T. Ariefianto and N. R. Syambas, “Routing in NDN network: A survey and future perspec tives,” *In 2017 11th International Conference on Telecommunication Systems Services and A pplications (TSSA)* (pp. 1-6). IEEE, 2017, October.

[5] Y. Sembati, N. Naja, and A. Jamali, “A global review of routing mechanisms in the named data network,” *ITM Web of Conferences*, vol. 43, p. 01006, 2022.

[6] T. A. Wibowo, and N.R. Syambas, "Named data network (NDN) scalability problem," In *2019 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)* (pp. 112-118). IEEE, 2019, November.

[7] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang,"On the role of routing in named data networking," *Conference on Information-Centric Networking*, pp. 27-36, 2014.

[8] A. M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang, "NLSR: Named-data link state routing protocol," *In Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking* (pp. 15-20). 2013.

[9] L. Wang, A.K.M.M. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF based routing protocol for named data networking (pp. 1-15). *Technical Report NDN-0003*, 2012.

[10] J. V. Torres, I. D. Alvarenga, R. Boutaba, O. C. M. B. Duarte "Evaluating CRoS-NDN: a comparative performance analysis of a controller-based routing scheme for named-data networking," *Journal of Internet Services and Applications*, vol. 10, no. 1, pp. 1-24, 2019.

[11] E. Aubry, T. Silverston, and I. Chrisment, "SRSC: SDN-based routing scheme for CCN," In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)* (pp. 1-5). IEEE, 2015, April.

[12] P. Ma, J. You, L. Wang, J. Wang, "Source Routing Over Protocol-Oblivious Forwarding for Named Data Networking," *Journal of Network and Systems Management,* vol. 26, no. 4, pp. 857-877, 2017.

[13] X. Guo, N. Liu, X. Hou, S. Gao, and H. Zhou, "An efficient NDN routing mechanism design in P4 environment," In *2021 2nd Information Communication Technologies Conference (ICTC)* (pp. 28-33). IEEE 2021, May.

[14] J. Lv, X. Wang, M. Huang, J. Shi, K. Li and J. Li, "RISC: ICN routing mechanism incorporating SDN and community division," *Computer Networks,* vol. 123, pp. 88-103, 2017.

[15] LANE, A. et al., 2019, "MiniNDN," MiniNDN Developers. [Online]. Available: https://github.com/named-data/mini-ndn.

[16] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang. "ndnSIM 2: An updatedNDNsimulator for NS-3." *Technical Report NDN-0028, Revision 2. NDN*.2016.

[17] A. Alameen, S. Rubab, B. Dhupia and M. Kolhar, "Security in OpenFlow Enabled Cloud Environment," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 7, 2017.

[18] N. Z. Abidin, A. R. Ismail and N. A. Emran, "Performance Analysis of Machine Learning Algorithms for Missing Value Imputation," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 6, 2018.

[19] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, Reinforcement learning, fast and slow. *Trends in cognitive sciences*, vol. 23, no. 5, pp. 408-422, 2019.