future internet

MDPI

Review

Software-Defined Named Data Networking in Literature: A Review

Albatool Alhawas * and Abdelfettah Belghith

Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia; abelghith@ksu.edu.sa
* Correspondence: aalhawas2@ksu.edu.sa

Abstract: This paper presents an in-depth review of software-defined named data networking (SD-NDN), a transformative approach in network architectures poised to deliver substantial benefits. By addressing the limitations inherent in traditional host-centric network architectures, SD-NDN offers improvements in network performance, scalability, and efficiency. The paper commences with an overview of named data networking (NDN) and software-defined networking (SDN), the two fundamental building blocks of SD-NDN. It then explores the specifics of integrating NDN with SDN, illustrating examples of various SD-NDN models. These models are designed to leverage SDN for NDN routing, caching, and forwarding. The paper concludes by proposing potential strategies for further integration of SDN and NDN and some open research questions. These proposed strategies aim to stimulate further exploration and innovation in the field of SD-NDN.

Keywords: SDN; NDN; IOT; caching; forwarding; routing; network architecture

check for updates

Citation: Alhawas, A.; Belghith, A. Software-Defined Named Data Networking in Literature: A Review. *Future Internet* **2024**, *16*, 258. https://doi.org/10.3390/fi16080258

Academic Editors: Izzat Alsmadi and Gianluigi Ferrari

Received: 13 June 2024
Revised: 9 July 2024
Accepted: 22 July 2024
Published: 23 July 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of things (IoT), intelligent transport systems, and wireless sensor networks, which focus on connecting various devices with limited resources, including mobile devices and sensors, to the Internet, are computer networking fields that are rapidly expanding. According to IoT Analytics, a market research firm, global IoT connections grew by 18% in 2022 and are projected to increase by another 16% by 2025 [1].

In these networks, prioritizing data retrieval is crucial, often outweighing the importance of identifying the specific nodes transmitting the data. Therefore, the emphasis is on the integrity and content of information rather than its origin [2]. However, the current host-centric network architecture, which relies on physical locations and IP addresses, presents challenges, especially for mobile and resource-constrained devices. This architecture impedes network mobility and strains resources by requiring data retrieval through potentially distant IP addresses, even when nearby nodes possess the required data.

An effective solution to this issue lies in adopting an information-centric networking (ICN) architecture, such as named data networking (NDN). ICN shifts the focus from data location to content, facilitating data exchange based on content names instead of locations. This approach prioritizes securing data content over communication channels, thereby enhancing network performance, scalability, security, and efficiency [3]. For instance, secure data content can be cached and accessed from proximate nodes rather than being retrieved from a distant producer, thereby improving efficiency.

Despite its numerous advantages, NDN encounters various design challenges. Specifically, NDN routers handle both data forwarding and routing, so they are burdened with tasks, such as managing the named data link state routing (NLSR) protocol, routing information base, content store (CS), forwarding information base (FIB), and pending interest table (PIT) [4]. To mitigate these challenges, software-defined networking (SDN) provides a solution by separating the control and management planes from the data planes. This approach enables SDN to augment NDN by enforcing global network policies, managing

and distributing routing and forwarding tables, handling data naming and matching, and managing in-network caching [5]. Leveraging SDN to relieve NDN routers of routing responsibilities, thus allowing them to focus solely on data forwarding, can significantly enhance their performance. In addition, transferring the routing task to a centralized control unit can provide a comprehensive network perspective, facilitating the development of more efficient in-network caching strategies for NDN. Furthermore, this global network perspective can bolster support for consumer and producer mobility within the NDN framework.

While merging NDN with SDN offers numerous benefits, compatibility issues between existing SDN protocols/controllers and NDN persist [5]. This necessitates either modifying current SDN protocols/controllers or developing new ones suitable for NDN while preserving the key features of both architectures. This paper surveys the various software-defined named data network models in the literature, compares them, and then provides a list of open research questions and potential strategies for SD-NDN.

The paper is structured as follows. First, we provide an overview of NDN and SDN. Following this foundational understanding, we proceed with a comprehensive survey of various implementations that leverage the advantages of both NDN and SDN. Subsequent to this survey, we discuss the surveyed implementations and provide insights into open challenges, potential strategies, and open research questions. Finally, we conclude by summarizing our findings and presenting our conclusions.

2. Related Work

Several SD-NDN models have been examined in previous works [6–8]. However, these surveys, while comprehensive, do not mainly focus on leveraging SDN to enhance NDN functionalities, which is the primary focus of our paper. For example, in [6], the authors primarily investigate IoT networks and how SD-ICN can enhance IoT services. In [7], the authors explore the implementation of ICN over SDN to enhance SDN functionality and vice versa. In this paper, however, our focus is mainly on models that are used to enhance NDN forwarding, routing, and caching. Table 1 presents a comprehensive comparison between our survey and other related surveys in the field. The table is organized into several columns: Reference, Year, SDN, NDN, Examples of SD-NDN, Comparison between SD-NDN, and Notes. These columns facilitate a detailed comparison across multiple dimensions which are the year of the survey, the inclusion of SDN background, the inclusion of NDN background, the provision of specific examples of SD-NDN implementations, and a comparative analysis of various SD-NDN implementations.

Table 1. Related work.

Ref.	Year	SDN	NDN	Examples of SD-NDN	Compare between the SD-NDN Examples	Notes
[8]	2016	Yes	Yes	Yes	No	Explores methods of implementing NDN over SDN.
[9]	2018	Yes	No	No	No	Compares between different SDN controllers.
[10]	2019	No	Yes	No	No	Surveys different name lookup schemes in NDN.
[7]	2022	Yes	Yes	Relevant to IOT services	Yes	Focuses on leveraging SD-NDN to improve IOT services.
[11]	2022	No	Yes	No	No	Explore different NDN forwarding strategy implementations.
[12]	2022	No	Yes	No	No	Surveys different NDN naming schemes.
[6]	2023	Yes	Yes	Includes examples of NDN over SDN	Yes	Explores the implementation of NDN over SDN and vice versa.
This paper	2024	Yes	Yes	Yes	Yes	Focuses on leveraging SD-NDN to improve NDN routing, forwarding, and caching.

3. Overview and Background

In this section, we provide an overview of two key networking paradigms: NDN and SDN. NDN represents a data-centric approach to networking, prioritizing content retrieval over traditional host-centric communication. We begin by examining the architecture and fundamental principles of NDN. Following this, we explore SDN, a paradigm that separates the control and data planes in network infrastructure, offering programmability and flexibility, and present an overview of SDN architecture and its components.

3.1. Named Data Networking

NDN employs an hourglass architecture similar to the TCP/IP protocol stack, as shown in Figure 1. From the figure, one can see that unlike the traditional Internet hourglass model, the NDN architecture introduces a paradigm shift by replacing IP packets in the network layer with content chunks, effectively eliminating the need for a separate transport layer. This transformation is further enhanced by the addition of a security layer on top of the content layer, ensuring data integrity and authenticity. Additionally, a strategy layer is introduced at the bottom of the content layer, which optimizes data retrieval based on network conditions, thus making NDN a more efficient and secure data retrieval system.

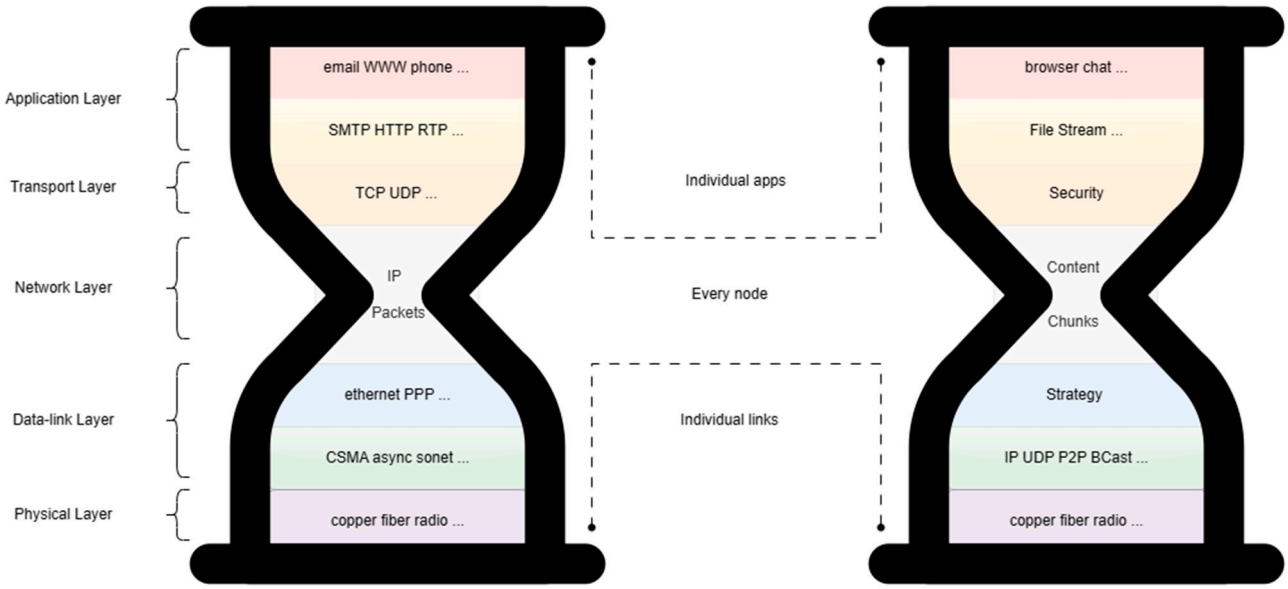

Figure 1. Hourglass-shaped NDN architecture (**right**) and hourglass-shaped Internet (**left**).

By maintaining this hourglass structure, NDN acknowledges the successful design of the Internet, which has facilitated significant growth by encouraging innovation across various layers [13]. However, NDN introduces a new shift by redefining the thin waist to prioritize the retrieval of data (contents) over data locations (IP addresses). This shift reflects a profound understanding of the strengths and limitations of the current Internet architecture. NDN fundamentally changes the semantics of network services by moving from delivering packets to specific destination addresses to retrieving data identified by names.

Furthermore, NDN addresses security concerns by incorporating basic security measures at the thin waist, ensuring data integrity through the signing of all named data by the data producers. By retrieving data based on names rather than IP addresses, NDN avoids issues related to security and enables seamless mobility, as data names remain unchanged, unlike IP addresses, which must change when users move to a different location. This approach is in line with modern application development trends, where applications prioritize the information they need over specific locations, enhancing the adaptability and functionality of NDN [14].

In NDN, several key terms are used to distinctly identify its various components [4].

- Packet: A packet refers to a named data object (NDO) that exceeds the network's maximum transmission unit (MTU) in size. When an NDO is broken down at the data link layer, these smaller pieces are known as frames. NDN typically processes two kinds of packets: interest and data (Figure 2).
    - o An interest packet signifies a request for specific data. When a consumer requires certain information, it dispatches an interest packet into the network to find the desired content.
    - o In contrast, a data packet carries the actual content being transmitted. It answers the requests initiated by interest packets, delivering the requested data to consumers.
- Consumer: A consumer is an entity capable of generating names and requesting data through interest packets.
- Producer: A producer is an entity that produces data packets to meet consumer requests.
- NDN node: An NDN node can function as a consumer, producer, or forwarder of cached content.
- Forwarding plane: The forwarding plane of NDN is composed of three data structures: CS, PIT, and FIB. The forwarding plane utilizes these structures to forward data using specific forwarding strategies as shown in Figure 3.
- NDN routers: NDN routers are devices that perform fundamental routing and forwarding operations. They contain CS, PIT, and FIB. They can cache content in their CS to fulfill future requests for the same content. Additionally, they forward data according to their forwarding tables. However, they are not to be mistaken for content providers.
- Routing plane: The routing plane refers to a collection of NDN protocols used to share information about the accessibility of a namespace. Whereas the routing plane calculates and decides globally on routes' availability, the forwarding plane makes per-node decisions about the preference and use of these routes based on their performance and status. In other words, the routing plane finds and calculates the costs of routes toward the requested content and provides them to the forwarding plane.
- Data/content names: Data/content names are unique, human-readable strings usually generated by applications and act as identifiers for NDOs.

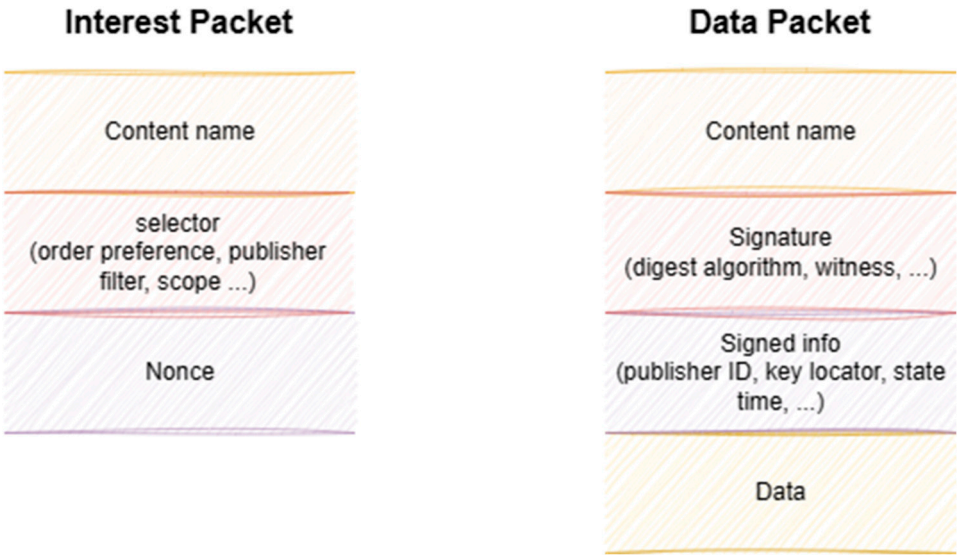

Figure 2. Packets in NDN architecture.

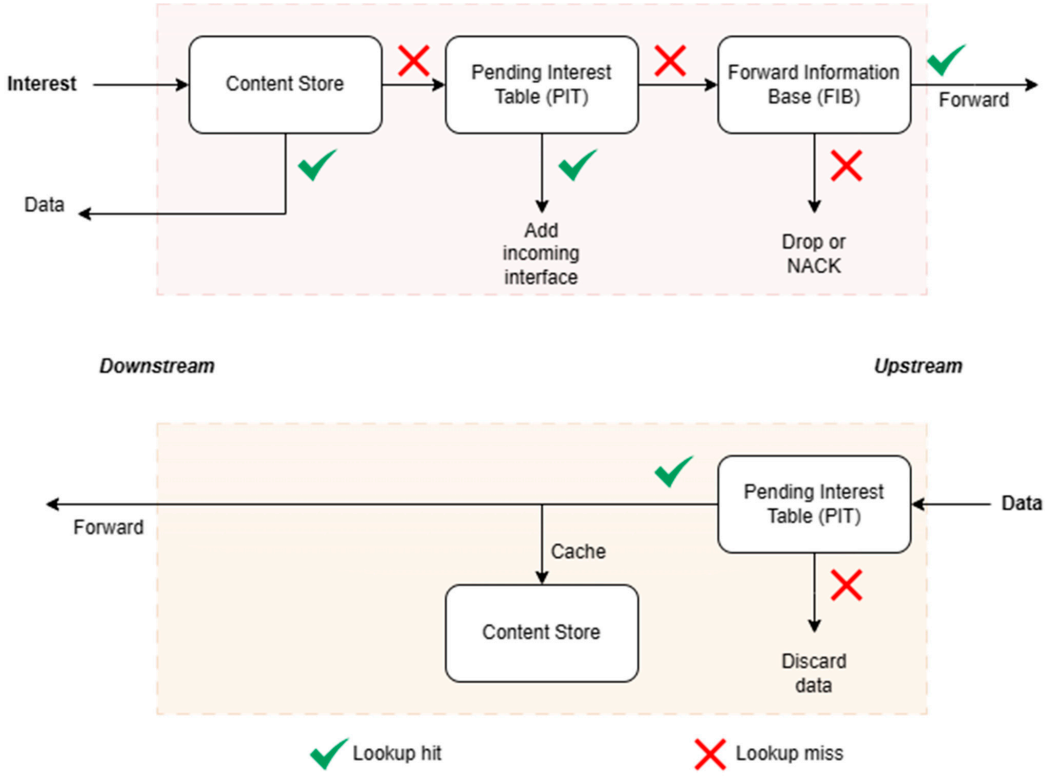

Figure 3. NDN forwarding process.

The following sections outline the main aspects of NDN naming, routing, NLSR, and forwarding.

3.1.1. NDN Naming

The NDN naming system reflects the structure of URLs, employing a hierarchical organization in which components are separated by a forward slash (/). For instance, a document produced by KSU might be named /KSU/doc/report.pdf. This hierarchical format provides a way for applications to indicate relationships between different data elements. For example, within this structure, chapter 3 of version 1 of the document could be identified as /KSU/doc/report.pdf/1/3 [14]. More about NDN naming can be found in [10,12].

3.1.2. NDN Routing

The routing process in NDN comprises three primary stages: name resolution, discovery, and delivery. Name resolution converts a name data object into its corresponding locator, while the discovery stage routes requests to the relevant data based on their respective names. Subsequently, the delivery phase ensures the transmission of the data chunk to the requesting user. Currently, NDN employs NLSR [15] for routing, which utilizes a hierarchical naming scheme for routers and link state advertisements (LSAs). Under this scheme, each router is named based on its corresponding network, such as <network>/<site>/<router>. The protocol generates multiple routes to content names, which are stored in the RIB before being used to update the FIB, considering route cost factors [4]. More about NDN routing can be found in [16–19].

3.1.3. NLSR

NLSR [15] propagates two types of LSAs: adjacency LSA and prefix LSA. The adjacency LSA advertises active links between an NDN router and its neighbors, including the router's name, active links, and link costs. It is advertised at the router startup and when there is a status change in a router's links.

The prefix LSA advertises a name prefix registered with a router, containing a validity flag and the name prefix. Each router may announce multiple prefix LSAs if it has multiple name prefixes. When a name prefix is deregistered, the NLSR updates the corresponding prefix LSA by setting the validity flag to 0 and disseminating the new LSA to other router nodes.

Obsolete LSAs are removed from the Link State Database (LSDB) when their lifetime expires. Therefore, if a router crashes, its neighbors will update the status of their LSAs, and traffic will not be directed over those broken links [16].

The NLSR ensures that all routers have a consistent view of the network topology and name prefix reachability through an LSDB synchronization process. This process begins with the computation of a hash tree over all data. The root hash is then exchanged between neighboring routers to detect inconsistencies.

If inconsistencies are found, the routers engage in the process of exchanging hash values of data on the next tree level. This continues until they identify the specific data causing the discrepancies. Once identified, the routers exchange the data to reach consistency.

The LSDB synchronization process is summarized in the following nine steps, which are illustrated in Figure 4 [16]:

1. Routers periodically send out a root advise interest that contains the hash value of its LSDB to their neighbors to check if their databases match. The neighboring routers reply only when they detect any differences.
2. When Router A's NLSR creates a new LSA, it writes the LSA in the CCNx repository (REPO), which changes Router A's LSDB hash value.
3. Router A then detects the difference from Router B's LSDB hash value and replays the root advise interest sent in Step 1.
4. The routers continue exchanging hash values at the next tree level until they find the exact data causing the mismatch. After that, Router B sends a content interest packet to retrieve the data.
5. Router A then sends the requested data (LSA) to Router B.
6. Once Router B's REPO receives the data (LSA), it sends a notification to its local NLSR with the data name.
7. Following that, Router B's NLSR requests the LSA from the REPO.
8. Subsequently, Router B's REPO replies with the LSA.
9. Lastly, Router B's NLSR updates its LSDB with the new LSA.

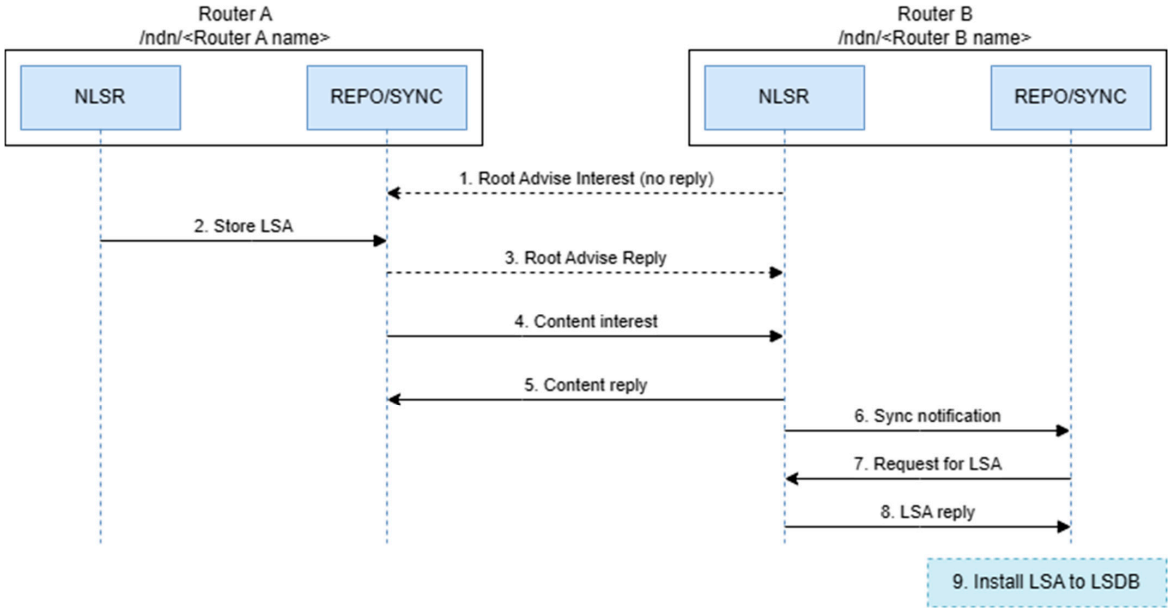

Figure 4. LSDB synchronization process.

3.1.4. NDN Forwarding

NDN routers depend on three essential data structures: a CS for caching data packets, a PIT for managing forwarded but unsatisfied interest packets, and an FIB for associating content name prefixes with forwarding interfaces. The forwarding process within NDN is shown in Figure 3. Upon receiving an interest packet, the router initially checks its CS. If the requested data packet is cached, the router promptly responds with the corresponding data packet via the same interface. Conversely, if the data are not found in the CS, the router searches the PIT. If the interest packet is already registered in the PIT, the router appends the new interface to the PIT and discards the packet. If not, the router consults the FIB to determine the appropriate interface for forwarding the packet. In the absence of a match in the FIB, the packet is discarded.

Alternatively, upon receiving a data packet, the router verifies whether the data are registered in the PIT. If not, the packet is discarded. Conversely, if the data are registered in the PIT, they are cached and forwarded to the appropriate interface based on PIT entries. The PIT plays a crucial role in maintaining a loop-free network in NDN, and interest packets are removed from it once matching data are received or a timeout occurs [20].

Based on the aforementioned forwarding process, functions of the forwarding plane in NDN can be categorized into four key areas [4]:

1. Name lookup: Matching content names with data structure entries. Generally, lookup in the FIB is performed by Longest Prefix Match (LPM), whereas in the PIT or CS, lookup is performed by searching with one-to-one correspondence. More about the NDN name lookup can be found in [10,12].
2. Forwarding strategies: Select the appropriate interface to forward interest packets. An example of such a strategy is the NDN basic forwarding strategy, which forwards the interest packet to all matching context prefix interfaces (except the interest incoming interface). Different forwarding strategies can be found in [21–25].
3. Cache placement: Determining where to place cached content. The default NDN cache placement strategy is Leave Copy Everywhere (LCE), which may lead to content redundancy, but there are many other strategies in the literature that can be implemented, such as [26–29].
4. Cache replacement: Employing replacement policies for cached items. Classical proposed replacement policies include Least Recently Used (LRU) and Least Frequently Used (LFU); however, there are many others, such as [30–33].

In summary, NDN is an innovative approach to networking that focuses on content names rather than IP addresses. It operates by broadcasting interest packets to request content, which are routed through the network until they find an NDN node that holds the desired data. This node then sends back a data packet along the reverse path to the consumer. The shift from IP addresses to content names enhances mobility support within the network, and the in-network caching of frequently requested content reduces latency and improves the overall network experience.

3.2. Software-Defined Networking

SDN is an emerging technology that offers numerous benefits to traditional networking architectures. By dividing the network into the three distinct layers shown in Figure 5—the application layer, the control layer, and the data layer—SDN revolutionizes network management and control.

In this innovative approach, intelligence is abstracted from individual network devices within the data layer and centralized within an SDN controller situated in the control layer. This centralized controller maintains a comprehensive view of the network topology and link status, facilitating dynamic routing and management decisions. Such centralized control mitigates the intermixed management, control, and data planes inherent in traditional networking architectures, thereby addressing serious security issues and promoting network scalability. Furthermore, it enables seamless protocol changes within one plane without disrupting others, making the network more adaptable to evolving requirements.

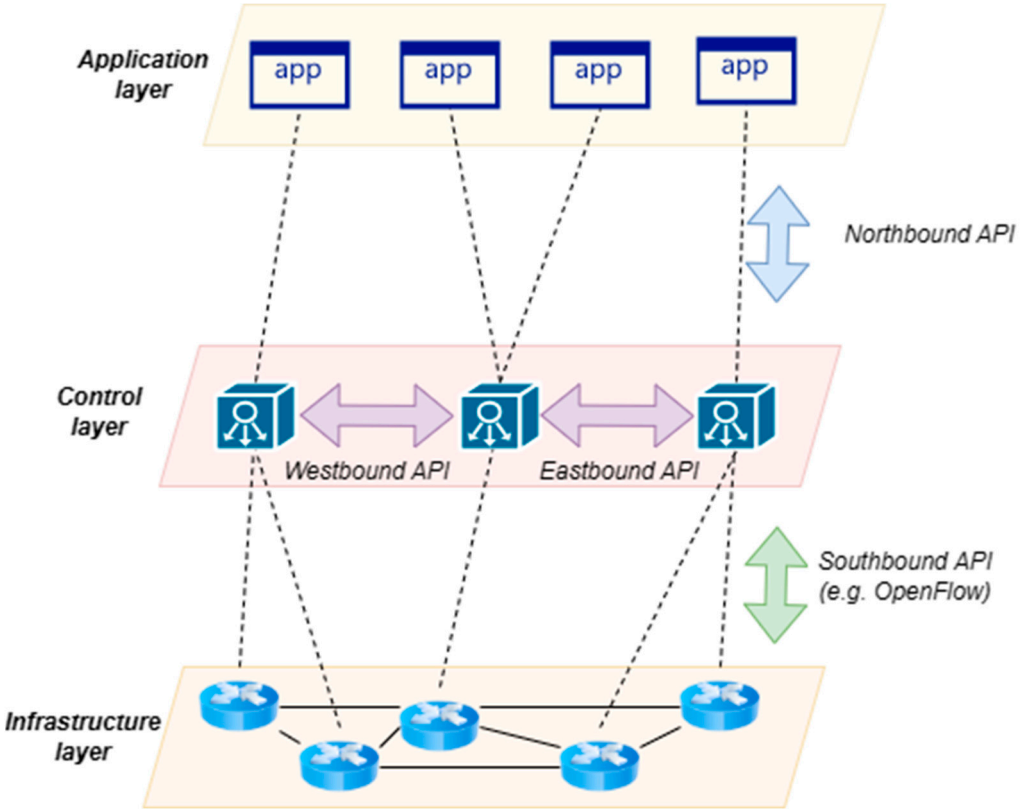

Figure 5. SDN structure.

The SDN controller communicates with switches in the data layer via a southbound API, with OpenFlow (OF) [34] being a prominent example of a southbound API. Through this interface, the controller pushes routing and management commands to the switches, allowing for efficient network configuration and optimization.

Additionally, the application layer in SDN encompasses a variety of applications that interface with the SDN controller through a northbound API. These applications enable network administrators to control and manage the SDN controller, providing flexibility and customization options. Among the tools used in this context is Frenetic [35], which is not an application itself but a programming language used to build SDN applications.

Moreover, in large-scale network environments, a west/eastbound API may be employed to connect multiple SDN controllers in the control plane. This facilitates coordination and cooperation among controllers, ensuring efficient network-wide management and control [36].

Furthermore, within the context of SDN, the OF protocol serves as a popular southbound API. It facilitates communication between the control layer and the data layer by establishing a secure channel between the OF switches and the SDN controller, as displayed in Figure 6. An OF switch supports three primary functions: a secure channel, flow table operations, and adherence to the OF protocol [37].

The operation of an OF switch's flow table involves multiple tables that are processed sequentially in a pipeline manner. When a packet arrives at an OF switch, it undergoes a matching process within this pipeline, as shown in Figure 7. Each entry in the flow table consists of three main components: the header field, action (also referred to as instructions), and statistics. Packet matching occurs based on the header fields, and the packets are processed according to the action specified in the flow entries. Additionally, the statistics component provides valuable information about the network status, encompassing aspects such as priority, counters, timeouts, cookies, and other relevant fields.

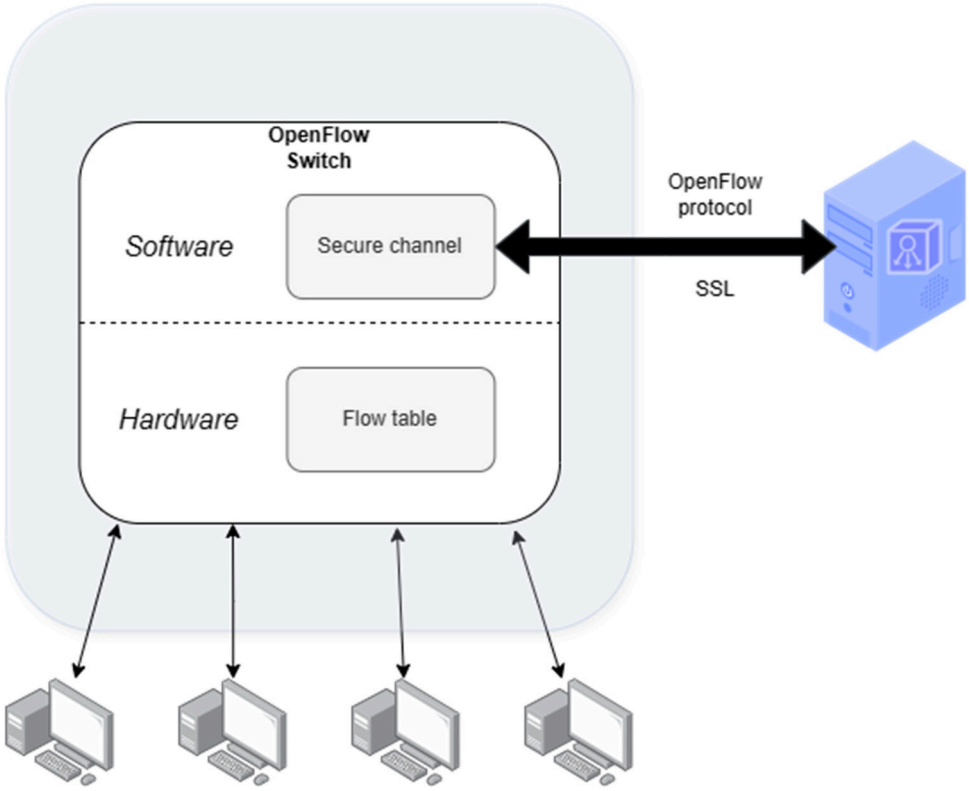

Figure 6. OF switch design.

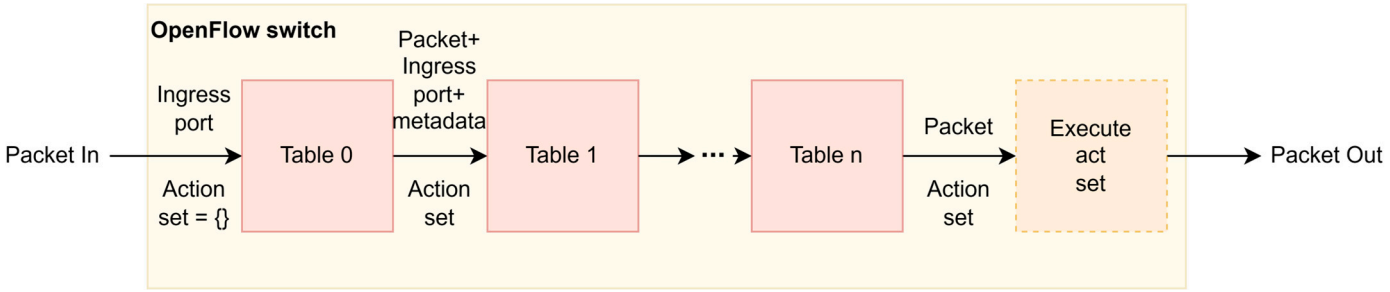

Figure 7. Pipeline matching process in OF switch.

The OF protocol consists of three primary categories of messages—controller-to-switch messages, asynchronous messages, and symmetric messages—each comprising various subtypes. Among the commonly exchanged messages is the packet-in message, which is sent when the switch fails to locate a match for a packet within its flow tables. Similarly, the flow-mod message is dispatched by the controller to modify flow table entries within an OF switch, while the packet-out message is sent by the controller to direct an OF switch to forward a specific packet. These examples illustrate the typical messages exchanged between OF switches and controllers [38].

SDN encompasses various types of controllers, including NOX, POX, Floodlight, and Ryu [39,40]. These controllers can update switch table entries in two modes: proactive and reactive. In proactive mode, the controller updates the entries autonomously. Conversely, in reactive mode, the switch requests the controller for updates. In summary, SDN divides the network into distinct layers, coupled with centralized control and flexible APIs, offering enhanced programmability, scalability, and adaptability compared to traditional networking paradigms.

4. Literature Review

In the rapidly evolving field of network architecture, the integration of SDN and NDN has emerged as a promising approach to enhancing various NDN mechanisms. This literature review aims to provide a survey of multiple implementations that have successfully combined SDN and NDN to improve key NDN mechanisms, such as routing, caching, and forwarding. By exploring these implementations, we hope to shed light on the potential of this integrated approach and its research gaps and open questions.

Adnan et al. [41] conducted a study on the integration of SDN with NDN to enhance QoS-aware mobility architecture. The authors proposed a model, software-defined proactive caching architecture for consumer mobility, which proactively dispatches data packets to new routers during handovers to manage consumer mobility in NDN. This dispatching is controlled by an SDN controller. While the simulation results showed improvements in network performance metrics, such as CPU usage, delay time, jitter, throughput, and packet loss ratio, it is important to note that this design may not be suitable for highly mobile networks due to the centralized control of data packet dispatching. This limitation is particularly relevant in scenarios where frequent handovers occur, potentially overloading the central controller and affecting network performance.

The authors of [42] proposed a novel approach to cache content placement in software-defined ICN–IOT networks. They introduced an adaptive cache placement framework, which leverages a sliding window–based mechanism to optimize cache placement in accordance with energy constraints and data attributes, such as freshness and popularity. The framework is managed by a software-defined cache controller (SDCC) that operates in both centralized and decentralized modes. While the simulation results demonstrated improvements in energy efficiency, it is important to note that the centralized model may face challenges due to its single point of failure and scalability issues. These limitations are particularly relevant in scenarios where a network grows, as all data statics must go to a single controller, potentially overloading the SDCC and affecting network performance. The decentralized model was introduced by the authors to address these limitations by distributing control across multiple controllers.

Guesmi et al. [43] proposed an SDN–NDN model that repositions the FIB from NDN routers to a controller that also manages a management information base (MIB) and a data information base (DIB). The MIB contains the neighbor lists of all nodes linked to the controller, while the DIB stores all prefixes found in the CS of all nodes connected to the controller. In this model, when an NDN router receives an interest packet that needs forwarding, it consults the controller to determine the forwarding interface. Although the simulations indicated positive network performance metrics, such as bandwidth, delay, cache hits, and memory overhead, the model has scalability issues. It is challenging to store and look up the data prefixes of all nodes in one controller. Furthermore, consulting the controller for every interest packet that needs forwarding is not practical.

Tariq et al. [44] developed a model in which a controller manages a routing information table (RIT), a global information table (GIT), and a flow forwarding information base (FlowFIB). The RIT stores routing information; the GIT contains information about neighboring nodes, their contents, and the energies of all nodes connected to the controller; and the FlowFIB contains the FIB of all nodes connected to the controller. Meanwhile, an NDN router manages a CS, a PIT, and a local information table, and when it receives an interest packet that is not in its CS or PIT, it forwards the interest to the controller based on a predefined priority window. The controller then computes the optimal route based on energy and priority and sends this information back to the router. While the simulations showed positive results in terms of energy consumption and content retrieval, the model has scalability issues, as it is challenging to forward all interest packets to the controller. Moreover, storing and looking up the data prefixes of all nodes in one controller can be resource-intensive and time-consuming.

Alhowaidi et al. [45] proposed a multipath forwarding strategy within an SDN framework (S-MP). In this approach, upon receiving an interest without a match in an NDN router CS, PIT, and FIB, the node forwards it to the SDN controller. The controller then

responds with multiple potential paths to the content holders. Next, the NDN router forwards the interest packet along all of these paths. To maintain a global view of the CSs, controllers must receive updates from NDN routers about their CSs. Testing on a wide-area network demonstrated the routing scheme's effectiveness, although scalability remains a concern because it is not practical to frequently consult the controller and make the controller keep track of all CSs.

In the ICNoSDN model [46], a consumer sends out an interest packet received by an OF switch. This switch then contacts the SDN controller to obtain the appropriate forwarding rules, which are subsequently distributed by the controller to the relevant switches in the path to the producer. After this, the interest packet navigates through the network until it reaches the producer. Once the producer receives the interest packet, it replies to the consumer using the reverse path forwarding (RPF) technique. Notably, the initial data packet is relayed to the SDN controller by an OF switch, which prompts the controller to establish and disseminate forwarding rules that direct the data packets to a cache server for storage. Although this model design was successful, it is not practical to burden the SDN controller with excessive caching and forwarding inquiries, potentially hindering the network's performance and scalability.

The CRoS–NDN model [47] functions similarly to the preceding models. When a router encounters an interest packet that lacks a corresponding entry in its FIB, it relays this interest to the controller. The controller then updates the FIB of all routers along the route with new entries. Once a content holder receives the interest, it dispatches the requested content back to the requester using an RPF algorithm. However, this model can potentially overwhelm the controller with the volume of interest and FIB updates, leading to scalability issues.

The NDNS [48] model operates by ensuring that unmatched interest packets are not forwarded to the controller, thus preventing it from becoming overwhelmed. Each NDN node maintains an MIB that records the details of neighboring nodes and a DIB that lists all prefix names associated with these neighbors. The controller possesses a global MIB (GMIB), a global DIB (GDIB), an RIB, and an FIB for all its connected nodes. It updates the nodes' FIBs based on changes in the GMIB and the GDIB. While this approach successfully reduces the number of interest queries directed at the controller, it still requires the controller to manage a large GDIB lookup table, which is impractical for scalability and efficiency.

In the SDICN [49] model, consumers initiate communication by generating interest packets. These packets are sent to OF switches, which then forward them to the SDN controller. The controller is responsible for locating the content, establishing an optimal placement policy, and determining the optimal path for packet delivery. Similar to the aforementioned models, this one is also not scalable.

Jeeva [50] introduced a model known as OF–ICN that combines SDN with NDN and significantly alters the routing process in NDN. In this model, when a consumer sends an interest to the OF switch, the switch initiates a search in its CS, PIT, and FIB for a corresponding match. If the search yields no results, the OF switch consults the OF–ICN controller. The controller, in turn, provides the necessary routing information. Armed with this information, the OF switch adds interest to the PIT and forwards it to the appropriate interface. Eventually, the data reach either the producer or a node that holds the data and are then sent back to the consumer. While the simulation results showed improvement in the round-trip time, it is important to consider the model's potential limitations. The design may not be optimal for large-scale networks due to the frequency of communication with the central controller, which could potentially overload the controller and impact network performance.

In the SRSC [51] model, a router forwards consumer-generated interest packets to the controller if it cannot find a match in its FIB. The controller then guides the interest through the network, either to a local content holder or toward a border node for external content. This process repeats across routers until the content is found and returned via the RPF strategy. Although effective, this model risks overwhelming the controller with frequent queries, potentially affecting network performance and scalability. Table 2 succinctly showcases a comparative analysis of the various models discussed throughout this section.

Table 2. Literature review.

Model Ref.	Date	SDN	Enhanced Mechanism	Simulation Tool	Evaluation Matrices	Model Limitations
SDPCACM [41]	2023	ODL controller	Consumer mobility	Mininet	CPU usage, Jitter, Throughput, and Packet loss ratio	Not suitable for networks with high mobility
[42]	2022	SDN-like (SDCC)	Caching	MATLAB	Energy consumption, and Cache hit ratio	Centralized model = single point of failure, and Not scalable
[43]	2021	SDN-like	Routing and forwarding	NDNSim	Bandwidth, Delay, Cache hit ratio, and Memory overhead	Not scalable, Memory overhead, and Efficacy issues due to large prefix tables lookups
EPF [44]	2020	SDN-like	Forwarding	NDNSim	Number of interests, Content retrieval time, Network lifetime, Number of high-priority interests, Retransmissions, and Average energy consumption	Not scalable, Memory overhead, and Efficacy issues due to large prefix tables lookups
S-MP [45]	2018	-	Routing	GENI testbeds [52]	Delay	Not scalable
ICNoSDN V2 [46]	2018	Floodlight controller	Routing and caching	OFELIA project testbeds [53] and Mininet	Crossing traffic of cache servers and data servers, and Number of cached items	Not scalable
CRoS-NDN [47]	2017	SDN-like	Routing	NDNSim	Content delivery delay, and Signaling efficiency	Not scalable
NDNS [48]	2017	SDN-like	Routing and caching	NDNSim	Bandwidth, Delay, Cache hit ratio, and Bootstrap (setup) time	Memory overhead, and Efficacy issues due to large prefix tables lookups
SDICN [49]	2016	NOX controller	Routing and caching	Testbeds	Storage overhead, Request hit ratio, Content download time, and Control computation time cost	Not scalable
OF-ICN [50]	2016	POX controller	Routing	-	Round trip time	Not scalable
SRSC [51]	2015	SDN-like	Routing	NDNSim	Bootstrap (setup) time, Cache hit ratio, and Number of data and interest messages	Not scalable

5. Discussion

The examination of the literature presented in Table 2 revealed that current SD-NDN models are typically tested within limited network scopes. As a result, scalability becomes a significant concern, primarily because the controller is burdened with an excessive number of queries. Moreover, the controller's role in maintaining data prefixes for the entire network or network zone proves impractical in terms of resource allocation and the efficiency of table lookups. From this, we can identify the following open research questions and challenges that researchers might consider when integrating SDN and NDN:

- Existing SDN controllers and southbound APIs are not compatible with the NDN framework. As a result, most of the models in Table 2 had to have their own SDN-like controller software developed, and these may lack some features that known SDN controllers possess.
- Alternatively, if existing SDN controllers, such as Floodlight, are to be used, one must consider the fundamental differences in the protocol stacks between SDN and NDN. SDN uses the IP stack to carry packets, while NDN eliminates the IP stack and relies solely on content names. Implementing a model that integrates both NDN and SDN requires modifications to either the SDN or NDN protocol stack. Two examples of changes that can merge existing SDN controllers with NDN include the following:
    - o Encapsulating SDN packets in NDN packets or vice versa, and
    - o Adding an additional layer to the SDN or NDN protocol stacks.

  In [46], the authors encapsulated NDN packets in SDN packets by employing OpenFlow switches. These switches acted as a proxy between the SDN controller and NDN nodes, transforming NDN packets into SDN packets that could be processed by an SDN controller.
  In contrast, reference [54] aimed to preserve the characteristics of both SDN and NDN. This was accomplished by adding additional layers to the NDN stack for use by SDN controllers, as illustrated in Figure 8. This allowed SDN controllers and switches to communicate with each other using the IP stack, while NDN nodes communicated using the NDN stack.
- When designing an SD-NDN model, one should consider the memory overhead of storing global data prefix tables in SDN controllers, as this consumes storage and increases table lookup time, leading to increased data retrieval times.
- Similar to the above point, controller scalability should be considered when designing an SD-NDN model, as controllers have limitations in the number of inquiries they can handle. A critical consideration is determining how many requests and routers a single controller can effectively manage. In [21], an approach was taken to prevent overwhelming the controller by allowing routers to drop interest packets not found in their FIB, rather than forwarding every unmatched interest to the controller. Additionally, in [24] and [17], an interest is sent to the controller only once, and the controller responds with the end-to-end forwarding path. This path is then forwarded, along with the interest, to the next hop by the router nodes. This contrasts with [27], where the controller is responsible for sending forwarding updates to all routers in the route of interest.
- A viable solution should be adaptable to dynamic topologies, especially in the context of mobility. In [28], controller discovery involved a node flooding the network to find a controller, accommodating node mobility. Conversely, [24] employed a strategy in which a controller floods the network to announce its presence, allowing nodes to bind with the controller. The former supports node mobility, while the latter supports controller mobility. However, it seems more practical to prioritize support for node mobility over controller mobility, given that controllers are less likely to change their locations.
- One of the factors that affect the performance of SD-NDN models is the request processing time and route discovery algorithms of the SDN controller. Most of the

aforementioned solutions rely on the Dijkstra algorithm. However, it is important to evaluate the processing time and efficiency of different route discovery algorithms and to choose the best one for SD–NDN models. In addition, one may consider using AI algorithms and machine learning to improve performance.

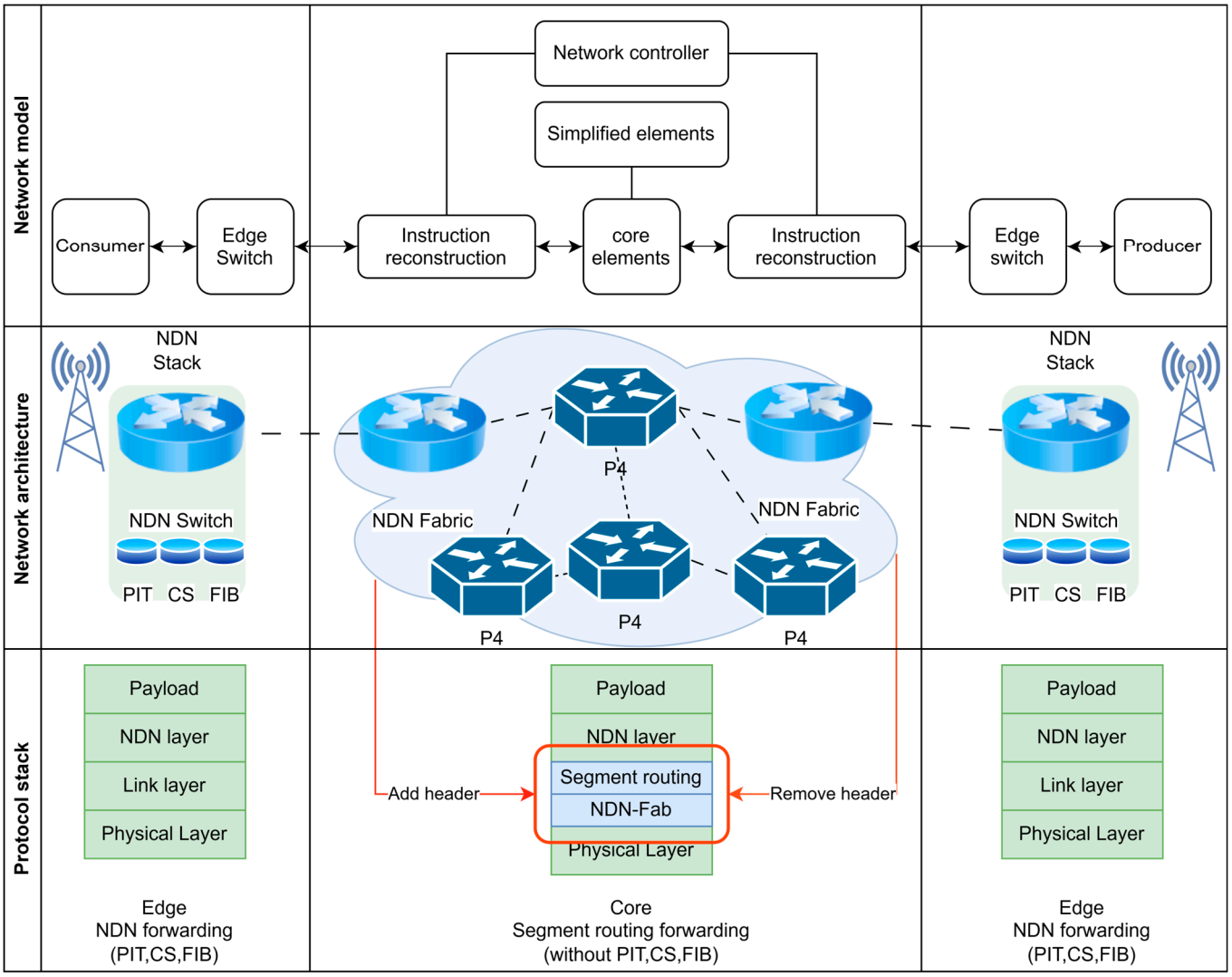

Figure 8. NDN-fabric architecture.

In addition to the points previously discussed, it is important to note that the majority of SD-NDN architectures, as presented in Section 4, could unintentionally compromise the benefits that NDN aims to provide. This could potentially result in the SD-NDN model being less effective than a pure NDN or a host-centric model. Therefore, when integrating SDN into the NDN protocol stack, one should consider avoiding major changes. Significant alterations could negate most of the benefits that one aims to achieve from NDN, potentially rendering the SD-NDN model less effective than an NDN-only model or a host-centric model. Figure 9 serves as an example architecture, demonstrating a possible approach to circumvent this issue. The architecture divides the network into two layers: the control layer and the infrastructure layer. The control layer is composed of a number of controller nodes that manage multiple NDN routers. To prevent controllers from being overwhelmed, the quantity of NDN routers that each controller supervises should be balanced and proportionate to its capacity. The infrastructure layer, on the other hand, is composed of NDN routers, producers, and consumers. Here, unlike regular NDN

networks, the router nodes are responsible only for packet forwarding and switching, and the routing burden is carried to the controllers.

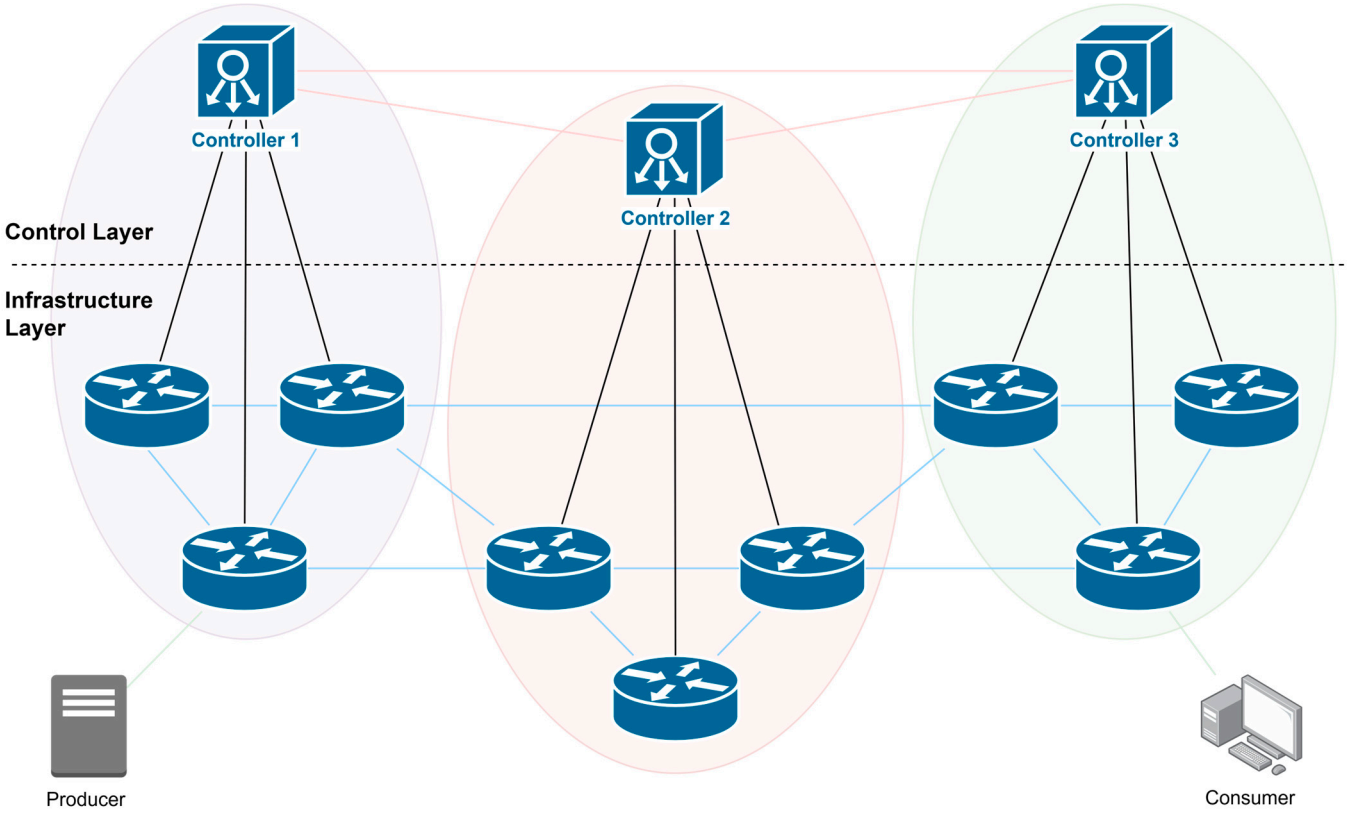

Figure 9. Software-defined NDN architecture.

Shifting the routing task to the controllers enables NDN routers to focus exclusively on forwarding tasks, thereby reducing the burden on NDN router nodes and persevering the benefits of NDN architecture. As explained in Section 3.1.3, in an NDN only network, exchanging LSA between two adjacent routers' NLSRs is done periodically to ensure that routers have a fully updated network view and detect broken links as soon as possible. However, frequent transmission of LSA can overwhelm the network. In addition, updates of the link status take a long time until they propagate through the network [16]. In our example architecture, these two issues might be minimized by adopting the routing process shown in Figure 10.

1. NDN routers periodically send root advise interest that contains the hash tree values of their LSDB to their corresponding controller. The controller will reply only when it detects differences between the hash value it has and the received hash value, indicating that there is an update on the network state.
2. Once a deference between the hash values is detected by controller A, the controller sends a root advise reply that contains a hash value of data at the next tree level to router A. The router and controller then keep exchanging hash values at the next tree level until they find the data that are causing the mismatch.
3. After detecting the data (LSA) that are causing the mismatch, controller A sends an interest packet to request the data (LSA).
4. Router A replies with a data packet containing the LSA.
5. Controllers periodically exchange root advise interest that contains the hash tree values of their LSDB. A reply to this interest will be sent only when controllers detect a mismatch between the hash values, indicating that there has been an update on the network state.

6. Similar to Step 2, once a mismatch is detected between the controllers' hash values, the controllers keep exchanging a root advise reply that contains a hash value of data at the next tree level until they find the data that are causing the problem.
7. After detecting the data (LSA), controller B sends an interest packet requesting the LSA.
8. Controller A replies with a data packet containing the requested LSA.
9. Controller B then sends a synchronization notification to all its connected routers.
10. Router B sends a data interest to request the LSA.
11. Controller B replies with the data.

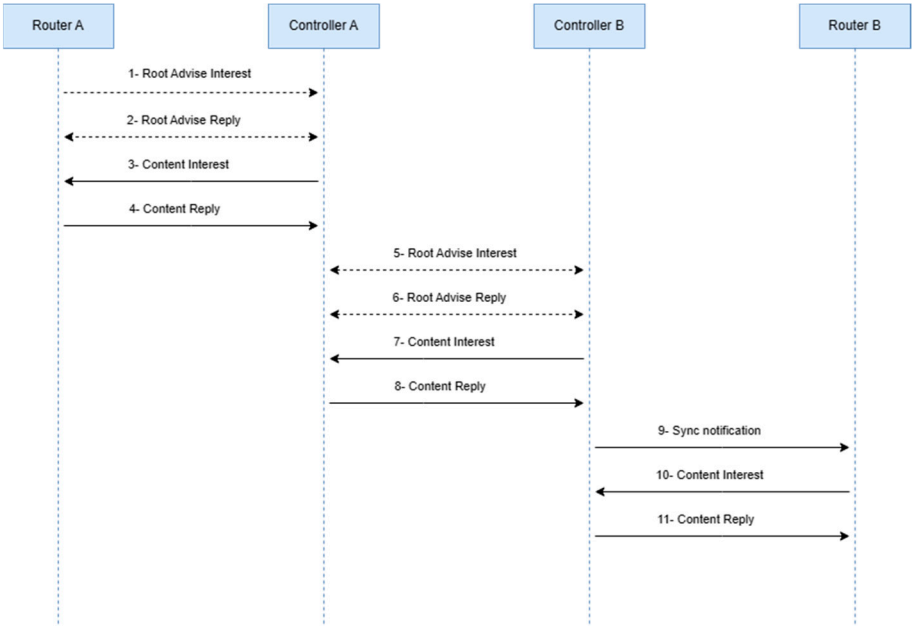

Figure 10. Software-defined NDN routing process.

In this process, once a controller receives a new LSA (Step 4), it will not only update other controllers but will send synchronization notifications to all its other routers, as shown in Step 9. We assume that this process will expedite network state update propagation and remove a huge burden from NDN routers without compromising any of the NDN or SDN benefits, hence making it possible to create an SD-NDN model that preserves the benefit of both NDN and SDN.

6. Conclusions

In this paper, we have explored the integration of SDN and NDN, identifying a range of challenges and posing numerous research questions. A primary concern is the scalability of SD-NDN models, as most existing models have been tested only within limited network scopes. Furthermore, the compatibility between existing SDN controllers and the NDN framework presents a significant issue, necessitating the development of specialized SDN-like controller software tailored for NDN networks.

The architecture we presented in Section 5 serves to illustrate a key point which is that the majority of SD-NDN architectures may inadvertently undermine the benefits that NDN aims to provide, potentially making the SD-NDN model less effective than a pure NDN or a host-centric model. Our intention was to demonstrate a possible approach to circumvent this issue.

In future work, we aim to complete the design of our proposed architecture and conduct comprehensive tests to evaluate its performance. This will provide valuable insights into the practical implementation of SD-NDN and help to fully realize its potential.

Author Contributions: Conceptualization, A.A. and A.B.; writing—original draft preparation, A.A.; writing—review and editing, A.A. and A.B.; visualization, A.A.; supervision, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the Deanship of Scientific Research at King Saud University for funding and supporting this research through the initiative of the DSR Graduate Students Research Support (GSR).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sinha, S. Number of Connected IoT Devices Growing 16% to 16.7 Billion Globally. IoT Analytics. Available online: https://iot-analytics.com/number-connected-iot-devices/ (accessed on 17 May 2024).
2. Al-Karaki, J.N.; Kamal, A.E. Routing techniques in wireless sensor networks: A survey. *IEEE Wirel. Commun.* **2004**, *11*, 6–28. [CrossRef]
3. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36. [CrossRef]
4. Singh, V.P.; Ujjwal, R.L. A walkthrough of name data networking: Architecture, functionalities, operations and open issues. *Sustain. Comput. Inform. Syst.* **2020**, *28*, 100419. [CrossRef]
5. Aboodi, A.; Wan, T.C.; Sodhy, G.C. Survey on the Incorporation of NDN/CCN in IoT. *IEEE Access* **2019**, *7*, 71827–71858. [CrossRef]
6. Rafique, W.; Hafid, A.S.; Cherkaoui, S. Complementing IoT Services Using Software-Defined Information Centric Networks: A Comprehensive Survey. *IEEE Internet Things J.* **2022**, *9*, 23545–23569. [CrossRef]
7. Aldaoud, M.; Al-Abri, D.; Awadalla, M.; Kausar, F. Leveraging ICN and SDN for Future Internet Architecture: A Survey. *Electronics* **2023**, *12*, 1723. [CrossRef]
8. Rowshanrad, S.; Parsaei, M.R.; Keshtgari, M. Implementing NDN using SDN: A review of methods and applications. *IUM Eng. J.* **2016**, *17*, 11–20. [CrossRef]
9. Bannour, F.; Souihi, S.; Mellouk, A. Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 333–354. [CrossRef]
10. Majed, A.; Wang, X.; Yi, B. Name Lookup in Named Data Networking: A Review. *Information* **2019**, *10*, 85. [CrossRef]
11. Tariq, A.; Rehman, R.A.; Kim, B.-S. Forwarding Strategies in NDN-Based Wireless Networks: A Survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 68–95. [CrossRef]
12. Nurhayati, A.; Mayasari, R.; Ahdan, S.; Negara, R.M.; Nurkahfi, G.N.; Syambas, N.R. Naming Scheme on Named Data Networking: A Survey. In Proceedings of the 2022 8th International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, 21–22 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–5. [CrossRef]
13. da Silva, E.T.; Costa, A.L.D.; de Macedo, J.M.H. On the realization of VANET using named data networking: On improvement of VANET using NDN-based routing, caching, and security. *Int. J. Commun. Syst.* **2022**, *35*, e5348. [CrossRef]
14. FIA-NP: Collaborative Research: Named Data Networking Next Phase (NDN-NP). 2014. Available online: https://api.semanticscholar.org/CorpusID:5454097 (accessed on 13 June 2024).
15. Hoque, A.K.M.M.; Amin, S.O.; Alyyan, A.; Zhang, B.; Zhang, L.; Wang, L. NLSR. In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking, Hong Kong, China, 12 August 2013; ACM: New York, NY, USA, 2013; pp. 15–20. [CrossRef]
16. Wang, L.; Lehman, V.; Hoque, A.K.M.M.; Zhang, B.; Yu, Y.; Zhang, L. A Secure Link State Routing Protocol for NDN. *IEEE Access* **2018**, *6*, 10470–10482. [CrossRef]
17. Yi, C.; Abraham, J.; Afanasyev, A.; Wang, L.; Zhang, B.; Zhang, L. On the Role of Routing in Named Data Networking. 2013. [Online]. Available online: http://named-data.net/techreports.html (accessed on 12 June 2024).
18. Zhang, Y.; Xia, Z.; Afanasyev, A.; Zhang, L. A Note on Routing Scalability in Named Data Networking. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]
19. Ariefianto, W.T.; Syambas, N.R. Routing in NDN network: A survey and future perspectives. In Proceedings of the 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), Lombok, Indonesia, 26–27 October 2017; pp. 1–6. [CrossRef]
20. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, k.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
21. Chowdhury, M.; Khan, J.A.; Wang, L. Leveraging Content Connectivity and Location Awareness for Adaptive Forwarding in NDN-based Mobile Ad Hoc Networks. In Proceedings of the 7th ACM Conference on Information-Centric Networking, Virtual, 29 September–1 October 2020; ACM: New York, NY, USA, 2020; pp. 59–69. [CrossRef]

22. Yi, C.; Afanasyev, A.; Wang, L.; Zhang, B.; Zhang, L. Adaptive forwarding in named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2012**, *42*, 62–67. [CrossRef]
23. Alkwai, L.; Belghith, A.; Gazdar, A.; Al-Ahmadi, S. Comparative Analysis of Producer Mobility Management Approaches in Named Data Networking. *Appl. Sci.* **2022**, *12*, 12581. [CrossRef]
24. Alkwai, L.; Belghith, A.; Gazdar, A.; AlAhmadi, S. Awareness of user mobility in Named Data Networking for IoT traffic under the push communication mode. *J. Netw. Comput. Appl.* **2023**, *213*, 103598. [CrossRef]
25. Alkwai, L.; Belghith, A.; Gazdar, A.; AlAhmadi, S. Transparent consumer mobility management in named data networking under the push communication mode. *Comput. Netw.* **2023**, *235*, 109953. [CrossRef]
26. Zhang, Z.; Lung, C.-H.; Lambadaris, I.; St-Hilaire, M. IoT Data Lifetime-Based Cooperative Caching Scheme for ICN-IoT Networks. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–7. [CrossRef]
27. Amadeo, M.; Campolo, C.; Ruggeri, G.; Molinaro, A. Beyond Edge Caching: Freshness and Popularity Aware IoT Data Caching via NDN at Internet-Scale. *IEEE Trans. Green Commun. Netw.* **2022**, *6*, 352–364. [CrossRef]
28. Alahmri, B.; Al-Ahmadi, S.; Belghith, A. Efficient Pooling and Collaborative Cache Management for NDN/IoT Networks. *IEEE Access* **2021**, *9*, 43228–43240. [CrossRef]
29. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K.; AlAhmadi, S. Cache Freshness in Named Data Networking for the Internet of Things. *Comput. J.* **2018**, *61*, 1496–1511. [CrossRef]
30. Putra, M.A.P.; Kim, D.-S.; Lee, J.-M. Adaptive LRFU replacement policy for named data network in industrial IoT. *ICT Express* **2022**, *8*, 258–263. [CrossRef]
31. Wei, Y.; Shi, K.; Li, J. A cache replacement strategy based on content features in named data networking. In Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering, Xiamen, China, 22–24 October 2021; ACM: New York, NY, USA, 2021; pp. 1577–1581. [CrossRef]
32. Kalghoum, A.; Gammar, S.M.; Saidane, L.A. Towards a novel cache replacement strategy for Named Data Networking based on Software Defined Networking. *Comput. Electr. Eng.* **2018**, *66*, 98–113. [CrossRef]
33. Al-Turjman, F.M.; Al-Fagih, A.E.; Hassanein, H.S. A value-based cache replacement approach for Information-Centric Networks. In Proceedings of the 38th Annual IEEE Conference on Local Computer Networks—Workshops, Sydney, NSW, Australia, 21–24 October 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 874–881. [CrossRef]
34. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
35. Foster, N.; Guha, A.; Reitblatt, M.; Story, A.; Freedman, M.J.; Katta, N.P.; Monsanto, C.; Reich, J.; Rexford, J.; Schlesinger, C.; et al. Languages for software-defined networks. *IEEE Commun. Mag.* **2013**, *51*, 128–134. [CrossRef]
36. Rawat, D.B.; Reddy, S.R. Software Defined Networking Architecture, Security and Energy Efficiency: A Survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 325–346. [CrossRef]
37. Gong, Y.; Huang, W.; Wang, W.; Lei, Y. A survey on software defined networking and its applications. *Front. Comput. Sci.* **2015**, *9*, 827–845. [CrossRef]
38. OpenFlow Switch Specification Version 1.5.1 (Protocol Version 0x06) for Information on Specification Licensing through Membership Agreements. 2015. [Online]. Available online: http://www.opennetworking.org (accessed on 12 June 2024).
39. Prabha, C.; Goel, A.; Singh, J. A Survey on SDN Controller Evolution: A Brief Review. In Proceedings of the 7th International Conference on Communication and Electronics Systems, ICCES 2022—Proceedings, Coimbatore, India, 22–24 June 2022; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; pp. 569–575. [CrossRef]
40. Zhu, L.; Karim, M.M.; Sharif, K.; Li, F.; Du, X.; Guizani, M. SDN Controllers: Benchmarking & Performance Evaluation. *arXiv* **2019**, arXiv:1902.04491.
41. Adnan, M.; Ali, J.; Ayadi, M.; Elmannai, H.; Almuqren, L.; Amin, R. Leveraging Software-Defined Networking for a QoS-Aware Mobility Architecture for Named Data Networking. *Electronics* **2023**, *12*, 1914. [CrossRef]
42. Sharif, S.; Moghaddam, M.H.Y.; Seno, S.A.H. Adaptive cache content placement for software-defined Internet of Things. *Future Gener. Comput. Syst.* **2022**, *136*, 34–48. [CrossRef]
43. Guesmi, T.; Kalghoum, A.; Alshammari, B.M.; Alsaif, H.; Alzamil, A. Leveraging Software-Defined Networking Approach for Future Information-Centric Networking Enhancement. *Symmetry* **2021**, *13*, 441. [CrossRef]
44. Tariq, A.; Rehman, R.A.; Kim, B.S. Epf—An efficient forwarding mechanism in sdn controller enabled named data iots. *Appl. Sci.* **2020**, *10*, 7675. [CrossRef]
45. Alhowaidi, M.; Nadig, D.; Ramamurthy, B.; Bockelman, B.; Swanson, D. Multipath Forwarding Strategies and SDN Control for Named Data Networking. In Proceedings of the International Symposium on Advanced Networks and Telecommunication Systems, ANTS, Indore, India, 16–19 December 2018; IEEE Computer Society: Washington, DC, USA, 2018. [CrossRef]
46. Siracusano, G.; Salsano, S.; Ventre, P.L.; Detti, A.; Rashed, O.; Blefari-Melazzi, N. A framework for experimenting ICN over SDN solutions using physical and virtual testbeds. *Comput. Netw.* **2018**, *134*, 245–259. [CrossRef]
47. Torres, J.V.; Alvarenga, I.D.; Boutaba, R.; Duarte, O.C.M.B. An autonomous and efficient controller-based routing scheme for networking Named-Data mobility. *Comput. Commun.* **2017**, *103*, 94–103. [CrossRef]

48. Kalghoum, A.; Gammar, S.M. Towards new Information Centric Networking strategy based on software defined networking. In Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC, San Francisco, CA, USA, 19–22 March 2017; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2017. [CrossRef]
49. Xiulei, W.; Ming, C.; Chao, H.; Xi, W.; Changyou, X. SDICN: A software defined deployable framework of information centric networking. *China Commun.* **2016**, *13*, 53–65. [CrossRef]
50. Jeeva, R. OpenFlow-Based Control Plane for Information-Centric Networking. 2016. Available online: https://publications.scss.tcd.ie/theses/diss/2016/TCD-SCSS-DISSERTATION-2016-058.pdf (accessed on 13 June 2024).
51. Aubry, E.; Silverston, T.; Chrisment, I. SRSC: SDN-based routing scheme for CCN. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), London, UK, 13–17 April 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–5. [CrossRef]
52. Berman, M.; Chase, J.S.; Landweber, L.; Nakao, A.; Ott, M.; Raychaudhuri, D.; Ricci, R.; Seskar, I. GENI: A federated testbed for innovative network experiments. *Comput. Netw.* **2014**, *61*, 5–23. [CrossRef]
53. Salsano, S.; Blefari-Melazzi, N.; Detti, A.; Morabito, G.; Veltri, L. Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed. *Comput. Netw.* **2013**, *57*, 3207–3221. [CrossRef]
54. Madureira, A.L.R.; Araujo, F.R.C.; Araújo, G.B.; Sampaio, L.N. NDN Fabric: Where the Software-Defined Networking Meets the Content-Centric Model. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 374–387. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.