Enabling Virtual Network Functions in Named Data Networking

Puming Fang and Tilman Wolf

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{pfang,wolf}@umass.edu

***Abstract*—Virtual network functions, which are placed logically in the path of a network connection, implement important processing steps, such as content inspection, load balancing, etc. While these types of operations are well-understood in the context of the current, connection-oriented Internet architecture, it is unclear if and how such functions can be enabled in emerging architectures, such as Named Data Networking (NDN). In this paper, we argue that virtual network functions can provide benefits to NDN. We present a novel way to use transformations to names in NDN to invoke network functions. We show that network functions can be called either explicitly (e.g., by an end-system) or transparently by intermediate nodes (e.g., by an edge router). We believe that enabling virtual network functions significantly expands the capabilities of Named Data Networking.**

***Index Terms*—named data networking, virtual network function, network protocol, content inspection.**

I. Introduction

Data communication networks aim to transfer data from one end-system to another. While a purist view on such a system may assume that the data that is transferred is unaltered and networks operate in a data-agnostic manner, the current Internet shows us that these assumptions do not hold in practice: (1) data is modified inside the network, ranging from media transcoding for low-bandwidth links to advertisement insertion in web content and content inspection that may drop some data entirely; and (2) network systems use information about the data transfer to optimize operations for load balancing, preferential forwarding, etc. These *network functions* are used throughout the network and can be implemented directly on network nodes or traffic can be redirected to compute nodes that implement them as *virtual network functions*.

One of the reasons why such network functions are necessary for practical networks is that different entities operate portions of the network and each may use different internal operations or lack trust relationships with other entities. For example, a user who brings their own end-system to a campus or corporate network that allows BYOD (bring your own device) would like to be able to access any web content. The operator of that network, in contrast, wants to make sure that only "safe" content (i.e., free of malware) is accessed. This is particularly important since the BYOD policy does not allow the operator to enforce malware protection on the end-system. Thus, the operator's only option to ensure that their network remains malware-free is to perform content-inspection on any incoming data in a transparent manner and in the form of virtual network functions.

Named Data Networking (NDN) [1], [2] is an emerging approach to moving data communication away from connection-oriented networking and toward a content-centric perspective. In NDN, all data is represented as content that has a hierarchically structured name. The network fetches that content from the source or from any other node that can provide it (e.g., from its local cache). Data is passed node-to-node, and thus no end-to-end connections need to be established or maintained.

While a purist view of the NDN architecture may argue that network functions are unnecessary or even undesirable since a network should carry content without processing it, we argue in Section III-B that practical networks do need such capabilities. In addition, our approach of encoding the invocations of network functions in the names of data requests follows the principle of NDN, which aims to encode protocol information in the name of data.

The main idea of our approach is to have certain entities in the network, transform the names of requests for data in a way that network functions (which are represented as name prefixes) are invoked on the requested data. Nodes that offer virtual network functions advertise these "function prefixes" within the scope of their local network. With our design, only a minimal change in the protocol operation of an NDN node is necessary to enable the powerful functionality of network functions. We present how our Named Data Network Functions (NDNF) can be realized in three different scenarios: one, where the end-system explicitly requests a network function to be invoked, and two, where either the edge router or any intermediate router invokes the network function transparently. As with network functions in conventional networks, our approach is scoped to the local network (and thus not visible to other entities in the network) and network functions can be chained, e.g., to create service function chaining.

The specific contributions of our work are:

- We articulate the need to support virtual network functions in NDN.
- We present an approach that uses modifications to names in data requests to invoke network functions in a manner that is consistent with the architectural principles of NDN.

- We present three different approaches on how to invoke network functions in NDN, which include explicit invocation by the end-system and transparent invocation by intermediate nodes. We also present how network functions can be chained and be used in the conjunction with data caching.

The remainder of this paper is organized as follows: Section II briefly reviews related work. Section III discusses our approach to using network functions in NDN. Section IV presents the designs of our Named Data Network Functions. We discuss how our network functions can be used in specific scenarios in Section V. Section VI concludes the paper.

II. Related Work

Content-Centric Networking (CCN) has been developed as a communication model that reflects the shift in Internet usage from a host-centric end-to-end communication to receiver-driven content retrieval [1], [3]–[5]. NDN is a research project that has developed a network architecture based on CCN, as well as important related issues, such as security, performance, etc. [2], [6].

Network function virtualization (NFV) is used as a flexible implementation and management of network services [7]–[10] and general in-network processing operations [11]. So far, research on NFV in the context of NDN has mostly focused on using software-defined networks with NFV as a substrate for NDN prototypes. Reference [12] has demonstrated that a content-oriented orchestrator can automatically deploy a virtual NDN topology and handle dynamic reconfiguration. In [13], an NFV-compliant architecture was proposed that can deploy CCN islands and connect via HTTP/NDN gateways.

Recently, researchers developed methods to implement the functionality of NFV in the context of specific use scenarios: In [14], three scenarios for load balancing in NDN are proposed; in [15], smart routers are used to defend against Internet censorship in NDN by acting as proxy web servers and making use of intermediate caches. Our work differs from these previous approaches by providing a *general approach* to handling network functions in NDN. Our Named Data Network Functions for invoking network functions can be used for any type of network service in NDN.

III. Network Functions in Named Data Networking

Before discussing our Named Data Network Functions in Section IV, we briefly discuss the operation of NDN and how our work fits into the context of the overall NDN architecture.

A. Overview of Named Data Networking

NDN is regarded as one of the promising architectures of the future Internet, in which information is conveyed by distributing named content ("Data"). The network carries content packets to the entities that requested this content (via "Interests"). This approach is fundamentally different from the connection-oriented end-to-end connections used in the current TCP/IP protocol stack.

An NDN node consists of three components: a forwarding information base (FIB), pending Interest table (PIT), and content store (CS). The FIB contains routing information to forward Interests. PIT keeps all pending Interests and the arrival interfaces of the Interests into entries, and each entry is removed when the matching Data is received or a timeout occurs. The CS is used for caching Data packets with the caching replacement policy.

When a producer publishes content, the producer must first advertise the name prefix for name routing to the NDN network. Then, in order to retrieve content, a consumer specifies the desired content name, including the producer's name prefix in the Interest, and sends that request into the NDN network. The corresponding Data can be returned from the content producer or the intermediate NDN node with the cached Data. After receiving the Data and the content producer's signature, the consumer can verify the signature [16].

B. Network Functions and NDN Architecture

As we show in Section IV, it is possible to enable network function in NDN. A key question that needs to be addressed first, however, is if this capability is necessary and desirable. The NDN architecture envisions that data is distributed through the network in a way that is agnostic to the underlying realities of the network. For example, content can be fetched directly or obtained from caches, even untrusted caches. The end-to-end protection mechanisms that protect the data itself can be used to ensure confidentiality, verify authenticity, etc. This idealistic, architectural view has certainly led to a clean design of NDN.

In a practical network, however, there are multiple entities with sometimes conflicting interests that need to collaborate. In such cases, the straightforward implementation of NDN is not feasible or desirable. Examples are:

- Security: A corporate network may not allow access to content that may introduce malware on hosts connected to the network. End-to-end security may not prevent this problem and hosts may not follow corporate network policies. Thus, network functions for content inspection may need to be invoked transparently by the corporate network administrator, as discussed in Section V-A.
- Resources: A network may have limited bandwidth resources. End-hosts may not be aware of or sensitive to these limits. Thus, media transcoding services may need to be invoked on the content to still deliver a lower-quality version of, say, a video. We discuss this further in Section V-B.

These examples illustrate the network functions may still be desirable in practical NDN deployments. Therefore, we think that our Named Data Network Functions can provide important functionality, while still being based on the operational principles of NDN, as we see in the next section.

IV. Named Data Network Functions

In this section, we describe how we can adapt Named Data Networks to support the implementation of Named Data

Network Functions (NDNF). We describe the use of NDNF in the context of three specific usage scenarios:

- End-system NDNF: In this scenario, the end-system explicitly invokes the network function and thus is explicitly aware of the network function operating on the requested data.
- Edge-router NDNF: In this scenario, the edge router of a network invokes the network function. This scenario is similar to how traditional (non-virtualized) network functions are invoked. The end-system is not aware of the execution of the network function (i.e., the network function is transparent).
- Any-router NDNF: In this scenario, any router in the local network can invoke network functions transparently. This approach enables network functions for any type of traffic, even traffic that does not traverse the edge router.

The concepts used to instantiate network functions in these scenarios are similar but differ in which entities invoke the network functions.

A. Named Data Network Functions Principles

To communicate in NDN, an end-system requests content by sending an Interest packet containing the name of the data. We utilize this same naming scheme to invoke the network function. The end-system or an intermediary node can apply function f to data **/name**, i.e., f(**/name**), by adding the function name as a prefix to the data name, i.e., **/f/name**.

With this approach, we can use the standard NDN mechanism to request and deliver data. The only change is that the NFV node produces content with prefix **/f** by executing the network function on the content that is identified in the reset of the name.

Thus, the NDNF components can be summarized as:

- The name of the network function f is translated into a name prefix as **/f**.
- The NFV node that executes the corresponding network function advertises the prefix **/f** as content that it can provide. This advertisement is constrained to the local network (or the scope where the network function should be offered).
- The network function is invoked either by an end-system explicitly or by an intermediate node transparently.

For the three different scenarios of how to invoke the network function (end-system, edge router, any router), which we discuss in the following sections, we present the following three important steps:

- **Name prefix advertisement:** If the NFV node can implement a network function f, it advertises the prefix **/f** within the scope of its local network. Any requested data that asks for this network function is therefore forwarded to this node. This advertisement is not forwarded by edge routers and is thus constrained to the local network.
- **Network function invocation:** A particular node can add the function prefix **/f** to any Interest name, thereby requesting that network function f should be executed on the data.
- **Network function execution:** When the NFV node receives the Interest with prefix **/f**, i.e., **/f/name**, it translates the prefix into the requested network function f. Following that, the NFV node first retrieves the content from the producer (i.e., **/name**), then executes the network function (i.e., f(**/name**)), and returns the result to the entity that requested **/f/name**.

Note that NDNF can work with multiple different and concatenated network functions. For simplicity in our explanation, we describe NDNF using a single network function.

B. End-System NDNF

The simplest way to invoke a network function is to have the end-system perform the network function invocation. The end-system can easily encode the requested network function in the name. Figure 1 shows this scenario. The three key steps are as follows:

NFV node advertises the name prefix: NFV node advertises the "function prefix" **/f** within the scope of its local network.

End-system invokes network function: The end-system invokes the function of adding a prefix **/f** to the original Interest **/name**.

NFV node executes network function: After receiving the Interest **/f/name**, the NFV node removes the prefix **/f** and translates it into the requested network function f. Following that, it retrieves the Data **/name** from the producer. Then, the NFV node implements the network function on the received Data. Finally, the processed Data is sent to the end-system. The actions of the NFV node are listed in Algorithm 1.

Algorithm 1 Actions of NFV Node

Require: NFV node advertises the name prefix **/f**.

```
1: if Received packet is Interest packet then
2:     if Interest contains the name prefix /f then
3:         remove the prefix /f and then send out the Interest
   to request Data
4:     else
5:         forward the Interest
6:     end if
7: else if Received packet is Data packet then
8:     execute network function and then send out the Data
9: end if
```

This scenario is the easiest way to invoke a network function. However, this approach requires the end-system to explicitly invoke the network function. This requirement may not work well in situations where the network functions needs to be called transparently by intermediate nodes (e.g., for network management or network security operations that the end-system is unaware of). Therefore, we proposed Edge-Router NDNF in the following section.

C. Edge-Router NDNF

In this scenario, the network function is called transparently by the edge router. The complete process consists of the eight steps shown in Figure 2. These steps are as follows:

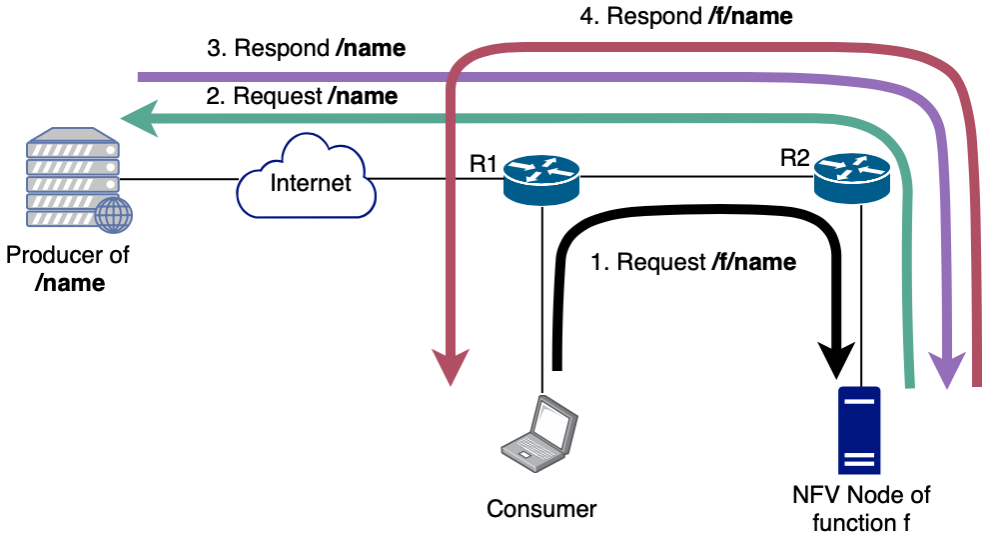

Fig. 1. End-system invokes network function.

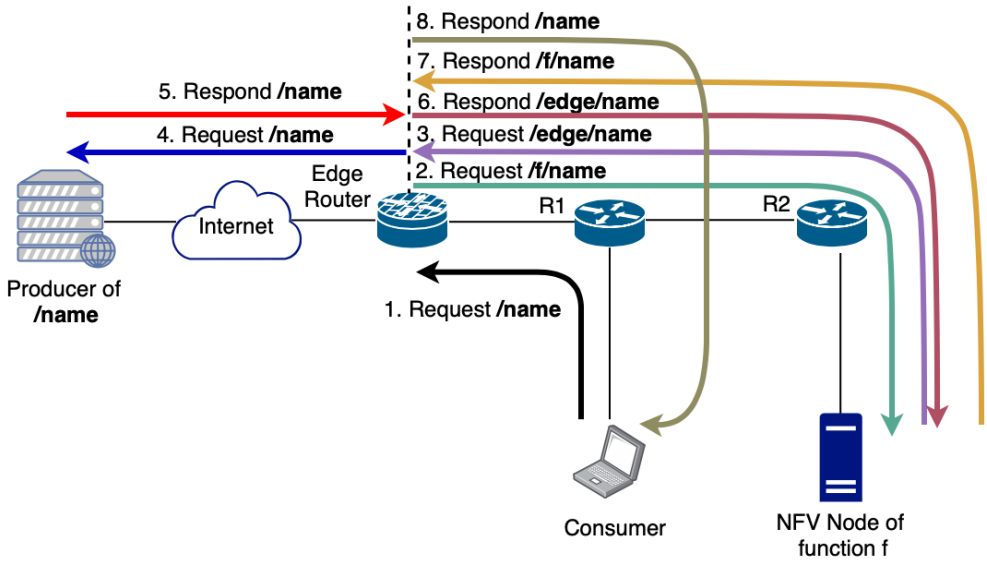

Fig. 2. Edge router invokes network function.

NFV node and edge router advertise name prefixes: NFV node advertises the "function prefix" **/f** within the scope of its local network. The edge router advertises the prefix **/edge** to create a "tunnel" for the NFV node to reach the edge router (and thus have Interest packets be distinguishable from requests from regular end-systems).

Edge router invokes network function: The edge router invokes the network function by adding the prefix **/f** to the Interest. (This can be done on any Interest, except those that start with **/edge**.) Interests starting with **/f** are sent back to the NFV node. The operation of the edge router is shown in Algorithm 2.

NFV node requests content via tunnel to edge router: To avoid invoking network function repeatedly after the edge router receives an Interest from the NFV node, NFV node uses its tunnel to the edge router (by adding the prefix **/edge**). The edge router requests this content from the producer and forwards it to the NFV node.

NFV node executes network function: NFV node, as shown in Algorithm 3, translates the name prefix **/f** into the requested network function f. The node executes this function f on the Data received via the edge router. The processed Data is returned to the edge router (which in turn sends it to the requesting end-system).

This scenario is very suitable for scenarios that invoke network functions transparently, such as content inspection. However, this approach requires traffic to go all the way to the edge router before the network function is invoked. Thus, the edge router may become a bottleneck or single point of failure. To address this concern, we also explore a scenario where the

Algorithm 2 Actions of Edge Router

Require: Edge router advertises the name prefix **/edge**.

if Received packet is Interest packet **then**
  if Interest contains no prefix **then**
    add prefix **/f** and then send out the Interest
  else if Interest contains prefix **/edge then**
    remove the prefix **/edge** and send out the Interest to request Data from the producer
  end if
else if Received packet is Data packet **then**
  if Data contains no prefix **then**
    add prefix **/edge** and then send out the Data
  else if Data contains prefix **/f/edge then**
    remove prefix **/f/edge** and then send out the Data
  end if
end if

Algorithm 3 Actions of NFV Node

if Received packet is Interest packet **then**
  if Interest contains prefix **/f then**
    Remove the prefix **/f**, add a prefix **/edge** for tunneling communication and send out the Interest
  else
    forward the Interest
  end if
else if Received packet is Data packet **then**
  execute network function and then send out the Data
end if

network function can be invoked by any intermediate router in the local network.

D. Any-Router NDNF

In this scenario, the network function is invoked by an intermediate router in the local network. Figure 3 illustrates this scenario in detail.

NFV node and edge router advertise name prefixes: NFV node advertises the "function prefix" **/f** and the edge router advertises the prefix **/edge** to create a "tunnel," as in the edge-router NDNF scenario.

Intermediate router invokes network function: After receiving the Interest **/name** from the end-system, the intermediate router modifies the prefix to **/f/name** to invoke the network function. The operation of the edge router is shown in Algorithm 4.

NFV node requests content via tunnel to edge router: To avoid invoking network function repeatedly after intermediate routers receive an Interest from the NFV node, NFV node uses its tunnel to the edge router (by adding the prefix **/edge**). The edge router requests this content from the producer and forwards it to the NFV node.

NFV node executes network function: NFV node, similar to the edge-router NDNF scenario shown in Algorithm 3, translates the name prefix **/f** into the requested network function f. The node executes this function f on the Data

Algorithm 4 Actions of Intermediate Router

if Received packet is Interest packet **then**
  if Interest Packet contains no prefix **then**
    add prefix **/f** and then send out the Interest
  else if Interest Packet contains prefix **/edge then**
    forward the Interest towards edge router
  end if
else if Received packet is Data packet **then**
  forward the Data
end if

received via the edge router. The processed Data is returned to the intermediate router (which in turn sends it to the requesting end-system).

While this scenario is very similar to the edge-router NDNF, the ability to "intercept" a packet at any point in the network and invoke the network function improves efficiency. The load on the edge router is reduced and unnecessary back-and-forth communication is avoided.

E. Enabling Multiple Virtual Network Functions

All of these scenarios can be used with multiple virtual network functions. Different NFV nodes can advertise one or more network functions (which are distinguished by different names). Entities that can invoke network functions can invoke a chain of n NFV functions f_1,f_2...f_n by using the name prefix **/f$_1$/f$_2$/.../f$_n$/name**. All the requested network functions are executed in the order $f_1(f_2(...f_n($**/name**$)...))$ before Data is sent to the requester.

V. NDNF Usage Scenarios

We briefly describe two usage scenarios that apply Named Data Network Functions in a specific context.

A. Content Inspection

Malicious content, such as malware, can exist in NDN, even when information security properties, such as authentication, can be ensured. A corporate network may want to enforce policies that protect connected end-systems from accessing such content. End-systems in BYOD environments, as described in Section I, might not actively invoke content inspection network functions.

Instead, edge-router NDNF or any-router NDNF can provide an effective solution. A network function for content inspection can be installed on the NFV node, and edge routers or intermediate routers can invoke this network function in accordance with corporate network policies. An important aspect of this scenario is that the **/edge** tunnel prefix needs to be disabled for end-systems to avoid that an end-system can bypass content inspection.

B. Video Transcoding

In live events broadcasts, the network may be limited in bandwidth, especially in outdoor events (e.g., bicycle race). Transcoding services can be used to reduce the bandwidth of video streams. Edge-router NDNF or any-router NDNF can

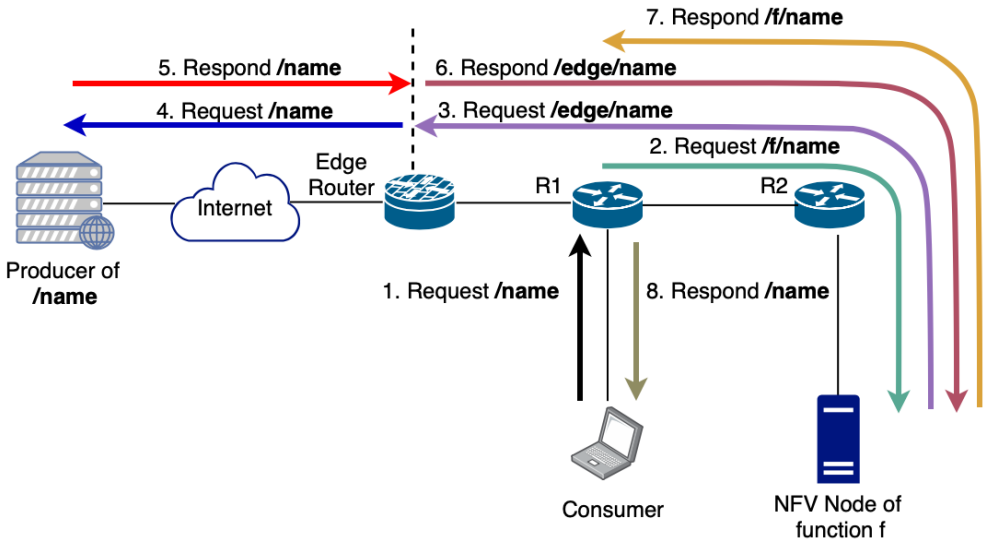

Fig. 3. Intermediate router invokes network function.

invoke these network functions based on network conditions in a matter that is transparent to the end-system. The NFV node that provides transcoding service advertises the network function and routers can invoke them on demand (and under changing network conditions). The end-system continues to receive an appropriately compressed video stream despite changes in the available bandwidth.

VI. Summary and Conclusions

In this paper, we have presented Named Data Network Functions, a technique to invoke virtual network functions in Named Data Networking. We discussed how network functions are necessary for practical networks, even NDN. We presented our approach, which is based on the principles of NDN, i.e., the embedding of protocol information in names, and demonstrated its use in the context of three different scenarios for invoking network functions. We also discussed how advanced uses involving function chaining and caching can be realized. We believe this work represents an important step toward making NDN practically useful in today's complex networking ecosystem.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT)*, Rome, Italy, Dec. 2009, pp. 1–12.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, Jul. 2014.

[3] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[4] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE communications surveys & tutorials*, vol. 16, no. 2, pp. 1024–1049, 2013.

[5] X. Qiao, H. Wang, W. Tan, A. V. Vasilakos, J. Chen, and M. B. Blake, "A survey of applications research on content-centric networking," *China Communications*, vol. 16, no. 9, pp. 122–140, 2019.

[6] A. Afanasyev, I. Moiseenko, L. Zhang *et al.*, "ndnsim: Ndn simulator for ns-3," *University of California, Los Angeles, Tech. Rep*, vol. 4, 2012.

[7] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[8] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[9] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[10] G. Mirjalily and Z. Luo, "Optimal network function virtualization and service function chaining: A survey," *Chinese Journal of Electronics*, vol. 27, no. 4, pp. 704–717, 2018.

[11] T. Wolf, "Service-centric end-to-end abstractions in next-generation networks," in *Proc. of Fifteenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Arlington, VA, Oct. 2006, pp. 79–86.

[12] H. L. Mai, M. Aouadj, G. Doyen, W. Mallouli, E. M. de Oca, and O. Festor, "Toward content-oriented orchestration: Sdn and nfv as enabling technologies for ndn," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 594–598.

[13] X. Marchal, M. El Aoun, B. Mathieu, T. Cholez, G. Doyen, W. Mallouli, and O. Festor, "Leveraging nfv for the deployment of ndn: Application to http traffic transport," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–5.

[14] D. Mansour, H. Osman, and C. Tschudin, "Load balancing in the presence of services in named-data networking," *Journal of Network and Systems Management*, 2020.

[15] X. Cui, Y. H. Tsang, L. C. Hui, S. Yiu, and B. Luo, "Defend against internet censorship in named data networking," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2016, pp. 300–305.

[16] D. Kondo, T. Silverston, V. Vassiliades, H. Tode, and T. Asami, "Name filter: A countermeasure against information leakage attacks in named data networking," *IEEE Access*, vol. 6, pp. 65 151–65 170, 2018.