Sharing digital object across data infrastructures using Named Data Networking (NDN)

de Jong, K.; Fahrenfort, C.; Younis, A.; Zhao, Z.

DOI
10.1109/CCGrid49817.2020.00013

Publication date
2020

Document Version
Author accepted manuscript

Published in
20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing

Sharing digital object across data infrastructures using Named Data Networking (NDN)

Kees de Jong, Cas Fahrenfort, Anas Younis, Zhiming Zhao
Multiscale Networked Systems
University of Amsterdam
Amsterdam, the Netherlands
kees.dejong@surfsara.nl, casfahrenfort@live.nl, anas.younis@os3.nl, z.zhao@uva.nl

***Abstract*—Data infrastructures manage the life cycle of digital assets and allow users to efficiently discover them. To improve the Findability, Accessibility, Interoperability and Re-usability (FAIRness) of digital assets, a data infrastructure needs to provide digital assets with not only rich meta information and semantics contexts information but also globally resolvable identifiers. The Persistent Identifiers (PIDs), like Digital Object Identifier (DOI) are often used by data publishers and infrastructures. The traditional IP network and client-server model can potentially cause congestion and delays when many consumers simultaneously access data. In contrast, Information Centric Networking (ICN) technologies such as Named Data Networking (NDN) adopt a data centric approach where digital data objects, once requested, may be stored on intermediate hops in the network. Consecutive requests for that unique digital object are then made available by these intermediate hops (caching). This approach distributes traffic load more efficient and reliable compared to host-to-host connection oriented techniques, and demonstrates attractive opportunities for sharing digital objects across distributed networks. However, such an approach also faces several challenges. It requires not only an effective translation between the different naming schemas among PIDs and NDN, in particular for supporting PIDs from different publishers or repositories. Moreover, the planning and configuration of an ICN environment for distributed infrastructures are lacking an automated solution. To bridge the gap, we propose an ICN planning service with specific consideration of interoperability across PID schemas in the Cloud environment.**

***Index Terms*—Named Data Networking, Persistent Identifier, Digital Object Interface Protocol, FAIRness, Network Functions Virtualization**

I. Introduction

Data infrastructures manage the life cycle of data assets, and allow users to effectively discover and utilize data for their specific purposes. Typical examples include EuroArgo[1] and SeaDataNet[2] for ocean observation data, Integrated Carbon Observation System Research Infrastructure (ICOS)[3] and Aerosol, Clouds and Trace Gases (ACTRIS)[4] for air monitoring data, and European Plate Observing System (EPOS)[5] for solid earth monitoring. Those data infrastructures (sometimes also called research infrastructures) become important facilities for enabling data centric research, and can significantly enhance the feasibility for research at the system level, e.g., for studying global climate change and natural disasters.

However, to effectively discover and utilize resources across different infrastructures, users still face the challenges of limited FAIRness of digital assets. It becomes a common problem for many research infrastructures to improve their FAIRness; several projects have been funded for this purpose. The EU recently funded project ENVRI-FAIR[6] is such an effort for improving the FAIRness for more than 10 environmental research infrastructures. In this context, the globally resolvable identifier (also called (PID)) of digital objects and their rich contextual metadata are often highlighted as important aspects, as recommended in the 11 principles of FAIRness[7].

Moreover, the classical client-server, or P2P solutions are lacking support for distributing digital objects from heterogeneous repositories, in particular for handling continuously growing data quantity, sources and involved users (network congestion). Protocols like the Digital Object Interface Protocol (DOIP) have also been recommended by the community of Research Data Alliance (RDA) to handle the sharing of digital objects. However, it is still challenging to share data with a PID among different infrastructures. This is due to the incompatible nature between the different schemas used in the PID standards.

Furthermore, network technologies such as Software Defined Networking (SDN) [1] provide application aware adaptability. While Network Functions Virtualization (NFV) [2] may provide efficient resource management and sharing over common network infrastructures. However, these advanced technologies still lack seamless support for being embedded in the data management life cycle, e.g., for discovering and sharing digital objects with PIDs. Koulouzis et al. discussed the feasibility to use ICN solutions like NDN to distribute the digital objects with PIDs. However, the work did not solve the challenge of the incompatible (heterogeneous) nature between the PID standards, and only implemented the PID schema from a single publisher [3]. Cloud caching [4] and data lake [5] are other types of solutions which focus on smart caching of the digital objects across distributed infrastructures, but did not have an explicit model of the data contents global resolvable

[1]https://www.euro-argo.eu/
[2]https://www.seadatanet.org/
[3]https://www.icos-ri.eu/
[4]https://www.actris.eu/
[5]https://www.epos-ip.org/
[6]www.envri-fair.eu/
[7]https://www.force11.org/

identifiers.

On the other hand, network technologies have made significant progress during the past years for improving distribution efficiency, e.g., Content Delivery Networks (CDNs) [6], BitTorrent [7] and NDN [8]. Like NDN, a CDN also places content closer to its users. This is done by caching data in multiple geographical locations which, in a CDN, are reachable over IP (often via anycast). In NDN, data is routed by name instead of IP. NDN has another advantage over IP, NDN is able to use multiple paths and unlike IP it is able to handle loops on the forwarding layer. A loop-free topology is realized by inserting a nonce in every interest packet (query) which allows to identify duplicate interests for named data, allowing multiple paths to be used which may increase throughput efficiency. Information about paths in the network are maintained in a layer called the strategy layer. This layer keeps track of two-way traffic and changes local forwarding decisions based on traffic observations. The term 'face' is used in NDN to describe a connection to a forwarder, which may be of any class of the underlay (e.g. IP, Bluetooth, etc.). Furthermore, similar to NDN, CDNs may serve as a mitigation for Distributed Denial-of-Service (DDoS) attacks [9].

BitTorrent tries to determine the best peers based on trail-and-error. The protocol 'chokes' and 'unchokes' peers in order to guess the best quality peers to download from. This method is applied because unlike NDN, BitTorrent has no knowledge of the network topology and routing policies. Furthermore, data can only be verified in the application layer. Unlike BitTorrent, NDN has the ability to detect corrupt data in the network layer (instead of the application layer) by the use of cryptographic hashes. Therefore, corrupt data is not forwarded once corruption is detected and thus saving bandwidth.

In this paper, we extend our earlier work of NDN-as-a-service for PID data objects (NaaS4PID) [3], and specifically look at three aspects in applying NDN in data infrastructures: 1) how to seamlessly publish version controlled digital objects with a PID, 2) how to plan and deploy a customized virtual NDN environment for user communities on different infrastructures, and 3) how to distribute digital objects from distributed data sources with heterogeneous PID publishers. We will use a case study from SeaDataNet and structure the paper in five parts. First we will briefly introduce the pilot use case, and review the state of the art and related work. And then we will present the proposed solution and demonstrate its usage based on the use case.

II. Problem background, challenges and related work

A data infrastructure often aggregates several data sources, and provide their metadata via catalogues for users to precisely discover data assets. To deliver high quality services to the user community, an infrastructure like SeaDataNet often has to face several challenges:

1) Fault tolerance challenges. SeaDataNet originally only gathers metadata from remote repositories, and the user has to access the individual remote repositories[8] to obtain data. Such client-server style not only creates high load for the repository but also network traffic congestion.
2) Performance challenges. A Cloud replica has been created in SeaDataNet to duplicate each remote repository in the Cloud. In this way, the clients can directly retrieve the copy of data assets from the Cloud server. However, such a solution also needs careful customization for load-balancing and service placement to reach the required quality of service or user experiences.
3) Scalability challenges. SeaDataNet faces the continuous growth of both data providers and the user communities. In many cases, geo special data retrieval needs calculation of the region, layers and the time duration. The system has to face scalability challenges of both network and the server capacity.

In the rest of the section, we will review the related work of these topics and discuss the key contributions of our work.

A. Persistent Identifiers

PIDs are in essence a permanent reference to a document, file, web page or data object. Generally, a PID is assigned to a digital data object on request of the client, after which the service guarantees that whenever this identifier is resolved, it will provide necessary meta information about the digital object. PIDs do not guarantee that the digital object will be available forever; a PID should still be resolvable (to the metadata) even if the digital object has been deleted.

PIDs often have different 'namespaces' which can be used to point to different sub-resolvers, much like how URLs can have different sub-domains. These namespaces can be used to allow a single resolver to resolve multiple top-level PIDs to a number of distinct resolvers which have their own PID resolution system and separate repositories for storing data. This structure allows multiple different organizations to resolve data using a single PID scheme, as long as they agree to uphold the persistence policies set by the top-level resolver. Typical schemas of PID include DOI, Uniform Resource Name (URN) and Persistent Uniform Resource Locator (PURL). PIDs are often provided by the third party registers, like the European Persistent Identifier Consortium (ePIC), DataCite, ORCID and ISNI.

B. PID objects over ICN

Karakannas proposed a mapping architecture for resolving PIDs to ICN names by the use of a mapping server, and for delivering big data with PIDs [11].

Mousa researched the fetching and sharing of DOI objects with ICN such as NDN. The researcher's approach focused only on DOI digital objects within NDNs. The researcher explained that the difference in NDN naming of different PID providers must be taken into account, such that the correct

[8]We distinguish repository and registry based on the Digital Object Architecture; registry is for storing metadata while the repository is for storing the data objects. They can often be combined [10].

prefix is used within NDN to identify specific PID types. In the researcher's design, the translation happened in the consumer's browser. The consumer has the choice to either request the digital object by its NDN name or PID [12].

Kolouzis et al. continued the research done by Karakannas [11] and Mousa [12] and proposed an architecture to map PIDs into the naming schema of NDN. A PID2NDN gateway was proposed for resolving PIDs to NDN names, an NDN node that implements a virtualized NDN router, and an NDN4PID manager for automating the management of the NDN overlay in Cloud or e-infrastructure.

C. Named Data Networking

When the Internet was conceived it was designed for host-to-host communication. This means that in order to retrieve data, a host needs to retrieve the data from an IP address. Nowadays the Internet is increasingly data-oriented rather than host-oriented. When many data consumers request the same content, congestion may occur. Data locality sits on top of that problem, since data needs to cross distance as well, resulting in latency. Several technologies and methods have been developed to assist in efficient data distribution to improve performance.

NDN is a popular ICN implementation, although it is not yet implemented on a large scale. There have been several large testbeds established during the past years, e.g., for scientific research in delivering climate data [13], and the hadron collider network [14]. From those practices, NDN provided performance improvements compared to classical data delivery techniques based on IP. In those work, cache size played an important role in optimising performance, and a 1GB NDN cache at the edge of the NDN can already significantly improve data distribution and reduce the network load. However, native NDN congestion control is still an open research area [15].

Koulouzis et al. investigated different strategies for caching, replacement and forwarding for different size data objects, and concluded that the 'leave copy everywhere' caching strategy provided the best performance ratio between cache size to digital object size for generic data usage [3]. However, 'leave copy down' and 'leaving copies with probability' performed best for delivering big data objects. Koulouzis et al. also concluded that the ascending ordering of digital objects enhances network performance when combined with the 'least recently used' caching strategy. Yuan et al. [16] used the Content-Centric Networking (CCNx) application[9] to investigate NDN forwarding strategies, and concluded that packets with long names often degraded performance. Tortelli et al. researched the effectiveness of two opposite forward strategies; flooding and best-route (with and without caching) [17]. Several experiments lead them to the conclusion that there are pros and cons in each forwarding strategy. But that it is difficult to determine the best performing forward strategy.

[9]https://wiki.fd.io/view/Cicn

Those practices and performance characteristics provide valuable information for designing and customising an NDN environment.

D. Network planning and customization

Planning and customizing the topology and capacity of a network is crucial to meet the requirements of applications. McCabe applies a systems methodology approach towards network design [18], and identifies three core phases; analysis, architecture and design. These simple, yet important planning phases describe how to make technology and topology decisions in a network, especially for large deployments. These decisions are guided based on inputs, the initial input may be from users and/or from network metrics. Then the analysis phase determines the relationships between users, applications, devices and networks and translates that into a flow analysis. The architecture and design phase determines a high-level network design and implementation plan. This helps to determine the location of the NDN nodes across (global) data centers by the use of cloud orchestrators, as will be discussed in more detail later.

Using SDN and other programmable network technologies, developers can control the flows via the service interface provided by the network control plane [19]. Using the Cloud environment, the virtual networks can be fully customized and manipulated based on the application constraints. In [20] [21], Wang et al., proposed a critical path based approach to plan a virtual infrastructure, including virtual machines, their topology, and the placement of SDN controllers. Those work were so far mainly for the traditional IP based networks. The planning solutions for an ICN network is still quite limited.

E. Summary

From the discussion, we can clearly see the advantages of using ICN technologies in distributing digital objects; however, an ICN has not yet been widely deployed as a physical infrastructure, except experimental testbeds. The PIDs of digital objects are crucial to make digital assets FAIR. Seamless integration between NDNs and the heterogeneous PID repositories are still a challenge due to the different PID schemas in use.

In this context, we specifically highlighted three challenges in setting up ICN environments in the Cloud: i) complexity of the data publishing and PID assignment, ii) automation of virtual NDN environment planning and provisioning and iii) interoperability among heterogeneous PID schema and the NDN namespace. To tackle those challenges, we will bring the follow three contributions:

1) Provide services to enable data managers to seamless publish data objects and obtain PIDs during the data management life cycle.
2) Provide a network planner to plan virtual NDN environments based on the characteristics of the data sources and user access patterns.
3) Develop a PID interoperability service in the NDN for handling different PID schemas.

III. Proposed solution

An *on demand NDN environment manager for data infrastructures, or Named Data Networking for Data Infrastructures (NDN4DI)*, is proposed to facilitate the application of NDN in the context of data infrastructures. Fig. 1 shows the basic idea. The system aims to provide support for 1) a data manager to publish data objects and obtain the PID for them, and 2) plan a customized NDN environment at Cloud providers, and 3) automate the provision and deployment of the planned environment. After the NDN environment is in operation, the NDN4DI also provides functions for handling PID repositories from different publishers.

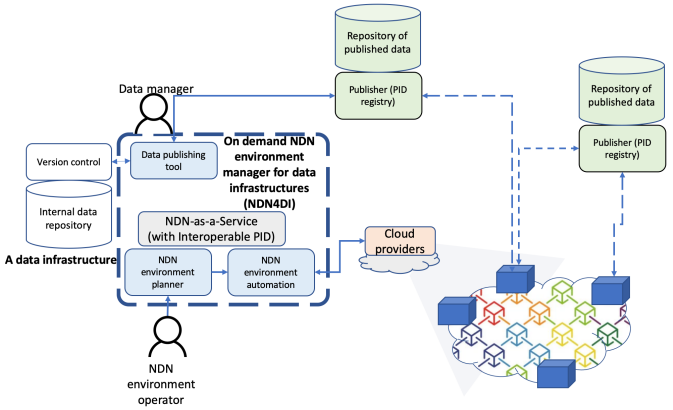

Fig. 1. Basic idea of the proposed on demand NDN planner solution for data infrastructures

A. PID assignment during data management

Lots of existing data infrastructures are originated from early legacy systems, e.g., environmental observation stations, and lack of a global data identifier centered design for data services.

Following from the workflow in Fig. 2, it is divided into three distinct parts: Version control system, data publication and data distribution. Content created by community content providers is processed and stored (in its internal repository) by the version control system in any form that is required by its specific use case. Once submitted content has been approved, it is published to persistent data and metadata repositories by any PID schema. From there, the data can be discovered and accessed by content consumers through the NDN by the use of the NaaS4PID service.

B. Support multiple PID repositories

To distribute data objects with PID over NDN, we have to handle the interface between NDN nodes and the repositories of PID data objects. The PID to NDN gateway is the key component, as shown in Fig. 3. The general idea is that a user enters a PID of the digital object that the user wants to retrieve at the client and gets back the requested digital object. The retrieval of a digital object depends if it is already published in the NDN or not. The PID to NDN gateway

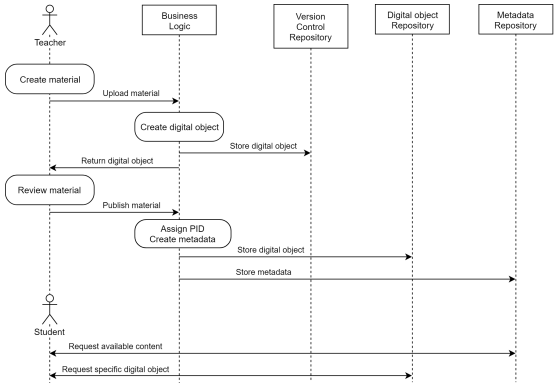

Fig. 2. A basic scenario for publishing data objects from a version control system.

implements the translation of different PID types and sends the translated name back to the client. Furthermore, we identify PID types based on pattern matching as described by Mousa [12]. PID metadata can be used to substitute missing fields in the PID URN in order to create an NDN compatible hierarchy. Research done by Olschanowsky et al. was used for deriving NDN names from metadata [22].

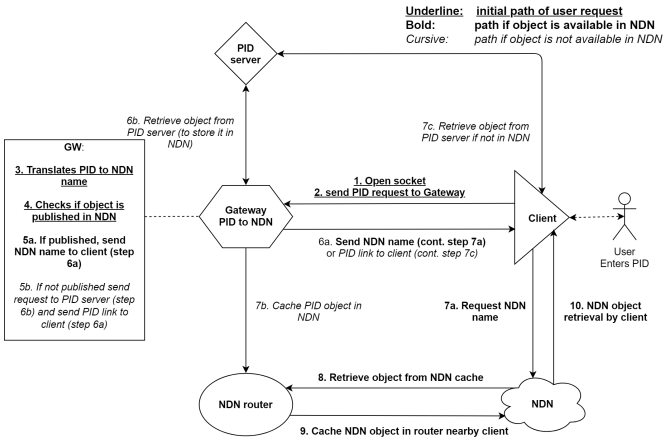

Fig. 3. Virtual NDN functions for achieving PID interoperability

The gateway is responsible for translating a PID to an NDN compatible name. If the digital object is already available in NDN, then the gateway sends the translated NDN name back to the client. The client then retrieves the digital object from NDN. If the digital object is not available in NDN, then the gateway sends back the PID link to the client and caches the digital object in NDN for future requests.

We implemented the Handle PID type, the URN PID type schema of the national library of the Netherlands, as well as the DOI type schema of PANGAEA. Based on pattern matching of the PID type schema[10], the gateway detects what kind of PID type it has to translate to NDN. Then, when a PID type matches, the associated function is called to translate the

[10]https://github.com/AquaL1te/rp2/blob/master/Scripts/pid_server.py#L58-L62

PID to an NDN compatible name[11]. The matching patterns of most standardized PID type schemas are documented in the ePIC Data Type Registry (DTR)[12], and can be implemented in the gateway to support a wide range of PID types.

C. NDN on virtual infrastructures

The NDN planning component models the NDN function as containers, and plans an NDN environment based on the VM based virtual infrastructure. Fig. 4 illustrates the basic idea.

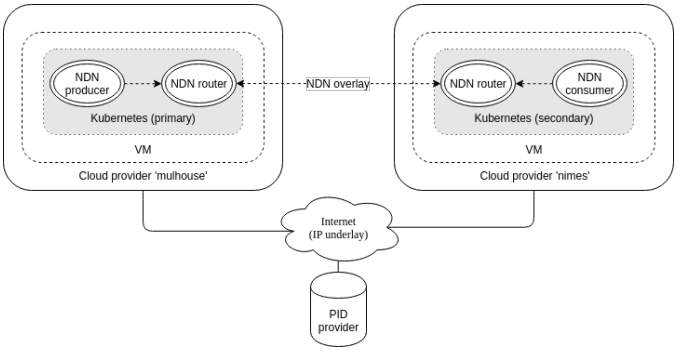

Fig. 4. High-level network design.

The virtual functions for NDN nodes are as follows, the producer is assigned the function to make data available in the NDN. The consumer is assigned to request data from the producer. However, in NDN, any node that has named data, can reply to interest packets. So the producer and consumer functions can be interchangeable. The router's function is to forward interest packets between the two cloud providers.

The planning procedure is designed based on the basic principle in [18], but in the following steps:

1) Select the proper size of the virtual machine, based on the location of the data sources and the users.
2) Based on the typical size of the data objects, estimate the optimal size of the cache of each node. This has to be bound to the total budget planned for the environment.
3) Place the NDN function on the nodes e.g., NDN gateways, and the NDN routing functions.

A virtual NDN environment is thus a set of networked virtual machines, with the deployment of containers for NDN functions. The description of the virtual machines, topologies, and the NDN functions are described using a standardised language called Topology and Orchestration Specification for Cloud Applications (TOSCA) [23]. TOSCA provides a rich set of elements, e.g., nodes, relationships and interfaces, to describe the basic structure of the virtual machines. Relationships between the virtual machines and the NDN functions may be enforced with TOSCA primitives such as 'dependsOn', 'hostedOn' and 'connectsTo'.

Interfaces are used to control the life cycle of a component and consist as a set of hooks to trigger actions, these actions are create, configure, start, stop or delete. These hooks can be triggered to e.g. configure and create containers, stop or start a service or do system maintenance such as delete artifacts after a service is stopped. An abstract example is shown in Fig. 5.

In Fig. 5, a TOSCA diagram is illustrated. This diagram represents an abstract template description of the TOSCA relationships, in which the gray rectangular boxes are the core scalability factors. The scaling properties are highlighted in the rectangular areas. The left area, highlighted as 'scaling in/out resources' contain a dependency chain of the virtual NDN functions.

D. NDN automation

An NDN is typically distributed geographically. Therefore, deploying NDN on global Cloud providers allows a broad distribution. The heterogeneous nature in Cloud environments can make deployment automation and management complicated. The NDN4DI automates the provisioning of the TOSCA description in the virtual NDN environment using Dynamic Real-time Infrastructure Planner (DRIP) [24], an engine developed by the same team in earlier projects. Kubernetes is employed to automate the orchestration of the containerized NDN functions.

In Fig. 5, a VM needs to be provisioned first (step 1), before a pod (container) can be deployed on Kubernetes (step 2 to 5). This is described by the 'dependsOn' relationship. Furthermore, with the requirements defined, input constraints are described. These constraints are used by the orchestrator to make sure that the NDN infrastructure has sufficient resources available to operate. Once a VM is deployed, the dependency for Kubernetes is satisfied, thus Kubernetes can then be setup (step 2). Kubernetes can then deploy pods by the use of interfaces (step 3). These interfaces feed the containers with environment variables such as the gateway, a list of routes, the transport protocol for NDN, the NDN strategies and on which Kubernetes node this pod should run. The environment variables are given to the interface via the TOSCA inputs. These environment variables are then used by scripts that run inside the pods to setup NDN. Several constraints are set for these environment variables such as which valid transport protocols can be used for NDN, which NDN strategies are valid and which nodes are available. These constraints are defined with e.g. 'valid_values' or 'greater_than' definitions. These constraints help to guide the orchestrator to verify the inputs that are given for the template description. As illustrated in the second gray area 'scaling in/out the application', several pods can be instantiated (step 5a, 5b and 5c) from the image (step 4). These pods enable the virtual NDN functions. These pods establish the NDN and therefore are connected via the 'connectsTo' relationship. This network expands over to other Kubernetes nodes in the cluster by the use of the Kubernetes built-in overlay network.

IV. System prototype

The data publishing tool is prototyped as a service, which provides an interface for both user interaction (Fig. 6) and

[11] https://github.com/AquaL1te/rp2/blob/master/Scripts/pid_server.py#L17-L37

[12] http://dtr-test.pidconsortium.eu/

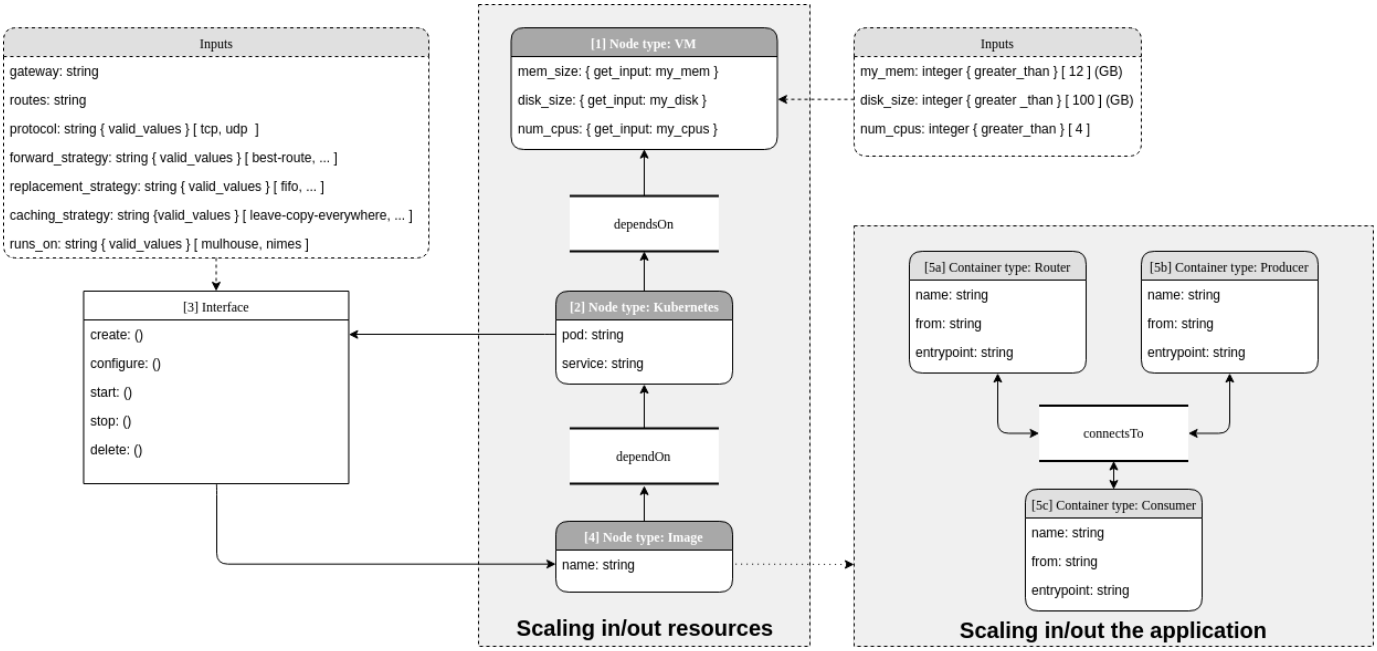

Fig. 5. TOSCA diagram.

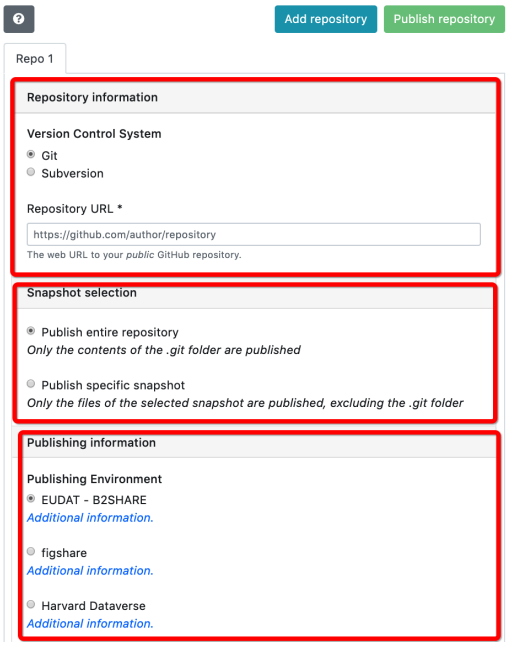

Fig. 6. Screen capture of the publishing tool.

a RESTful API. Via this interface, a user can configure an expected external repository, select a data snapshot from the version control system, and perform the publishing operation to obtain a PID for it.

The orchestrators mentioned, when combined with a TOSCA parser are still in an experimental phase. Therefore, in our prototype we deployed the 2 VMs and Kubernetes nodes manually. In practice the life cycle of also the Kubernetes pods are managed by a TOSCA orchestrator. Without having a TOSCA-ready orchestrator available, steps 2 through 5 in Fig. 5 were be carried out by Kubernetes exclusively. This was done by defining the configuration properties of the pods manually[13]. These properties include the NDN function name, e.g. router, producer or consumer. And also includes the routes (NDN prefixes) and the associated NDN face with the transport protocol to use (TCP or UDP). These parameters were then inserted into the NDN Forwarding Information Base (FIB) by the scripts that were executed inside the pod[14]. The NDN strategies were also configured by these scripts. Furthermore, the PID to NDN name translation prototype (section III-B) is containerized as NFV pods in our deployment prototype (NDN4DI. This enables flexible scaling by the use of Cloud providers.

A. Performance characterstics

The current prototype of NDN4DI has been benchmarked in a test environment (Fig. 4). We ran the following performance tests within our proof of concept using a 10MB, 100MB, 250MB, 500MB and a 1000MB digital data object size. We performed the performance tests ten times for each digital object size and protocol. The different digital object sizes were chosen in order to determine if there is a certain trend between the digital object size and performance. The performance trend is illustrated in Fig. 7 and shows the average of the test runs. We can observe that NDN over UDP outperforms the TCP/IP connection used for retrieving data objects from the

[13]https://github.com/AquaL1te/rp2/blob/master/Kubernetes/expanded-cluster.yml

[14]https://github.com/AquaL1te/rp2/blob/master/Docker/producer/docker-entrypoint.sh

Handle PID server that we setup with all chosen object sizes. Furthermore, NDN over TCP outperforms NDN over UDP. This result correlates with the research done by Lim et al. [13]. Fig. 7 illustrates that the performance converges with a 250MB digital object. We can therefore conclude that the relative performance difference between NDN and TCP/IP becomes smaller with object sizes larger than 250MB. Research done by Oran concluded that this observation is due to NDN's nature of handling big object sizes. Which may cause a performance problem due to the cost of retransmission when interests are retransmitted (or re-issued) [25].

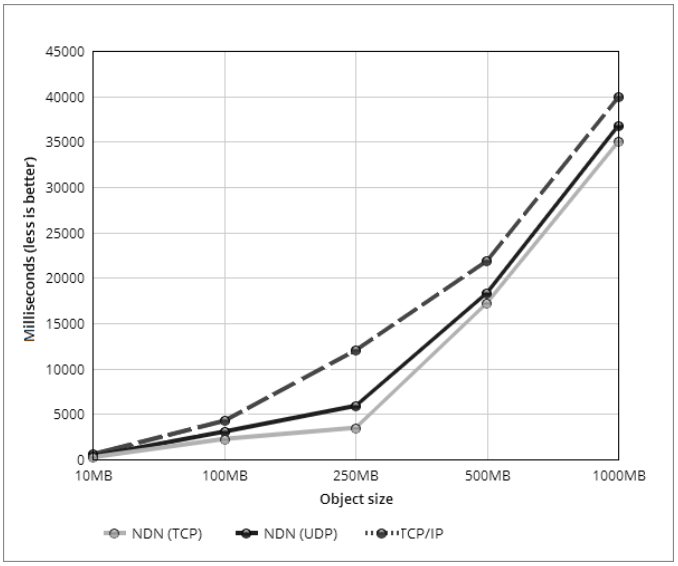

Fig. 7. Object size to performance relation.

The results gathered were merely based on best-effort test scenarios and are inconclusive. Therefore, further and more detailed research is required.

B. A SeaDataNet case study

Taking the data management scenario of SeaDataNet described in II as an example, the proposed NDN4DI solution can be utilized in several ways.

1) Fault tolerance challenges. By using the on demand NDN environment, the digital data objects from the distributed repositories will be cached based on the access frequency in the Cloud. The planner can be used to compute and optimize cache sizes, and the topology of the NDN nodes. In this case, the single point of failure of the remote repository can be avoided.
2) Performance and scalability challenges. Using the containerized NDN function over a virtual infrastructure, nodes and the containers can be scaled out efficiently. We have not demonstrated the software in this paper; however, the feasibility of auto-scaling of the virtual machine and the auto-configuration of NDN nodes can clearly realise it.
3) PID of data objects. The data publishing tool can be utilized by the remote data sources to automate the publishing of their data products. In the context of data infrastructures, a PID is often not a technical question; the most difficult part is for the community to agree on a standard. In this paper, we did not discuss that part, but only focused on the technical aspects in automating the publishing workflow.
4) With NDN4DI, multiple PID types can be integrated into the NDN. Furthermore, by utilizing NDN, network load may be more distributed by the use of in-network caching, which also lowers latency for the data consumers. NDN4DI allows flexible scaling by the use of NFV and Cloud providers. This flexibility mostly stems from the central point of orchestration which enables an infrastructure to adjust for higher network loads.

V. Summary

In this paper, we discussed the data sharing challenges in the data infrastructures and proposed an NDN based solution (NDN4DI) to tackle the challenge. We presented three key components in the NDN4DI system, and discussed their prototypes based on the PID, Cloud and NDN technologies.

A. Discussion

We adopted a system based network planning approach, based on McCabe's method [18] to create a high-level design of the NDN in TOSCA. By using the TOSCA standard, deployments in TOSCA-ready Cloud providers are possible with a uniform deployment description. This flexibility allows the data distribution network to scale easily and make management uncomplicated. However, TOSCA-ready Cloud providers are still rare, for our prototype to be more relevant, a wider adoption is needed.

Kubernetes was utilized to keep a central control over the NDN. However, if these pods do not share the same persistent cached data, cache misses may occur, which results in performance degradation. Cache misses are expensive since they require an update of the cache, which puts load on the original publisher of the data. Therefore, pods preferably are configured with persistent data volumes. Kubernetes can also load-balance requests between a set of identical pods.

The NDN configuration is still complex; a generic solution is still lacking, which was due to the immature NDN routing protocols. However, there are two promising routing protocols in development; Open Shortest Path First for NDN (OSPFN) by Lan Wang et. al. [26] and Named-data Link State Routing (NLSR) [27]. With a routing protocol, the NDN management process would become less complicated and more resilient.

Furthermore, in data infrastructures, identification services are used and utilize different PID schemas. Our prototype offers better integration between the identification and the data transmission services. However, our prototype exists outside of the NDN source code. For the best application of this functionality, we recommend the integration of these interoperability functionalities into the native NDN source code. This would remove the need to run a translation gateway.

Large data infrastructures are in general a federated architecture, where each federation is responsible for its own budget and infrastructure. However, our prototype assumes central control over the NDN. Our prototype could be approached in several ways. The discussed solutions could be deployed as an internal data-sharing platform per infrastructure and interconnect those NDNs, thus maintaining the federated model. Or, it could be deployed as a third party data-sharing platform, where one can deploy and operate the NDN for multiple infrastructures. Our research did not explore these subjects. However, it provides the flexibility to deploy these solutions in such a manner.

B. Conclusions

From the work, we can conclude that Cloud environments provide elastic resources for planning and customizing NDN based environments for the distributed data infrastructures to share data objects. The automated planning, provisioning and deployment of NDN environments are important to improve the usability of NDN in distributed data infrastructures. The proposed NDN4DI solution moves towards that direction.

C. Future work

The work will be continued with case studies with real operational data infrastructures. First, a detailed performance comparison with the current Cloud replica solution used by SeaDataNet and the NDN4DI will be performed. Moreover, the ENVRI-FAIR is a data infrastructure project which recently started, more than 10 different infrastructures are in the project. In this context, more use cases will be identified to improve the implementation of NDN4DI.

ACKNOWLEDGMENT

This research is partially supported by the European Unions Horizon 2020 research and innovation program under Grants No. 824068 (ENVRI-FAIR project), 825134 (ARTICONF project), 862409 (BlueCloud project) and LifeWatch ERIC.

REFERENCES

[1] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *arXiv preprint arXiv:1406.0440*, 2014.

[2] 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.

[3] S. Koulouzis, R. Mousa, A. Karakannas, C. de Laat, and Z. Zhao, "Information centric networking for sharing and accessing digital objects with persistent identifiers on data infrastructures," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2018, pp. 661–668.

[4] D. Dash, V. Kantere, and A. Ailamaki, "An Economic Model for Self-Tuned Cloud Caching," in *2009 IEEE 25th International Conference on Data Engineering*. Shanghai, China: IEEE, Mar. 2009, pp. 1687–1693. [Online]. Available: http://ieeexplore.ieee.org/document/4812593/

[5] D. E. O'Leary, "Embedding ai and crowdsourcing in the big data lake," *IEEE Intelligent Systems*, vol. 29, no. 05, pp. 70–73, sep 2014.

[6] B. Lee, H. Jeon, S. Yoon, and H. Song, "Towards a cdn over icn." in *DCNET/ICE-B/OPTICS*, 2012, pp. 46–51.

[7] S. Mastorakis, A. Afanasyev, Y. Yu, and L. Zhang, "ntorrent: Peer-to-peer file sharing in named data networking," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–10.

[8] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[9] Cloudflare, Inc. (2019) What is a cdn? [Online]. Available: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

[10] G. Berg-Cross, R. Ritz, and P. Wittenburg, "Data Foundation and Terminology Work Group Products," Aug. 2015. [Online]. Available: {https://doi.org/10.15497/06825049-8CA4-40BD-BCAF-DE9F0EA2FADF}

[11] A. Karakannas and Z. Zhao, "Information centric networking for delivering big data with persistent identifiers," *University of Amsterdam*, 2014.

[12] R. Mousa, "Application aware digital objects access and distribution using Named Data Networking," *University of Amsterdam*, 2017.

[13] H. Lim, A. Ni, D. Kim, Y.-B. Ko, S. Shannigrahi, and C. Papadopoulos, "Ndn construction for big science: Lessons learned from establishing a testbed," *IEEE Network*, vol. 32, no. 6, pp. 124–136, 2018.

[14] S. Shannigrahi, C. Papadopoulos, E. Yeh, H. Newman, A. J. Barczyk, R. Liu, A. Sim, A. Mughal, I. Monga, J.-R. Vlimant *et al.*, "Named data networking in climate research and hep applications," in *Journal of Physics: Conference Series*, vol. 664, no. 5. IOP Publishing, 2015, p. 052033.

[15] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking–a survey," *Computer Communications*, vol. 86, pp. 1–11, 2016.

[16] H. Yuan, T. Song, and P. Crowley, "Scalable ndn forwarding: Concepts, issues and principles," in *2012 21st International Conference on computer communications and networks (ICCCN)*. IEEE, 2012, pp. 1–9.

[17] M. Tortelli, L. A. Grieco, and G. Boggia, "Performance assessment of routing strategies in named data networking," in *IEEE ICNP*, 2013.

[18] J. D. McCabe, *Network analysis, architecture, and design*. Elsevier, 2010.

[19] S. Koulouzis, A. S. Belloum, M. T. Bubak, Z. Zhao, M. ivkovi, and C. T. de Laat, "SDN-aware federation of distributed data," *Future Generation Computer Systems*, vol. 56, pp. 64–76, Mar. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X1500312X

[20] J. Wang, A. Taal, P. Martin, Y. Hu, H. Zhou, J. Pang, C. de Laat, and Z. Zhao, "Planning virtual infrastructures for time critical applications with multiple deadline constraints," *Future Generation Computer Systems*, vol. 75, pp. 365–375, Oct. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X17301905

[21] J. Wang, C. de Laat, and Z. Zhao, "QoS-aware virtual SDN network planning," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. Lisbon, Portugal: IEEE, May 2017, pp. 644–647. [Online]. Available: http://ieeexplore.ieee.org/document/7987350/

[22] C. Fan, S. Shannigrahi, and C. O. et. al., "Managing Scientific Data with Named Data Networking," *NDM15*, 2015. [Online]. Available: https://named-data.net/wp-content/uploads/2015/11/Managing_Scientific_Data.pdf

[23] OASIS. OASIS. [Online]. Available: https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.2/os/TOSCA-Simple-Profile-YAML-v1.2-os.pdf

[24] S. Koulouzis, P. Martin, H. Zhou, Y. Hu, J. Wang, T. Carval, B. Grenier, J. Heikkinen, C. de Laat, and Z. Zhao, "Time-critical data management in clouds: Challenges and a dynamic real-time infrastructure planner (drip) solution," *Concurrency and Computation: Practice and Experience*, p. e5269, 2019.

[25] D. Oran. (2019) Maintaining CCNx or NDN flow balance with highly variable data object sizes. [Online]. Available: https://tools.ietf.org/id/draft-oran-icnrg-flowbalance-01.html

[26] L. Wang, M. Hoque, C. Yi, A. Alyyan, and B. Zhang, "OSPFN: An OSPF Based Routing Protocol for Named Data Networking," *Technical Report NDN-0003*, 2012. [Online]. Available: https://www.named-data.net/techreport/TR003-OSPFN.pdf

[27] V. Lehman, M. Hoque, Y. Yu, L. Wang, B. Zhang, and L. Zhang, "A Secure Link State Routing Protocol for NDN," *Technical Report NDN-0037*, 2016. [Online]. Available: https://named-data.net/wp-content/uploads/2016/01/ndn-0037-1-nlsr.pdf