

Article

Leveraging Software-Defined Networking Approach for Future Information-Centric Networking Enhancement

Tawfik Guesmi ¹, Anwar Kalghoum ^{2,*}, Badr M. Alshammari ¹ , Haitham Alsaif ¹ and Ahmed Alzamil ¹

¹ College of Engineering, University of Ha'il, Ha'il 81481, Saudi Arabia; tawfik.guesmi@istmt.rnu.tn (T.G.); bms.alshammari@uoh.edu.sa (B.M.A.); h.alsaif@uoh.edu.sa (H.A.); aa.alzamil@uoh.edu.sa (A.A.)

² CRISTAL LAB, National School of Computer Science, Manouba 2010, Tunisia

* Correspondence: anwar.kalghoum@ensi-uma.tn

Abstract: Information-centric networking (ICN) has been developed as a potential candidate for future networks. In this model, users are provided with content rather than communication channels between the different hosts. The ICN network has several problems such as scalability issues and bandwidth consumption. However, software-defined networking (SDN) has been proposed to improve the networking architectures. The goal of our paper is to propose a new approach to named-data networking (NDN) based on the paradigm of SDN. Our work introduces various research studies carried out in the SDN and ICN contexts. We first present the SDN architecture. Then, we focus on work that combines ICN and SDN architectures. Finally, we show the effects of using the SDN architecture on the named-data network (NDN). Our experimental results show that the use of the SDN architecture has a positive effect on NDN network performance.

Keywords: future Internet; content-centric networking; named-data networking; software-defined networking



Citation: 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. <https://doi.org/10.3390/sym13030441>

Academic Editors: Basil Papadopoulos and Tomohiro Inagaki

Received: 22 January 2021

Accepted: 4 March 2021

Published: 9 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 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

1.1. Research Background and Motivation

The Internet is a global data network used steadily over the past 30 years [1]. The emergence of the first services such as electronic messaging (e-mail) and the Web largely contributed to this network's success in the late 1990s and the increase in its number of users. More recently, new content distribution services have appeared; beyond textual content such as the Web or e-mail, users could, in fact, access multimedia content such as audio or video files, first by downloading them and then via video-streaming systems. If in the 2000s, many users downloaded audio files (in MP3 format), notably via the appearance of P2P platforms such as Napster, and video files via new P2P systems such as eDonkey or BitTorrent, new video-distribution platforms then emerged such as YouTube, Dailymotion and Netflix. These allowed users to watch streaming videos, that is, by watching the video as it was downloaded rather than downloading the entire file first.

Today, video streams represent the bulk of total Internet traffic; OEM Cisco announces that by 2021, 82% of traffic will be dedicated to video [1]. The uses of the Internet today have evolved considerably over the years, while the architecture itself has changed little. The infrastructure has improved with the deployment of optical fibers, which have made it possible to increase speeds, going from a few kb/s to several Gb/s [1], and wireless networks (WiFi) and mobile networks (3G, 4G, 5G) have appeared, providing access to the Internet from any location [2]. However, the Internet architecture has not evolved to meet these new features and is still based on the TCP/IP protocol stack.

Even though improvements have been proposed, such as group communications (multicast), peer-to-peer networks (P2P) [3] and content distribution networks (CDNs) [4], the TCP/IP architecture was not designed to allow the distribution of content on a huge scale and to a huge number of users [5].

As users are mainly interested in content and not in its location, new content-oriented architectures have been proposed for the Internet of the future [5,6]. These architectures, generically called information-centric networks (ICN) [5], focus on the data themselves, not the network hosts; the data are named, but the hosts are not. Therefore, the routing of messages is no longer performed according to the host address, as is the case with the IP architecture, but according to the name of the data. These new architectures are designed to allow the distribution of content on a large scale while adding new enhancements in the network, such as anycast communications, making it possible to direct requests to one or more sources owning the content. They also allow network entities (routers) to store data in their memories to retransmit them during future requests. Thus, the content becomes closer to the users, lightening the load on the network, reducing the delays for the users, and improving their quality of experience.

In conjunction with work on content-oriented networks, software networks or software-based networks (SDNs) have been defined to provide consistency in network management. Software networks make it possible to decouple the data plane (data transmission) from the control plane (routing) of the system through the use of a programmable entity called a controller. The routing decisions are externalized in the controller, thus resulting in a good routing, flow or packet. This flexibility seemed to us to be quite suitable for providing a new routing plan for content-oriented architectures.

1.2. Contributions of the Work

Software-defined networking (SDN) [7–9] is considered important since it helps the continued development of network architectures. This networking model's key features are its isolation of control planes and users and its logically unified control plane [7]. In addition, the devices specify the rules and actions that a possibly-remote controller must take. Software-defined networking's key function is to abstract the network, generating a networking software age marked by a broad capacity to advance and innovate.

The goal of this paper is to propose a new approach to named-data networking (NDN) based on the paradigm of SDN [10,11]. This strategy ensures excellent coordination between the routing plane and the forwarding plane. We plan to use SDN technology to enhance NDN scalability, bandwidth usage, latency and cache hit rates by using cooperation between NDN and SDN. Our solution is offered as a clean-slate approach. This solution is based on two packages (interest and data messages) for exchanging information between the nodes and the controller. It eliminates the need to rely on the TCP/IP infrastructure.

In this paper, we present studies carried out in the SDN and ICN contexts. We also show the effects of using SDN architecture in the named-data network (NDN) [6,12]. The simulations' results demonstrated the positive influence of using SDN on the NDN network. This integration increased the percentage of cache hits and reduced the use of bandwidth. These advantages improved the efficiency of the NDN network.

1.3. Structure of the Paper

The paper's remainder is structured as follows: An outline of the ICN method is provided in Section 2. The SDN paradigm is outlined in Section 3. The research work that incorporates ICN and SDN is detailed in Section 4. In Section 5, we describe our architecture. The simulation tests and outcomes are presented in Section 6. Finally, the paper is finalized in Section 7, presenting potential work.

2. Information-Centric Networking

New architectural criteria, such as support for the delivery of scalable content, mobility, confidentiality, trust, etc., have emerged with the enormous growth of the Internet and the introduction of new technologies to meet emerging needs. The Internet, however, has never been configured to satisfy such demands. A vicious circle of features began to appear, such as mobile IP, to help it grow. Many of these fixes also expanded the architecture of the Internet's complexity. Moreover, the Internet can still not sufficiently solve many existing

and emerging applications in their current form. This raised the question of whether “fix the fixes” would proceed or whether a new architectural solution to the Internet will be required [1]. A research culture has been established in this context. This group addressed the Internet’s core criteria and priorities in the future to come up with new architectures and new paradigms to solve existing issues after recognizing the shortcomings of the Internet in its current state.

Information-centric networking (ICN) [1] has appeared in this sense as a promising contender for the future of Internet architecture. Driven by the growing use of the Internet for the sharing of information rather than for peer contact between end hosts, ICN strives to better represent current and future needs than existing issues. ICN encourages the implementation of caching and multicast in the network by naming information at the network layer, enabling the reliable and fast distribution of information to customers. In the ICN network, however, research efforts are more important than the dissemination of knowledge. Such initiatives use expertise to overcome a range of new restrictions in the existing architecture.

ICN [12] represents a promising proposal for future research studies. This approach influences the interaction models, multiparty communication, and in-network caching. The objective of proposing ICN is to introduce a new network infrastructure service that can satisfy modern needs, especially mobility and content distribution, and that may be adaptable to disruptions and failures. Actually, the ICN has been widely studied by much research work in terms of ICN approaches to addressing several key functionalities such as naming, name resolution, routing, forwarding, security, caching [13] and mobility [12,14]. Various and different propositions are given to deal with these functionalities (Figure 1). We present the most popular architectures: DONA [14], NDN [15], Netinf [14] and PSIRP [16].

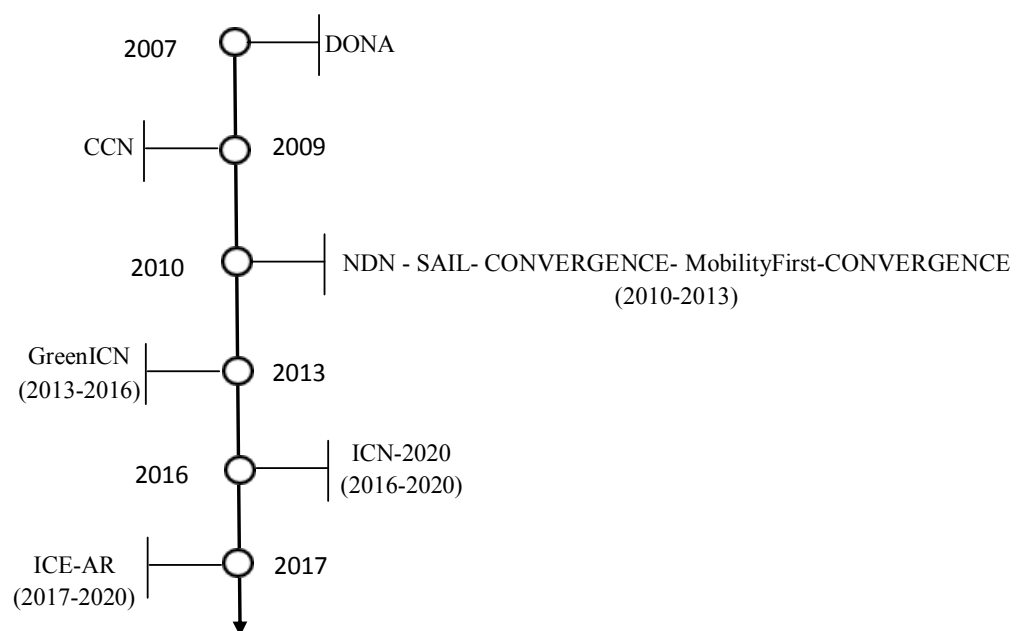


Figure 1. The main information-centric networking (ICN) solutions.

Below, we identify the key features that serve as a basis for presenting and comparing the various ICN initiatives in the rest of the document.

- **Nomimage:** One of the key features of each ICN architecture plan is the structure of the name assigned to the information (or service). Information-relative names are independent of their positions within all ICN architectures. On the other hand, the titles may range from flat to hierarchical, depending on the approach, which may or may not be readable by the recipient.

- Name resolution and data routing: The resolution of the name involves associating an information name with a service or source that can supply it, while data routing involves constructing a route from the provider to the requesting host to transfer information. Whether these two functions are coupled or decoupled is the crucial issue. The request for information is forwarded to the information provider in the coupled method, returning the information to the requesting host after the reverse direction on which the request was returned. The name-resolution feature of the decoupled method does not define or limit the path that the data takes from the supplier to the subscriber.
- Caching: We understand that caching [17] is both on the path and off the path. The network exploits the information stored along the path taken by a name-resolution request in on-path caching, while the network exploits the cached information in off-path caching. This is just hidden outside of this path.

Since the ICN paradigm arose, many research groups have come up with their own ICN solutions. In the next section, we present a few that appear to be relatively advanced and mature.

2.1. ICN Architectures

2.1.1. DONA

The data-oriented (and beyond) network architecture (DONA) [14], the first ICN architecture described in 2007, is shown in Figure 2. The data supplier, known as the Principal, has a public–private key pair in this approach, which is used to generate data names [14]. The content names have the form of P:L, in which L represents the mark given to the content, and P denotes the cryptographic hash of the public key of the publisher. Lookups are carried out by the local RH of a given customer; if the content is found, it is delivered to the requester. When no reference is observed, the query is transmitted to the tree till a source is found. Thus, a source creates an out-of-band delivery (over IP). From the aforementioned ideas, we notice that, in the DONA early-binding approach, the mappings between the identifier and locator are stored by the providers. Obviously, these mappings should be solved before deliveries.

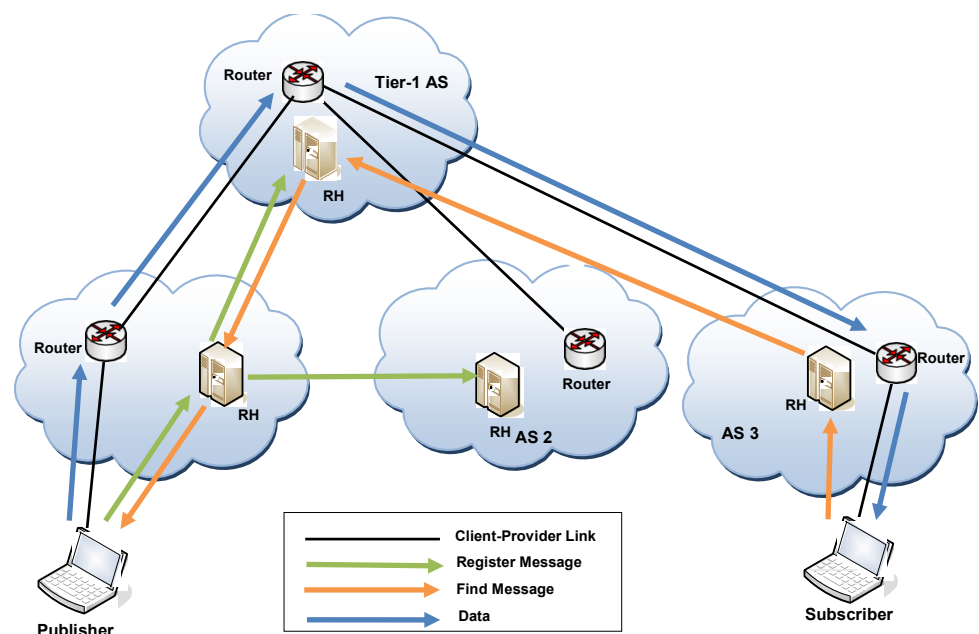


Figure 2. Data-oriented (and beyond) network architecture (DONA).

2.1.2. NDN

In named-data networking (NDN) [15], nodes handle three tables: a content store (CS), forwarding information base (FIB) and pending interest table (PIT). The NDN network architecture is presented in Figure 3.

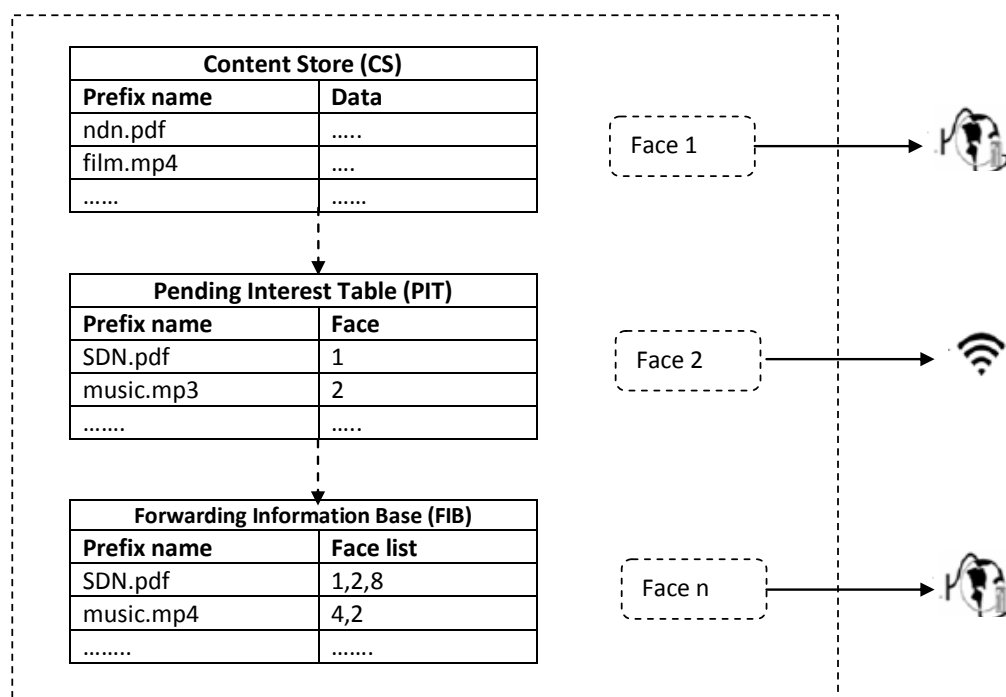


Figure 3. Normal data networking (NDN) node model.

Content naming relies on hierarchical names ciphered by segment information (or packet), the data digest, the publisher, the version number, and the content identifier. Indeed, NDN relies on two packet types: interest and data packets. When looking for content, an interest packet is sent within the network. The node, receiving this request, checks if an entry exists in its pending interest table (PIT). Then, it records the interface from which the request was received. Later on, the request is forwarded according to the information found in the forwarding information table (FIB). Therefore, if available, the response of the source consists of sending a data packet following the reverse path back to the requester using “breadcrumbs” left in a pending interest table on each router. Each node in the path removes the request from the PIT and may cache data according to its caching policy. Unlike DONA, during the routing process, NDN follows the late-binding method. In this approach, a content request to a specific location is barely resolved. Thus, there is no direct mapping between the identifier and locator dealt with before the request.

2.1.3. PSIRP

In the publish–subscribe paradigm [18] architecture, based on the publish/subscribe concept, we can distinguish three steps: rendezvous, managing the network topology and forwarding. The content is published to the network by the providers. The rendezvous system aims at finding a matching of a request. Then, according to the network topology, a route is created between the publisher and the subscriber.

2.1.4. NetInf

This network of information (NetInf) [14] architecture proposes decoupling name resolution (NR) from routing or coupling them as offered in NDN. Indeed, two objects are used in NetInf: (i) a data object representing a digital object, and (ii) an information object showing the real object. Similar to DONA, NetInf content names have the P:L form, in

which L denotes the label given to the content, and P stands for the publisher public key cryptographic hash.

2.2. ICN Problem

The major ICN [12] weakness consists of the fact that the number of name objects and that of content data objects are expected to be orders of magnitude higher than the number of the network nodes. Thus, carrying out name-based routing in each node in the path between nodes can be cost-demanding, which leads to bandwidth and delay increases. We suggest applying the SDN approach [8] to minimize the shortcomings posed by this problem. The SDN architecture can overcome some crucial ICN issues such as scalability and bandwidth consumption. It allows the data plane (forwarding plane) to be decoupled from the control plane (such as the routing decisions, intelligent control of network resources, and configuration of network traffic management policies) in the network. For this objective, a programmable entity (the SDN controller) should manage the switches and routers. These nodes become forwarding devices. This flexibility seemed to us to be suitable for improving NDN network performance.

3. Software-Defined Networking

Today, data communication networks are generally made up of hosts or user devices interlinked with the network infrastructure. The latter, shared by all the hosts, uses switching elements such as switches, communication links and routers in order to transfer data between the different hosts. Indeed, switches as well as routers are considered “closed” systems, having vendor-specific control and limited interfaces. Thus, it is somewhat hard for the current network infrastructure to be improved. That is to say, the major problem faced by the current network is that using new versions of existing protocols, such as IPv6, is very difficult [8].

SDN [8,19,20] is a promising and very flexible new network architecture that goes beyond the limitations of existing IP networks’ transmission mechanisms. This is because SDN networks establish a clear separation between the control plane and the data plane, where the two planes are physically separated. The forwarding devices (switch and router) have only the data plane, and the control plane is moved to a centralized entity (the controller). In SDN, forwarding devices (or switches) are considered virtual devices that only follow the redirection policies defined by an external programmable entity: the controller, which implements the control plane (Figure 4). Thus, it is no longer necessary to have multiple black-box network devices with specific network functions such as switches, routers, firewalls, etc. In SDN networks, any necessary changes in redirect policies are programmed only once in the controller, and the controller propagates the redirect rules to all SDN devices [8].

An SDN network is made up of four basic components: (i) a controller that implements the control plane; (ii) forwarding devices (switches) that own the data plane; (iii) a southbound interface, e.g., OpenFlow [9], which represents the communication channel between the controller and the routing devices; and (iv) a northbound interface, which allows the controller to communicate with the application’s network, defined by the user. SDN is a four-pillar network architecture:

- The planes of control and data are decoupled. The network devices are excluded from the control functionality, which makes it easy to switch objects.
- Transition decisions are based on the flow, not the destination. A flow is generally defined by a set of field values that function as a filter criterion and a set of acts (instructions). A flow is a series of frames between a source and a destination in the sense of SDN/OpenFlow. All the frames in a flow receive similar transfer equipment service policies. The abstraction of flow unites the actions of numerous items of network equipment, including routers, switches, firewalls and middleboxes. Flow programming provides unparalleled precision, restricted only by the capability of the applied flow tables.

- The control logic is passed on to the external party, the SDN controller, or the operating system of the network (NOS). The NOS is a software interface that draws on core server technologies and offers critical mechanisms and abstractions to logically promote forwarding system programming based on an abstract and centralized view of the network. Hence, its function is close to that of a traditional operating system.
- Via software programs running on the NOS, which communicate with the underlying data plane computers, the network is programmable. This is a basic function of SDN, perceived to be its key attribute.

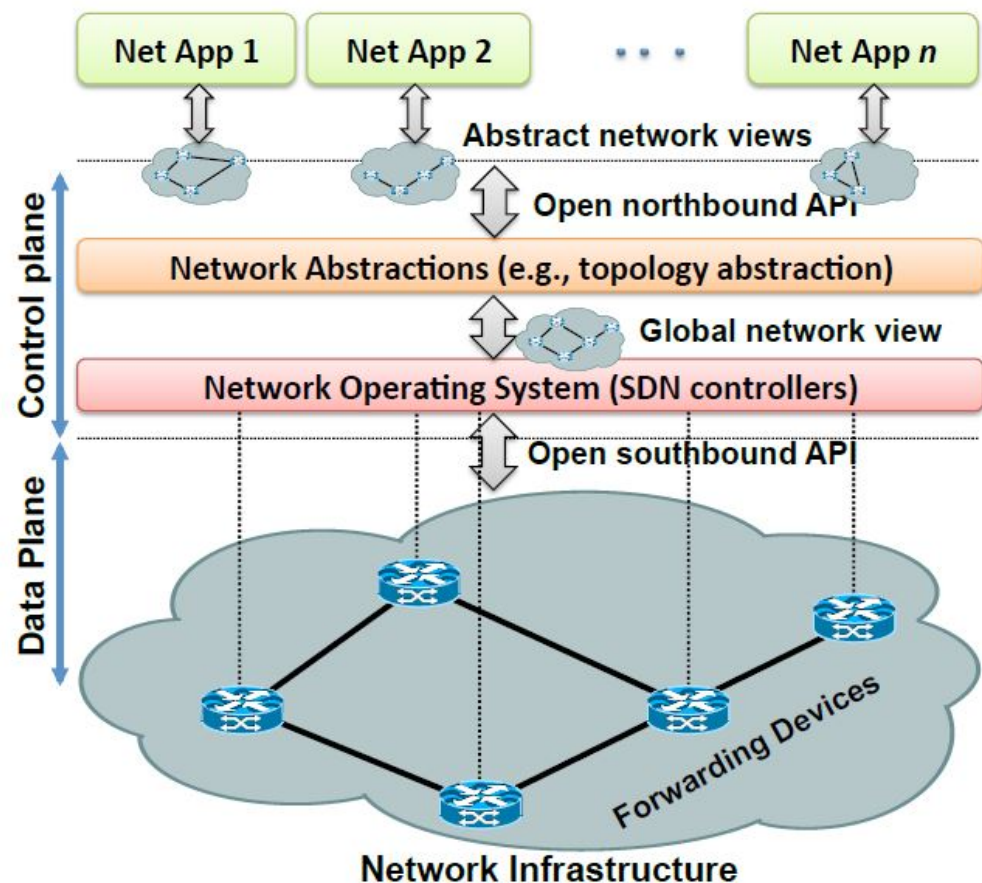


Figure 4. The software-defined networking (SDN) architecture decouples control logic from data logic.

3.1. Current Internet-Architecture Issues

The concept of Internet “ossification” [8] refers to the close coupling established between the control planes and that of data. In such an environment, using novel network functionalities or applications is absolutely important because their implementation in the infrastructure must be directly performed. Besides, because of the nonexistence of a common control interface for the different devices of the network, other straightforward tasks, such as policy enforcement and configuration, may be hardly achieved. Therefore, many workarounds such as the use of “middleboxes” (network address translators, firewalls, intrusion detection systems, etc.) were introduced and used to overcome the network-ossification effect.

The main objective of deploying software-defined networking (SDN) [8] is to make innovation easier and afford the network data path simple programmatic control. As shown in Figure 4, isolating the forwarding hardware from the so-called control logic leads to simpler, straightforward network management and visualization, and the easier use of novel applications, protocols and applications, as well as different middleboxes’ consolidation into software control.

3.2. Current SDN Architectures

In this section, we present two commonly used SDN architectures, called ForCES flow [21] and OpenFlow [22]. Both of them are based on the principle of differentiating the data and control planes. They assimilate the exchange of data between various planes. Nevertheless, technically speaking, these architectures are considerably different if we consider the protocol interface, design and forwarding model.

3.2.1. ForCES

The ForCES (forwarding and control element separation) IETF workgroup [21] is added. It reviews the internal design of the network interface with the control factor separated from the forwarding one. The network unit, however, remains a single entity. In addition, the driving usage case was represented within a single network system by the work group to connect third-party access with novel forwarding hardware. Both the control and data planes, such as the same room or box, would also be in close proximity. The control plane, however, is entirely separated into OpenFlow-like SDN structures from the network computer. In particular, two logic entities called the forwarding entity (FE) and control element (CE) are used for ForCES. All use the ForCES protocol to communicate.

Controlling and signaling functions are the responsibility of the CE. To tell FEs how to treat packets, the ForCES protocol is applied. This protocol is based on a slave–master model in which the masters are represented by CEs, while the slaves are denoted by FEs. The key building block of the powers architecture is the LFB (logical feature block). This block represents a well-determined functional block overseen by CEs through the ForCES protocol on the FEs. It enables the CEs to control FEs' setup and the latter to process packets. Indeed, the previously mentioned architecture was standardized in 2003 when the working group released several papers. The protocol for coordination between the control and forwarding components within a network component and a modeling language describing the logical functions within a forwarding component are among these papers [21].

3.2.2. OpenFlow

Similar to powers, this method standardizes the sharing of data between the two listed planes in accordance with the SDN concept, consisting of the distinction of data forwarding and control planes. The forwarding unit, also called the OpenFlow [22] switch, includes one or more flow tables and an abstract layer that uses the OpenFlow protocol to ensure safe communication with the controller, as seen in the configuration shown in Figure 5.

There are flow entries in the first component of the forwarding device (flow tables). Flow tables describe the manner in which packets of the same flow are stored and then passed. Flow entries are usually made up of the following: (1) Match fields, also referred to as matching rules, which are added to match the packets sent. They include metadata, packet headers and entry port data. (2) Counters used to capture information on individual flows, such as the flow length, number of packets sent and byte number. (3) A group of acts or orders that should be followed in a match. They show how they need to work with the corresponding packets. Packet header fields are drawn and then compared to the matching field sections of the flow table entries. The switch follows the accurate instructions or behavior collection relevant to the matched flow entry when a matching area is encountered. In addition, if no match is made by the look-up procedure of the flow chart, the switch takes action according to the directions determined by the table-miss flow entry. Each flow table must have a table-miss entry to accommodate table misses. The above determines the range of acts to be taken if no match, such as lowering a packet, is found for an incoming packet. The matching phase is continued on the next flow table or the packet is forwarded via the OpenFlow [22] channel to the controller from version 1.1. We note that OpenFlow manages pipeline processing as well as several tables. In addition, in the case of hybrid switches, nonmatching packets are exchanged using standard IP-forwarding schemes. In particular, the controller and switch communicate using the OpenFlow protocol, which specifies the collection of messages transmitted via a

protected channel from and to these entities. Flow entries in the flow tables of switches can be omitted, added or modified by applying the OpenFlow process, which can be achieved either proactively or reactively.

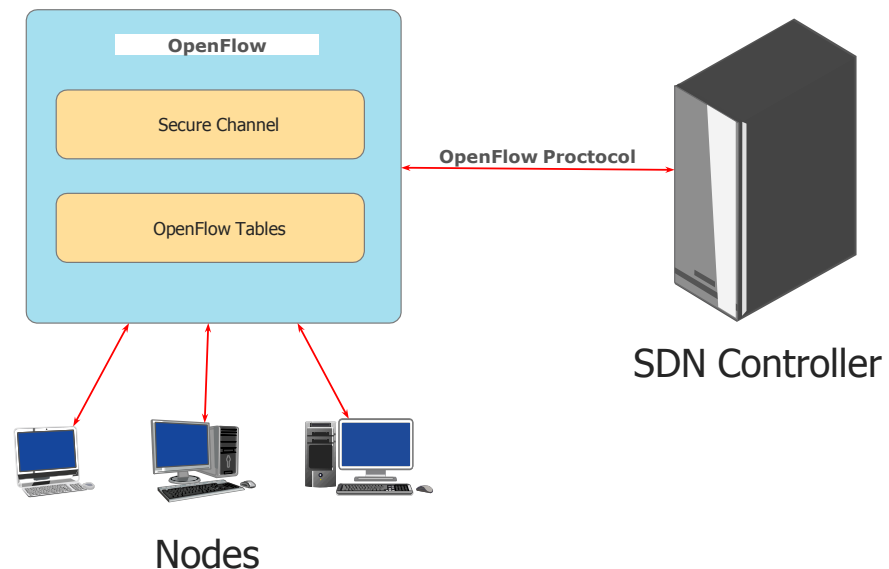


Figure 5. Communication between the controller and the switch via OpenFlow protocol.

4. Related Work

Much research work has tried to study the potential of combining SDN and ICN [23–25]. However, the majority of this work has applied SDN technology in NDN architectures. The principal approaches combining SDN and ICN are described below:

4.1. CroS-NDN

The CroS-NDN [23] approach isolates the control plane from the data plane. It reinforces the former on a dedicated node (controller). This consolidation enables the network nodes to store and request network data without being spread in the whole network. A controller-based routing strategy for NDN nodes dynamically stores network information on a controller. These nodes also reactively request novel routes for the controller according to the interests of the consumer to locally unknown name prefixes.

To minimize the overhead routing signaling, CroS-NDN limits the network interest flooding. Indeed, the different nodes of the network flood it merely to search for the controller. Then, the latter finds the flooding happening exactly upon the nodes for the controller's interest's timeout. The major functions of each CroS-NDN node are to control its one-hop neighbors and store all the variations in the controller. These nodes create the name prefixes of the locally produced content. However, a controller gathers the network data and obtains knowledge about the topology of the network as well as about the prefixes of the content names in order to form a node identifier map.

End-to-end CroS-NDN implementation considers only one controller route request. Indeed, the node of route requesting provides information about its identifier as well as the desired content name within the interest route to controller. On this route, the controller first determines the requesting nodes. Then, it locates that of the content producer. After that, it calculates the node identifiers' sequence path between the consumer and the producer and responds to the route request. Upon receiving the answer, the requesting node forms a specific interest that implements novel entries on each node via the network in the route towards the content producer. It is worth noting that the calculations of paths are based on node identifiers, while the content-request-interest forwarding depends on the content names. Clearly, content-mobility and topology variations can result in invalid FIB entries implemented on nodes. In fact, CroS-NDN utilizes data plane feedback to get

rid of these invalid FIB entries. Besides, non-answered data plane interests lead to the expiration of PIT entries after interest lifetime timeouts.

4.2. NDNFlow

NDNFlow[24] is an open-source software implementation of a forwarding architecture focused on named-data networking within OpenFlow-controlled software-defined networks (SDNs). As it is not feasible both to use application-specific IP broadcasting and to expand the OpenFlow [22] protocol to handle application-specific forwarding schemes in order to distinguish between IP and ICN traffic, NDNFlow has proposed a second application-specific layer to OpenFlow. Instead of the old OpenFlow communication channel and method, the latter generates a controller model and a separate communication channel.

Obviously, in this application, all path computations and communications, concerning the ICN, are dealt with independently of the common IP and Ethernet plane. In the OpenFlow approach's standardization, to have SDN functionality not being affected by limitations and changes, we must isolate the ICN layer from the regular OpenFlow layer.

4.3. SRSC

SRSC denotes the SDN-based routing scheme for CCN (SRSC) [25]. To establish a communication channel among the controller and nodes, this scheme utilizes both data messages and content-centric networking (CCN) [15] interest. Therefore, by applying this scheme in the case that no entry is allocated to the node in its FIB, the node transmits the interest to the controller. Thus, the latter must represent the whole network's general knowledge (paths, nodes, content stored in caches and topology). It can respond with a piggy-backed forwarding rule put into the data message, sent by the controller towards the node through the reverse path. Therefore, by using the SRSC, no network flooding will happen in order to obtain the content. Besides, the controller informs the nodes about routing the interest message towards the closest requested content. Its workload changes according to the network size and the node number. Therefore, a large-scale network must be divided into many domains managed by a controller.

We compare, in Table 1, the characteristics of CroS-NDN, NDNFlow and SRSC.

Table 1. Summary of characteristics of the SDN-based approach for ICN.

	CroS-NDN [23]	NDNFlow [24]	SRSC [25]
Clean-slate design	Yes, based only on NDN protocol	No, based on IP protocol	Yes, based only on CCN protocol
ICN protocol	NDN-based	NDN-based	CCN-based
SDN protocol	SDN-based	OpenFlow-based	SDN-based
Routing schema	controller-based	IP-based	Controller-based
Security	Signature-based	Signature-based	firewall-based

5. Named-Data Networking Based on SDN

Named-data networking based on SDN (NDN-SDN) implies that the network is split into many regions, consisting of NDN nodes and SDN controllers (Figure 6). An SDN controller and multiple NDN nodes are allocated to any region. The controller uses the OpenFlow [22] protocol to connect with NDN nodes.

We apply the bootstrapping stage and routing step to strengthen the NDN-SDN:

- **Bootstrapping step:** During this step, the controller identifies and associates the nodes with their identity. The nodes therefore know the interface on which their requests are relayed to their handler. Indeed, bootstrapping helps the controller to explore and provide a global view of the network's topology.
- **Routing:** The routing stage consists of filling in the nodes' FIB to allow them to relay messages of interest to the nearest source. Therefore, where a node does not know the

interface to which to send a message of interest, the controller is interrogated. This exchange is carried out using NDN interest/data messages via the contact channel (Figure 6).

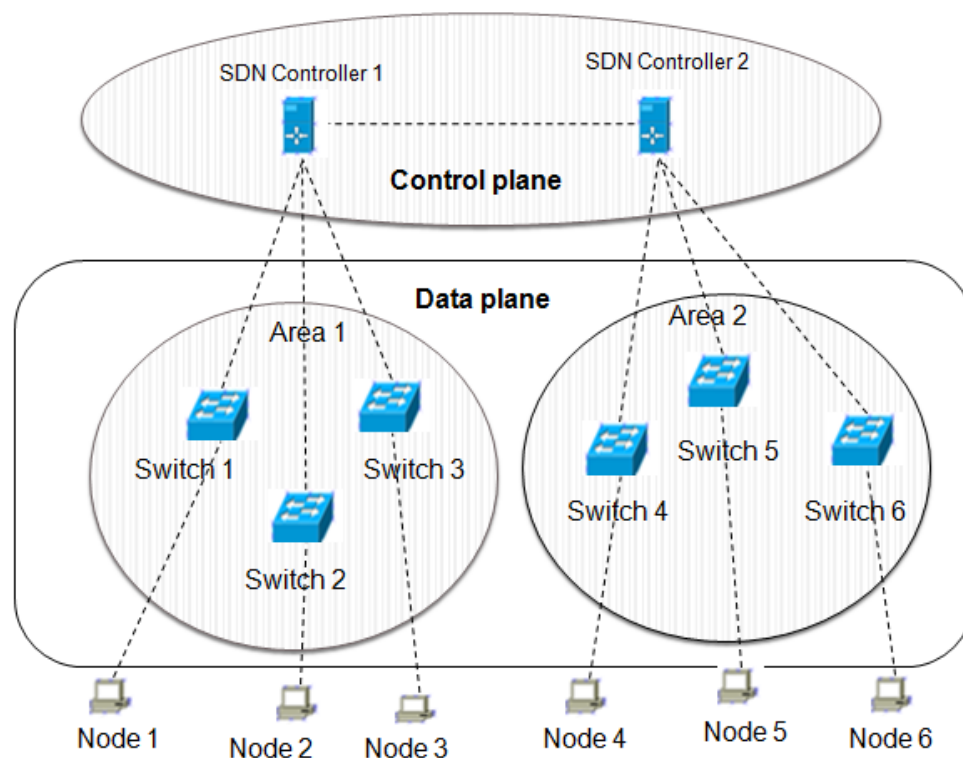


Figure 6. NDN-SDN architecture.

5.1. Structure of an NDN-SDN Switch

In our proposal, the NDN-SDN switch contains the following structure (Figure 7):

- The content store (CS): A cache installed in the NDN Switch.
- The pending interest table (PIT): The PIT table stores the currently-being-processed requests.
- Management information base (MIB): The MIB includes the neighborhood information (node, switch and controller).

5.2. Structure of an NDN-SDN Controller

In our proposal, the NDN-SDN controller contains the following structure (Figure 7):

- The content store (CS): A cache installed in the NDN-SDN controller.
- The management information base (MIB): The MIB table contains information about all the nodes connected to the controller.
- The data information base (DIB): The DIB includes all the prefixes of all the nodes connected to the controller.
- The forwarding information base (FIB): The FIB contains the routing information for the data.

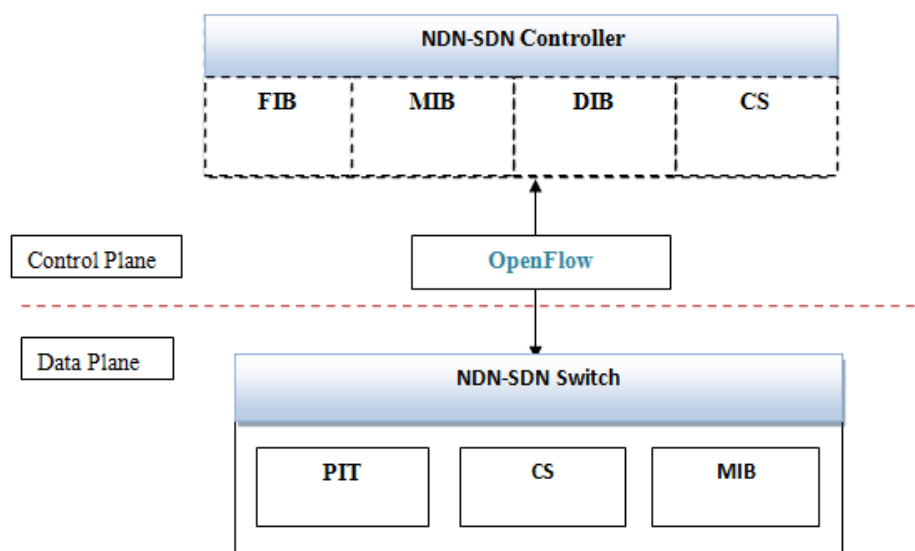


Figure 7. NDN-SDN structure.

5.3. NDN-SDN Topology Management

Initially, the switches do not know where the controller is located, and as a result, the switches flood all the interfaces with link layer discover protocol (LLDP) packets with the controller's name to discover the port to the controller. When a router receives a controller's response, it adds the port for communicating with the controller to its management information base (MIB).

Subsequently, all the nodes send a link layer discovery protocol (LLDP) packet to inform the directly connected neighbors of their presence (Figure 8). The nodes send periodic LLDP packets with their identifier ID on all the interfaces. Each node that receives an LLDP packet responds with an LLDP packet. Each switch constructs the local topology table MIB, which contains all the directly connected neighbor nodes, using these messages. After this step, the switch sends the MIB table to the zone controller.

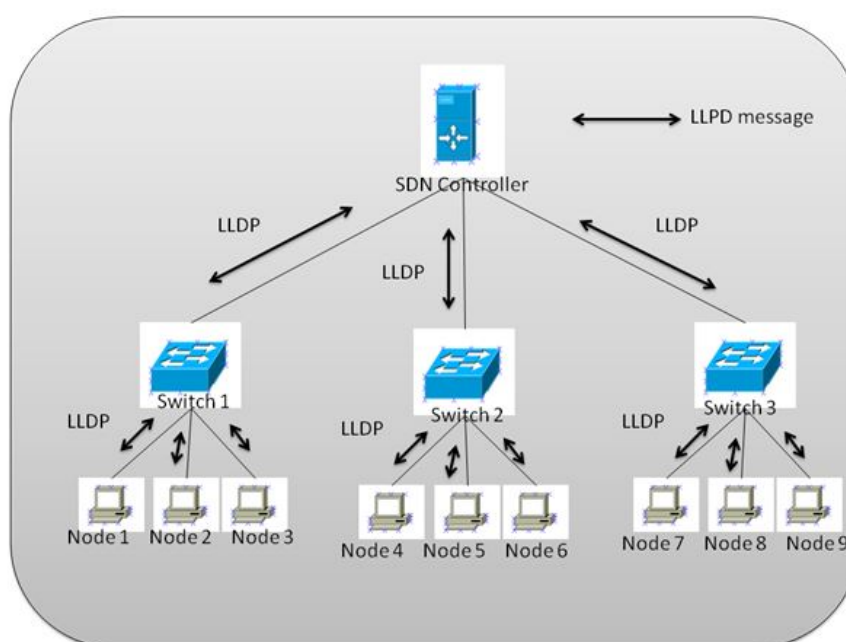


Figure 8. NDN-SDN topology management.

Upon receipt of the MIB tables from the switch, the controller constructs its MIB table. This table contains all the information of all the nodes that are related to the controller. It presents the virtual topology of the network. After this phase, the controller can install the routes and the forwarding rules on the switches because it knows all the routes to any switch on the network.

Unlike the traditional NDN network architecture, which does not have a topological control mechanism, our proposal allows the controller to have a global view of the topology, to control the network better in real time using regular packet exchanges.

5.4. NDN-SDN Data Management

When a node publishes new unregistered named data, it sends an “AddPrefix” packet to indicate the intent to register the prefix (Figure 9). When the switch receives this packet, the switch sends it to the controller. When the controller receives this prefix name, it constructs the data information base (DIB) table, representing the virtual topology of the data distribution in the network.

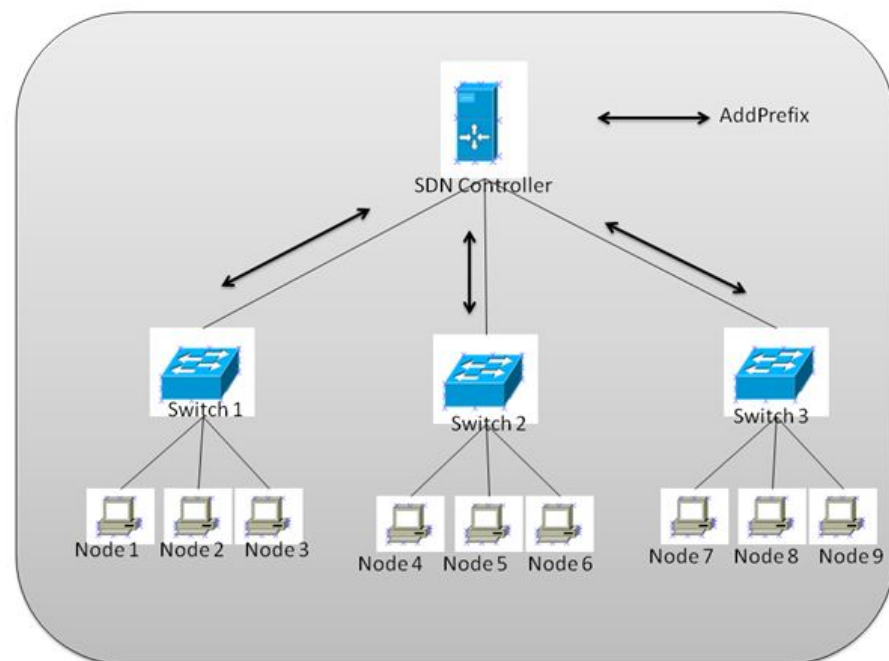


Figure 9. NDN-SDN data management.

5.5. NDN-SDN Forwarding Management

Based on the information available in both the DIB and MIB tables, the NDN-SDN controller constructs the forwarding information base (FIB). This table represents all the data available in this network zone and the path to follow to retrieve it. If a switch wants to retrieve a data packet, it sends an interest message to the zone controller; the controller looks for the prefix in the FIB table; if the interest exists, the controller returns a data packet that contains the path to retrieve these data.

5.6. NDN-SDN: Detecting Failures

We use the link layer discovery protocol (LLDP) to detect failures in the network topology. The NDN-SDN switch sends LLDP packets to adjacent switches. Suppose the switch does not receive an LLDP from the adjacent switch for a fixed period. In that case, NDN-SDN continues to send LLDP packets to detect this link’s recovery, but at a relatively long interval, to avoid a high message overload during a long outage duration. If there is no response, the switch determines that a link failure has occurred. The switch informs the controller. The controller then updates the FIB table.

6. Experimentation and Results

In this section, we discuss some experiments in order to demonstrate the advantages of using ICN's SDN architecture. NS-3 [26] and ndnSIM were used to run our experiments [26].

There were four examples we introduced. We added the NDN architecture to the first one. Indeed, each node has a 100-element content store (CS) capability and uses the caching technique LRU [27]. We used a flooding-based routing technique. The name of this case is NDN-Flooding. However, we changed the forwarding strategies in the second case, called "NDN-Best-Route", with the best-route strategies [28]. We substituted the routing strategies with the NCC strategies in the third example, "NDN-Ncc" [28]. The implementation of the last "named-data networking based on SDN" (NDN-SDN) scenario seeks to improve the architecture of the NDN using that of the SDN. For each 100 nodes, we placed one SDN controller, responsible for data routing. The correspondence between the nodes and the controller is carried out by the protocol OpenFlow [22]. Each node has a capacity of 100 components in the content store (CS) and follows the LRU caching strategy.

Table 2 presents the simulation parameters.

Table 2. Simulation parameters.

Parameter	Value
Router percentage	5%
Number of nodes	150, 300, 400, 600, 1000
Cache strategy	LRU
Forwarding strategy	Flooding, NCC, best-route
Topology	Hierarchical
Number of consumers	30%
Number of producers	10%
Distance between nodes	200 m
SDN protocol	OpenFlow
Time of simulation	120 s

We evaluate the NDN-SDN network's performance in all scenarios using traditional metrics such as the bandwidth consumption, average latency and cache-hit ratio.

- Bandwidth consumption: this is the cost of the communication generated by each switch on the network per second. The average value of all the content switches was taken into account.
- Average latency time: this is the time taken for complete content (that is to say, a file) to be downloaded. The average value of all the content routers was taken into account.
- Cache-hit ratio: this is the proportion of interest requests served by the content store (CS) of switches or SDN controllers. The average value of all the switches was taken into account in our work.

6.1. NDN-SDN Approach and Bandwidth

The number of packets per second, relative to the number of nodes, is shown in Figure 10. We note that, with the number of nodes, the number of packets increased. A distinction between the two NDN-SDN and NDN-Flooding situations is also noted. The use of NDN-SDN actually minimized the number of packets on the network. For instance, we detected a reduction of 19% in bandwidth usage with 400 nodes. The clear separation of the control plane from the data plane in our solution helped to minimize the bandwidth usage compared to that with NDN without SDN.

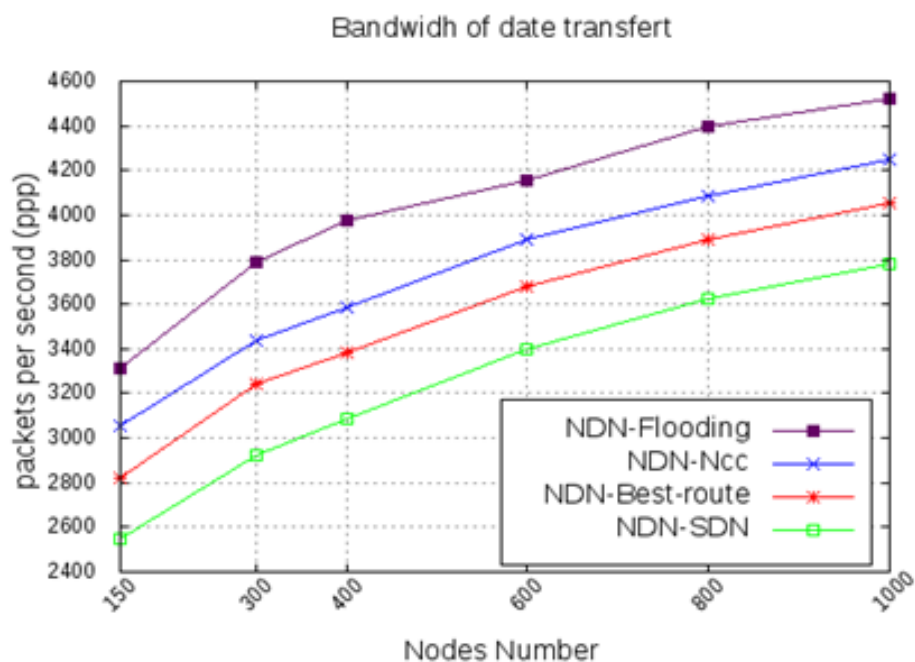


Figure 10. Bandwidth consumption.

6.2. NDN-SDN Approach and Delay

Figure 11 demonstrates how the SDN approach affects the output of NDN networks, especially the delay between sending interest and receiving results. For example, using the NDN architecture with SDN (NDN-SDN) and 400 nodes reduces the delay time by 27 percent compared to NDN without SDN (NDN-Flooding). This improvement is achieved due to the SDN controllers having a global view of all the data on the network. This global view minimizes the latency and increases the cache-hit rate.

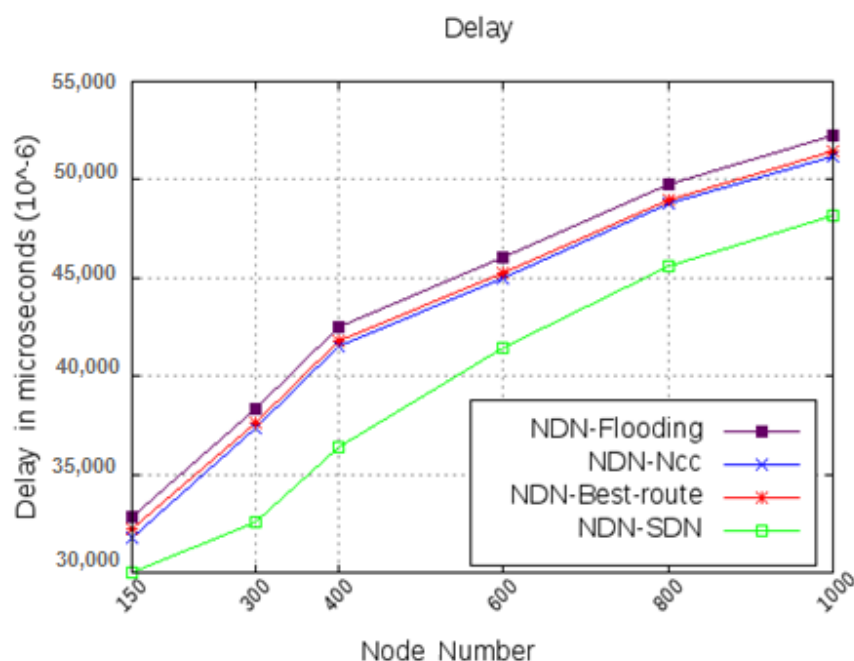


Figure 11. Delay time.

6.3. SDN Approach and Cache Hits

Figure 12 above illustrates how the SDN technique affects the network output of cache hits. For example, with 300 nodes, the use of the SDN (NDN-SDN) NDN architecture increases the percentage of cache hits by 29 percent compared to NDN without SDN (NDN-Flooding).

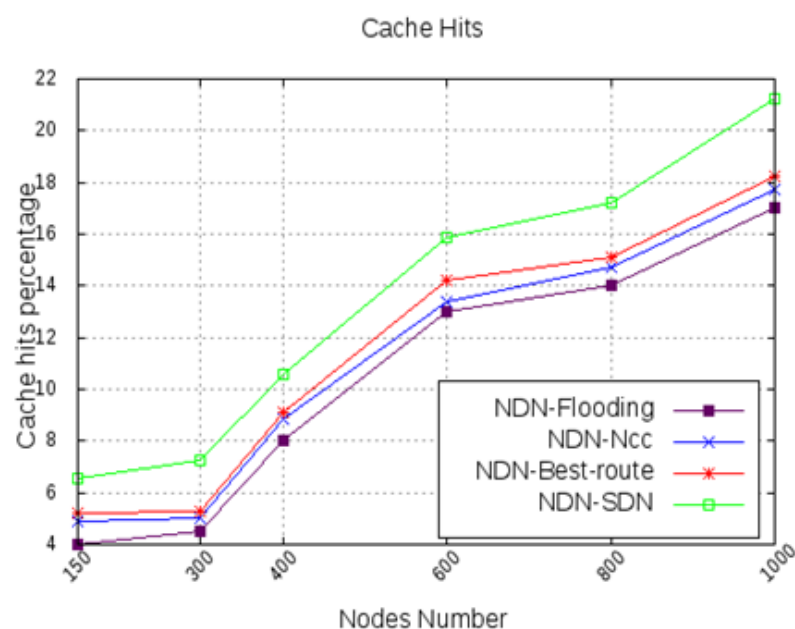


Figure 12. Cache hits.

6.4. NDN-SDN Approach and Memory Overhead

Figure 13 shows how the NDN network efficiency, particularly the overhead usage of memory, is affected by the SDN strategy. For example, with 1000 nodes, the use of the SDN (NDN-SDN) NDN architecture reduces the memory overhead by 9% compared to NDN without SDN (NDN-Flooding). In our solution, the clear separation between the control plane and the data plane minimizes the memory usage of the switches.

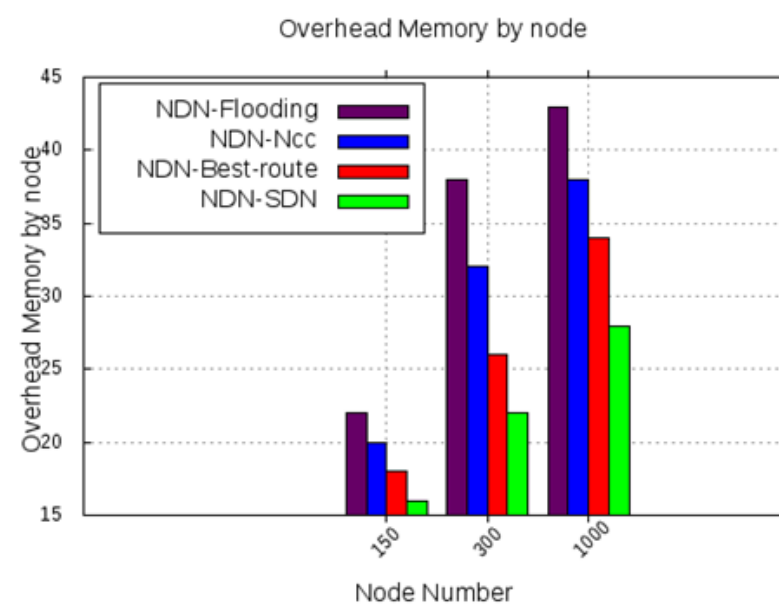


Figure 13. Memory overhead.

6.5. Discussion of Results

All the simulation results clearly show the positive effect of applying the SDN architecture on NDN networks. This exploitation permits increasing the cache-hit percentage and minimizing the use of bandwidth, delay and memory consumption. NDN-SDN also improves network caches' performance (cache hits), reducing the number of messages and minimizing the bandwidth requirements on the network, which is quite beneficial for the NDN architecture. Indeed, we chose to evaluate NDN-SDN with the NDN default cache strategy (LRU). Our improvement with NDN-SDN greatly reduces the required network resources such as the bandwidth, the delay and the consumption memory.

We note that NDN-SDN provides a 19% reduction in bandwidth usage compared to the NDN-Best-route strategy. This enhancement is explained by the separation between the data plane and the control plane, which minimizes the number of packets in the NDN network. Moreover, the NDN-SDN method reduces by 9% the consumption of switch memories compared to NDN-Flooding. This advantage is due to the separation of the control plane and the data plane, which allows minimizing the memory space use of the NDN-SDN switches. Furthermore, the NDN-SDN approach has the lowest latency time compared to NDN-Best-Route and NDN-Flooding approaches. The decrease in latency is explained by the use of our strategy, which minimizes the number of accesses to the FIB, PIT and CS tables.

In this proposal, we used the basic NDN routing and forwarding protocol. However, in [29], we used the NDNS protocol. The NDN-SDN method is more adaptive to the real Internet architecture because it preserves the properties of NDN. In fact, in the NDNS network, each switch has a set of FlowFib tables. The size of these tables is very large. Each switch must consult the three tables FlowFib, CS and PIT to find data, which considerably increases the latency time. In our NDN-SDN approach, the switch sends the packet of interest directly to the controller, which has a global view of the distribution of the data in the network. Then, the controller sends the shortest path to this "interest" for the switch, which requested these "data". Therefore, we can consider NDN-SDN to be an improvement of NDNS.

We compare, in Table 3, the characteristics of NDN-SDN, NDNS and NDN.

Table 3. A brief comparison between NDN-SDN, NDNS and NDN.

	NDN-SDN	NDNS [29]	NDN [28]
Latency	Very fast	Fast	Very slow
SDN-based	Yes	Yes	No
CPU switch consumption	Low	High	Very high
Bandwidth consumption	Low	Low	Very high
Detect failures	Yes	No	No

7. Conclusions

The evolution of Internet use from machine-to-machine communication to data-centered communication has urged us to shift the IP to the ICN network communication model. This advancement is causing many problems. Indeed, to enhance the networking architectures, software-defined networking (SDN) was proposed. In this paper, we analyzed the SDN strategy's effect on the ICN network. Our implementation demonstrates that the use of the SDN approach ameliorates the NDN network's performance. Indeed, it minimizes the latency and the use of bandwidth. The NDN-SDN enhancements result from the separation between the control plane and the data plane, reducing bandwidth consumption. Indeed, the SDN controller realizes an overall view of content distribution that allows latency reduction and a cache-hit-rate increase.

This paper has introduced SDN functionalities implemented in the NDN architecture to facilitate interactions between nodes, switches and controllers. Our study results show

that we can use SDN to improve NDN, which allows it to be deployed in a real network. As future work, we will suggest an NDN architecture focused on the use of SDN to enhance the ICN network's performance.

Author Contributions: Supervision and funding acquisition, T.G.; writing—original draft, review and editing, A.K.; methodology, B.M.A.; resources and validation, H.A. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of the Scientific Research of the University of Ha'il, Saudi Arabia (project: RG-20091).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Cisco. Cisco Annual Internet Report (2018–2023) White Paper. 2020. Available online: www.cisco.com (accessed on 27 February 2021).
2. Cisco. Cisco Annual Internet Report. 2020. Available online: www.cisco.com (accessed on 27 February 2021).
3. Belarmino, A.; Koh, Y. A critical review of research regarding peer-to-peer accommodations. *Int. J. Hosp. Manag.* **2020**, *84*, 102315. [CrossRef]
4. Salah, H.; Zimmermann, S.; Cabrera G., J.A. Chapter 5—Content distribution. In *Computing in Communication Networks*; Fitzek, F.H., Granelli, F., Seeling, P., Eds.; Academic Press: New York, NY, USA, 2020; pp. 93–103.
5. Fotiou, N. Information-Centric Networking (ICN). *Future Internet* **2020**, *12*, 35. [CrossRef]
6. Yu, K.; Eum, S.; Kurita, T.; Hua, Q.; Sato, T.; Nakazato, H.; Asami, T.; Kafle, V.P. Information-Centric Networking: Research and Standardization Status. *IEEE Access* **2019**, *7*, 126164–126176. [CrossRef]
7. Singh, S.; Jha, R.K. A Survey on Software Defined Networking: Architecture for Next Generation Network. *J. Netw. Syst. Manag.* **2016**, *25*, 321–374. [CrossRef]
8. Masoudi, R.; Ghaffari, A. Software defined networks: A survey. *J. Netw. Comput. Appl.* **2016**, *67*, 1–25. [CrossRef]
9. Isyaku, B.; Zahid, M.S.M.; Kamat, M.B.; Bakar, K.A.; Ghaleb, F.A. Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey. *Future Internet* **2020**, *12*, 147. [CrossRef]
10. Kalghoum, A.; Saidane, L.A. FCR-NS: A novel caching and forwarding strategy for Named Data Networking based on Software Defined Networking. *Clust. Comput.* **2019**, *22*, 981–994. [CrossRef]
11. Jmal, R.; Fourati, L.C. Content-Centric Networking Management Based on Software Defined Networks: Survey. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 1128–1142. [CrossRef]
12. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A Survey of Information-Centric Networking Research. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 24–49. [CrossRef]
13. Yovita, L.V.; Syambas, N.R.; Edward, I.J.M.; Kamiyama, N. Performance Analysis of Cache Based on Popularity and Class in Named Data Network. *Future Internet* **2020**, *12*, 227. [CrossRef]
14. Vasilakos, A.V.; Li, Z.; Simon, G.; You, W. Information centric network: Research challenges and opportunities. *J. Netw. Comput. Appl.* **2015**, *52*, 1–10. [CrossRef]
15. Saxena, D.; Raychoudhury, V.; Suri, N.; Becker, C.; Cao, J. Named Data Networking: A survey. *Comput. Sci. Rev.* **2016**, *19*, 15–55. [CrossRef]
16. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of Information (NetInf)—An information-centric networking architecture. *Comput. Commun.* **2013**, *36*, 721–735. [CrossRef]
17. Naeem, M.A.; Rehman, M.A.U.; Ullah, R.; Kim, B.S. A Comparative Performance Analysis of Popularity-Based Caching Strategies in Named Data Networking. *IEEE Access* **2020**, *8*, 50057–50077. [CrossRef]
18. Naeem, M.A.; Nor, S.A.; Hassan, S. Future Internet Architectures. 2020; pp. 520–532. Available online: https://www.researchgate.net/publication/336989259_Future_Internet_Architectures (accessed on 27 February 2021).
19. Cox, J.H.; Chung, J.; Donovan, S.; Ivey, J.; Clark, R.J.; Riley, G.; Owen, H.L. Advancing Software-Defined Networks: A Survey. *IEEE Access* **2017**, *5*, 25487–25526. [CrossRef]
20. Zhang, X.; Cui, L.; Wei, K.; Tso, F.P.; Ji, Y.; Jia, W. A survey on stateful data plane in software defined networks. *Comput. Netw.* **2021**, *184*, 107597. [CrossRef]
21. Haleplidis, E.; Salim, J.H.; Halpern, J.M.; Hares, S.; Pentikousis, K.; Ogawa, K.; Weiming, W.; Denazis, S.; Koufopavlou, O. Network Programmability with ForCES. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1423–1440. [CrossRef]
22. Lara, A.; Kolasani, A.; Ramamurthy, B. Network Innovation using OpenFlow: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 493–512. [CrossRef]

23. Torres, J.V.; Alvarenga, I.D.; Boutaba, R.; Duarte, O.C.M. An autonomous and efficient controller-based routing scheme for networking Named-Data mobility. *Comput. Commun.* **2017**, *103*, 94–103. [\[CrossRef\]](#)
24. van Adrichem, N.L.M.; Kuipers, F.A. NDNFlow: Software-defined Named Data Networking. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), London, UK, 13–17 April 2015; pp. 14–20.
25. 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.
26. Mastorakis, S.; Afanasyev, A.; Moiseenko, I.; Zhang, L. ndnSIM 2.0: A New Version of the NDN Simulator for NS-3. Technical Report NDN-0028. 2015. Available online: www.named-data.net/ (accessed on 27 February 2021).
27. Lal, N.; Kumar, S.; Kadian, G.; Chaurasiya, V.K. Caching methodologies in Content centric networking (CCN): A survey. *Comput. Sci. Rev.* **2019**, *31*, 39–50. [\[CrossRef\]](#)
28. Afanasyev, A.; Shi, J.; Zhang, B.; Zhang, L.; Moiseenko, I.; Yu, Y. NFD Developer's Guide. Technical Report NDN-0021. 2017. Available online: www.named-data.net/ (accessed on 27 February 2021).
29. Kalghoum, A.; Gammar, S.M. Towards New Information Centric Networking Strategy Based on Software Defined Networking. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017.