Information Centric Networking for Delivering Big Data with Persistent Identifiers

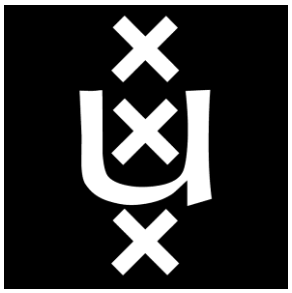
Research Project 2

Andreas Karakannas
andreaskarakannas@os3.nl

Supervised by:

Dr. Zhiming Zhao

University of Amsterdam



System and Network Engineering (MSc)

July 11, 2014

Abstract

Information Centric Networking (ICN) is a new and promising network concept which is founded upon the idea that most users in the internet are interested in accessing digital objects, irrespectively of their locations. Digital objects in ICN have unique names that are used in order to route data content from the source node to the destination node. During the content delivery from sources to destinations, the data contents are cached to intermediate nodes in order to achieve efficient and reliable distribution of the data content among the network infrastructure (in-network caching). In this research project we focus on the efficiency of ICN for delivering Big Data with Persistent Identifiers. We proposed a Mapping Architecture for resolving PIDs to ICN names and we evaluate the efficiency of in-network caching when delivering Big Data objects. Our results showed that in-network caching can offer significant performance benefits when the cache size of the network elements that perform in-network caching is bigger than the Big Data object size.

Contents

1 Introduction

Big Data [1] applications face challenges in acquiring, storing, sharing, transferring, analyzing and visualizing data with very large quantities and from distributed sources. Data research infrastructures (e.g. in the ENVRI project [2]) manage data from different sources and provide access services for scientists to perform inter-disciplinary researches. The persistent identification (PID) is an important mechanism for publishing Big Data objects, referencing data objects, and retrieving data contents.

Delivering data content from sources to destinations to perform further process and analysis is another important requirement in data research infrastructures, in particular when multi sources are involved for applications. The bandwidth limitation for Long Fat Networks (LFN) [3] is a performance bottleneck encountered during the distribution of Big Data in IP networks that led the researchers to use new state of the art technologies in order to overcome it. These new state of the art technologies include TCP tuning [4] Multipath TCP (MTCP) [5] data movement applications (GridFTP, bbFTP, FDT), applying light-paths for extending bandwidth and use Software Defined Networking (SDN) to dynamically control the network. All the aforementioned solutions are based on IP networks (end-to-end communication), and each solution has its shortcomings. Extending bandwidth technologies are limited from congestion problems in the network, thus they can only temporally solve the delivery of big data and dynamic control of the network using SDN is very complex and difficult to implement.

On the other hand, Information Centric Networking (ICN) [6] concept, a new network paradigm which is founded upon the idea that most users are interested in accessing data contents, irrespectively of their locations is a new and promising approach for networking. Instead of having an end-to-end communication model between nodes, data objects have unique names that are used in order to route data content from the source node to the destination node. During the content delivery from sources to destinations, the data contents are cached to intermediate nodes in order to achieve efficient and reliable distribution of the data content among the network infrastructure. It is thus clear, that ICN can offer a natural architecture for transferring Big Data.

However, using ICN to deliver PID based data objects in big data infrastructure is still in its very early stage, and there are also technical gaps between using PID for publishing and for retrieving Big Data content and using Information Centric Networking for delivering data. These gaps motivate us to investigate how to use ICN for delivering Big Data with Persistent Identifier (PID).

1.1 Scope

This research project focuses on how available ICN implementation can be used in Big Data infrastructure for delivering data with most common PID types. We highlight our research in the current PID standard schemes [7] and most well-known ICN implementations [8] respectively. Furthermore, we will also evaluate the

efficiency of current caching algorithms for delivering Big Data in ICN implementations.

1.2 Research Questions

The main research question of the project is to investigate:

Can Information Centric Networking (ICN) efficiently be used for delivering Big Data with Persistent Identifiers?

We will answer this question from the following three sub questions:

- What are the state of the art of ICN approaches and PID standards?
- How can PID standards be mapped to ICN's Object Identifiers?
- How are the current ICN caching mechanisms behaving when delivering Big Data contents?

1.3 Approach

We will conduct our research via three steps:

First, we will review the state of the art of current ICN approaches and PID standards. During the theoretical study in ICN approaches we pick the most popular ICN approach for further experimental studies.

Afterwards, the key elements that constitute each PID standard are identified and we will propose a Mapping Architecture for resolving PIDs to the ICN approach selected in the first step.

The last step of our approach will focus on evaluating the effectiveness of current caching algorithms for Big Data. For the evaluation we performed simulations under different scenarios using ccnSIM [9] caching simulator.

The rest of the report is organized as follows: In chapter 2 and 3, we will review the current ICN approaches and PID standards. In the chapter 4, we will discuss proposed Mapping Architecture. In chapter 5, we will present the experimental simulation scenarios and methodology, and discuss the results of our simulations. Finally, we will conclude the research and discuss the future directions.

2 Information Centric Networking

Information Centric Networking (ICN) is an umbrella term used to describe a number of research projects (DONA, NDN, PURSUIT, SAIL) which aim to evolve the Internet infrastructure and share the same objectives and structuring architectural properties. In this chapter, we first introduce the basic concepts of the ICN architecture and then we review the design choices that the most popular ICN approaches choose in order to implement these concepts.

2.1 Basic concepts of ICN

The basic concepts of the ICN architecture is the binding of each digital object with a unique name which is the object identifier(OI) of this digital object, the use of this OI in the network layer for routing and forwarding the digital object and the caching of the digital objects in the network infrastructure (in-network caching).

Binding each digital object with an object identifier (OI) plays a critical role in the ICN concept. While in the IP network, the digital objects are named based on the location from which they can be retrieved using Uniform Resource Locators (URLs) (e.g. *www.os3.nl/cia/dns_1.pdf*), in the ICN concept the digital objects are named independently of their location, thus an OI in ICN is location independent. The naming syntax of the OIs in the ICN approaches is the key factor for routing and forwarding the digital object as we will show later.

Routing and forwarding in ICN approaches is not based on the location of the server that hosts the digital object to be retrieved like in IP networks. In ICN routing and forwarding is based on the OI of the digital object requested. There are two approaches for routing and forwarding in ICN approaches that depends on the naming syntax of the OI. The first approach requires a service that is responsible for resolving OIs to a location from which the digital object can be retrieved. The second approach does not require a service for resolving OIs to a location. Instead, the OI itself is used in order for the request to be forwarded to a location from which the digital object can be retrieved.

In network caching in ICN means that the digital objects are dynamically cached by the network elements (e.g. Routers) during the routing and forwarding procedure of the digital objects. The In-network caching in ICN aims to efficiently distribute the digital objects among the network infrastructure.

To summarize table 1 shows the main differences between IP and ICN networking.

Building Blocks	**IP**	**ICN**
Digital Object Identifier	URL: Location Dependent	OI: Location Independent
Routing & Forwarding	Based on Location (IP) of the digital object	Based on the OI, Location Independent
In-Network Caching	NO	YES

Table 1: IP and ICN Networking differences

2.2 Data Oriented Network Architecture (DONA)

The Data Oriented Network Architecture (DONA) [10] is the first complete ICN approach introduced from UC Berkeley.

2.2.1 Naming Syntax of Object Identifier

In DONA naming syntax has a flat structure in the form P:L. The P part of the OI is globally unique and is defined as the principal which contains the cryptographic hash of the publisher's public key. On the other hand L is a locally unique name (in the scope of P) which uniquely identifies an object within the local scope. L syntax is left to the principals' (P) which may choose to just give a human readable name or even a cryptographic hash of the objects' content. The P:L naming syntax ensures that each OI in DONA is globally unique.

2.2.2 Name Resolution and Data Routing

Name Resolution in DONA is achieved through a Name Resolution System which is consisted by Resolution Handlers (RHs). Each Autonomous System (AS) in DONA has at least one RH and all RHs in the architecture are interconnected, thus creating a hierarchical Name Resolution System as shown on Figure 1.

Publishing of OIs into the network is done by sources that are allowed to register data into the Name Resolution Infrastructure (arrows 1-3). When a source register a name to its local RH, the RH stores a pointer mapping the OI with the source that register the OI in a table and also forwards this OI to its peering and parent ASs, causing each intermediate RH to store a mapping between the OI and the address of the RH that forwarded the registration. As all the registrations are forwarded up to Tier-1 ASs which are peers with each other, Tier-1 contains all the registered OIs. Authoritative sources can also register wild cards OIs (e.g. P1:*) to their local RH, indicating that they can provide services for all the requests for the principal P1.

Request for resolving an OI to its location is send by a subscriber in the form of a FIND(OI) message to its' local RH (arrows 4-7). The local RH look-up its mapping table and if a pointer mapping the requested OI is found, the FIND(OI) message follows the pointers route created by the registration procedure eventually reaching the publisher. If the pointer mapping the requested OI is not found at the local RH the FIND(OI) message is forwarded to the parent RHs until a pointer mapping the requested OI is found. Since Tier-1 knows all the registrations in the system, a pointer will eventually be found if the requested OI exists in the system. Data Routing back to the subscriber can be coupled or decoupled (arrows 8-11). In the coupled case, the FIND(OI) message saves the RHs that traversed during the resolution procedure and the answer containing the data is send to the subscriber using the reverse path. In the decoupled case, the answer containing the data is routed irrespectively of the routing path of the FIND(OI) message using regular IP routing and forwarding.

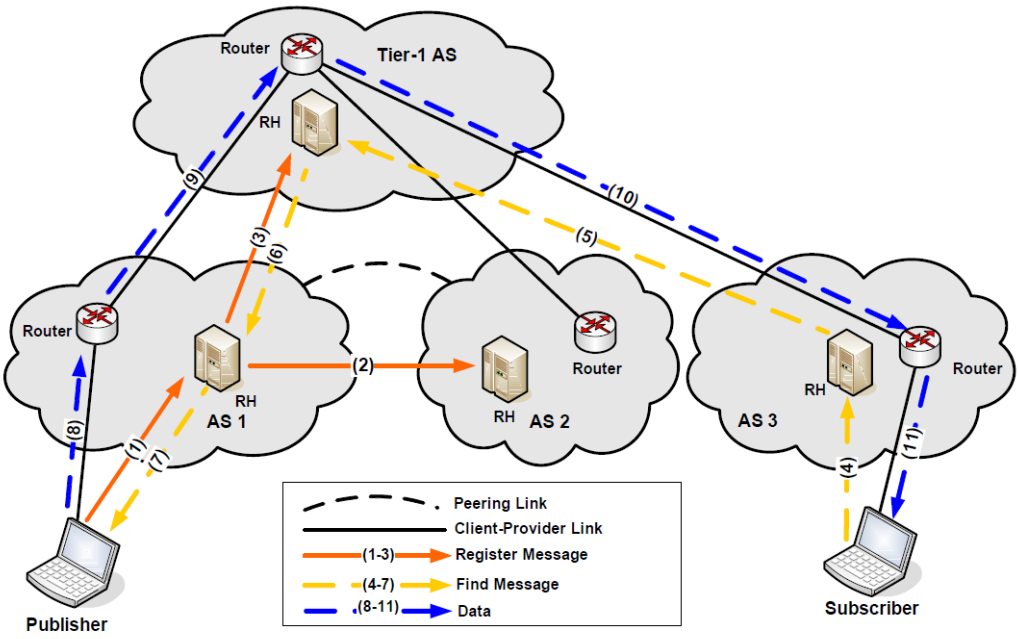


Fig. 1: The DONA Architecture [8]

2.2.3 In-Network Caching

In Network Caching in DONA is achieved using the RH infrastructure. In the coupled data routing case, the answer packet containing the OIs data will be routed to the subscriber thus the RHs can choose either to cache or not the OIs' content. In the decoupled data routing option, an RH can replace the IP address of the FIND(OI) message with its' own IP address, forcing the data routing to be done through it which it can then decide to cache or not the data. A subsequent FIND(OI) message can be then served from a cached copy in an RH.

2.3 Named Data Networking (NDN)

Named Data Networking (NDN) [11] formerly Content Centric Networking [12], is another pioneering internet architecture approach from PARC [13] which follows the ICN concept.

2.3.1 Naming Syntax of Object Identifier

In NDN the naming syntax [14] has a hierarchical structure which is based on the URI [15] naming syntax. The "/" delimiter is used to separate the hierarchy layers in an NDN OI, thus an OI can be */www.os3.nl/courses/cia/dns.pdf* where *www.os3.nl* indicates the higher layer in the hierarchy structure. OIs in NDN do not necessarily have to be human-readable at all hierarchy layers thus the */www.os3.nl/courses/cia/<dns.pdf content hash>* could also be an OI. The hierarchical structure of the naming syntax in NDN is very critical for the scalability of the architecture since it allows aggregation of OI prefixes in OI resolution and data routing. An NDN name can be an identifier of a file, an endpoint, a service in an endpoint or even a command to turn on some lights.

2.3.2 Name Resolution and Data Routing

The name resolution in NDN is achieved by using the hierarchical structure of the OIs requested Figure 2. The basic component of the NDN architecture is the Content Routers (CRs). Each CR maintains three tables, the Forwarding Information Base (FIB), the Pending Interest Table (PIT) and the Content Store table (CS). The FIB is the equivalent of the FIB in IP but instead of saving IP prefixes, it stores a mapping of OI prefixes and their corresponding output interface (next hop). The FIBs in NDN are populated using the OSPF-N [16] protocol which is the equivalent of OSPF in IP networks. The PIT stores a mapping between the requested OI and the interface from which the request was received. The CS is used as the caching store of the CR, and it contains a mapping between the OI and the OIs' data.

OI prefixes are published into the network by sources that are authorized to announce prefixes. When a publisher announces a prefix (e.g. */www.os3.nl/*) it has to be able to provide the data that corresponds to all OIs that starts with this specific prefix. The prefix that the publisher announced is populated by the CRs using the OSPF-N mentioned above. A publisher can be multihomed, and it can publish the same NDN name prefix in all its' providers. CRs that will receive OSPF-N prefix advertises from both providers of a multihomed publisher choose to store on their FIB table the output interface that corresponds to the provider that gives the best cost towards the publisher.

When a subscriber wants to retrieve the data that corresponds to an OI, it creates an INTEREST(OI) message and sends it to its' NDN default gateway(its' NDN CR) (arrows 1-3). When a CR receives an INTEREST(OI) message it first look-ups its' CS table to see if it has already the data regarding the OI requested and if so, it returns the data by issuing a DATA(OI,<data>) message and sending it to the interface through which it received the INTEREST(OI) message. If the CR did not find the requested OI in its' CS it performs a longest prefix match to its' FIB, forwards the INTEREST(OI) packet to the corresponding output interface(next hop) and saves the mapping of the OI requested and the interface from which it received the request to its' PIT. When a CR receives a DATA(OI,<data>) message it makes a look-up on its' PIT and forwards the DATA(OI,<data>) message to the interfaces from which this OI was requested (arrows 4-8). It is thus clear from the description above that the data routing back to the requester is coupled and that the DATA(OI,<data>) is routed through the reverse path of the INTEREST(OI) message.

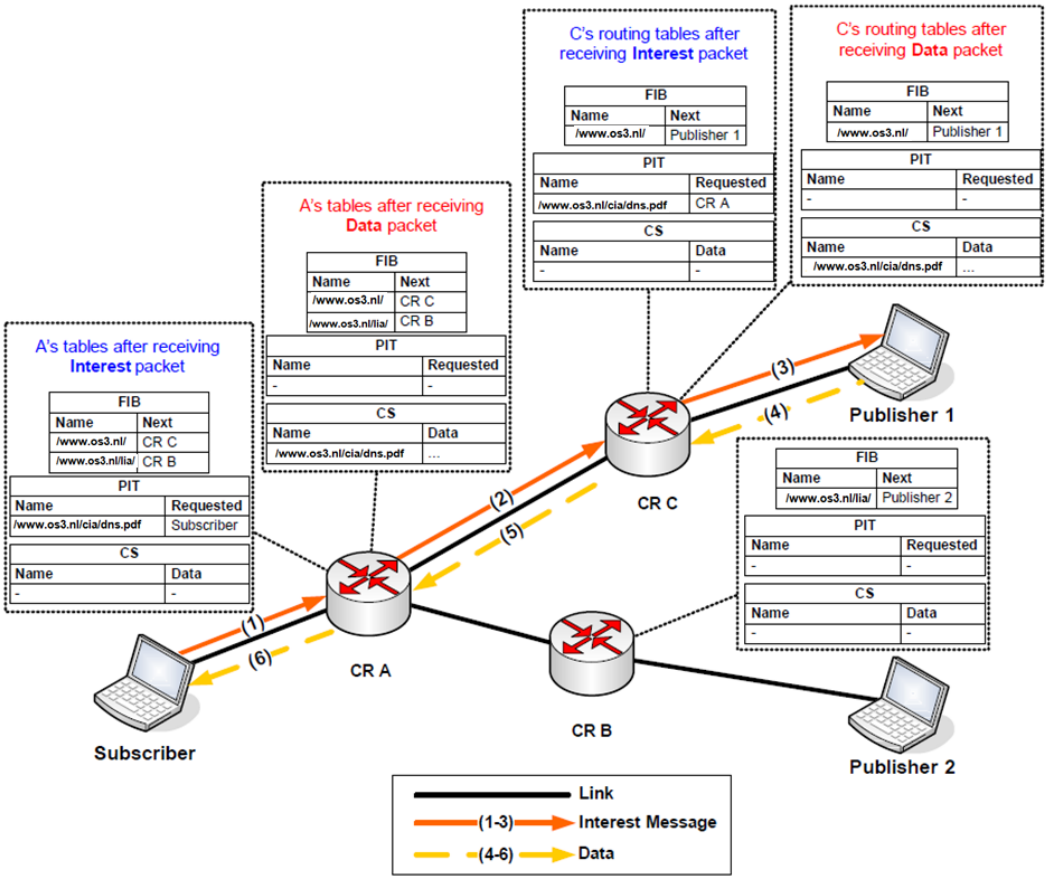


Fig. 2: The NDN Architecture [8]

2.3.3 In-Network Caching

In-Network Caching in NDN is achieved at the CR as mentioned above. When a CR receives a DATA (OI,<data>) message it caches it on its' CS. In the NDN the caching decision algorithm currently used is Leave Copy Everywhere (LCE) in which each CR caches every DATA (OI,<data>) it sees.

2.4 Publish Subscribe Internet Technology (PURSUIT)

Publish Subscribe Internet Technology (PURSUIT) [17] formerly Publish Subscribe Internet Routing Paradigm (PSIRP) [18] is one more of the ICN approaches founded by the EU Framework 7 Programme.

2.4.1 Naming Syntax of Object Identifier

In PURSUIT, the naming syntax has a flat structure in the form <Scope_ID><Rendezvous_ID>. The scope ID groups related information objects and can have a hierarchical structure (e.g. /Family_Photos/2013/, /Family_Photos/2014/) with the "/" delimiter used to separate the hierarchy layers. On the other hand Rendezvous ID is the identifier of a particular object within the scope ID. Both scope IDs and rendezvous IDs can be a cryptographic hash, thus */Family_Photos/2013/<wedding.png content hash>* can be an OI.

2.4.2 Name Resolution and Data Routing

Name resolution in PURSUIT is achieved through the rendezvous function. The rendezvous function is implemented by a collection of Rendezvous Nodes (RNs), called the Rendezvous Network (RENE). The RENE is implemented as a hierarchical distributed hash table (DHT) [19] as shown on figure 3. Each RN is responsible for managing and resolving a number of Scope IDs.

Published of OIs into the network is done by sources that are authorized to publish data into the RENE infrastructure (arrows 1-2). A source can publish an OI by sending a PUBLISH(OI) message to its' local RN, which is in turn routed by the DHT to the RN responsible for the Scope ID specified in the OI.

When a subscriber wants to retrieve the content of an OI it sends a SUBSCRIBE(OI) message to its' local RN which in turn is forwarded by the DHT to the RN responsible for the specific OI (arrows 3-6). When the responsible RN receives the SUBSCRIBE(OI) message it contacts the Topology Manager (TM) (arrows 7-8) which is responsible in creating the route for delivering the OI data from the publisher to the subscriber. . The TM implements the topology management function through which it discovers the network topology and the forwarding function which uses the network topology to create routes from source to destination when requested by an RN. After the TM has created the route from the publisher to the subscriber it provides this route to the publisher (arrows 9-10) which is in turn add this route along with the OI and OIs' content to a message and sends it to a Forwarding Node (FN) (arrows 11-14). Each FN uses the route from the message it receives to identify in which output interface (next FN) the message will be forwarded. The FNs does not need to keep any state of the routing path, thus the OIs' data routing in PURSUIT is decoupled.

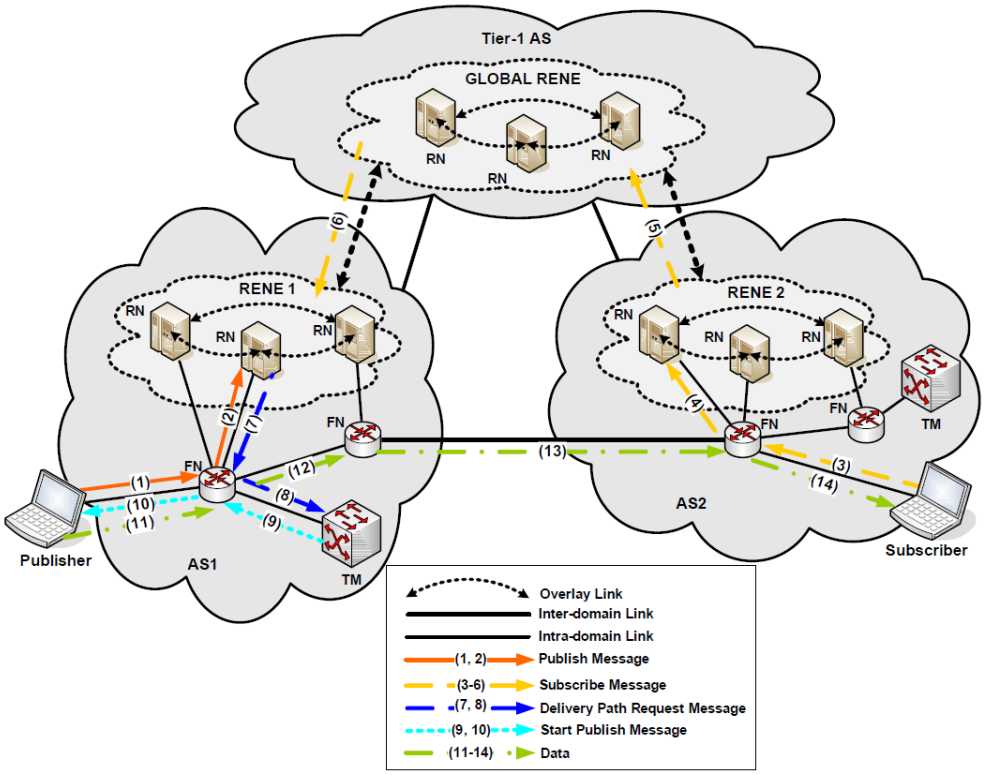


Fig. 3: The PURSUIT Architecture [8]

2.4.3 In-Network Caching

In-network caching in PURSUIT is performed in the FNs during the OIs' data routing from the publisher to the subscriber. However, in-network caching may not be so effective due to the fact that the SUBSCRIBE(OI) messages can follow entirely different route from the OIs' data messages.

2.5 Scalable and Adaptive Internet Solutions (SAIL)

Scalable and Adaptive Internet Solutions (SAIL) [20] formerly Architecture and Design for the Future Internet(4WARD) [21] is another ICN approach funded by the EU Framework 7 Programme.

2.5.1 Naming Syntax of Object Identifier

Naming Syntax in SAIL has flat structure in the form ni://A/L. The A part of the OI identifies the authority and is globally unique and the L part identifies the object and is locally unique within the authority, thus ensuring global uniqueness of the OI. Each part can be a cryptographic hash or a regular string (e.g. *ni://Authrity_1/<Hash of DNS.pdf data>, ni://<Hash of Authority_1 public key>/DNS.pdf, ni://< Hash of Authority_1 public key>/<Hash of DNS.pdf>*). OIs can also have a hierarchical structure as in NDN.

2.5.2 Name Resolution and Data Routing

Name resolution in SAIL can be achieved by either using the Name Resolution System (NRS) or by using the hierarchical structure of the OIs through Content Routers (CRs) like in NDN. The NRS is based on a multilevel DHT and is consisted by the Global NRS and multiple Local NRSs as shown on figure 4. Each authority (A) in SAIL maintains a local NRS in which it registers the L part of an OI along with its' corresponding locator (i.e. IP address) from which the OIs' data can be retrieved. The Global NRS has a mapping between the authorities and their corresponding local NRS. Each local NRS is responsible for aggregating the mapping between the L parts and their locators and registering this aggregation in the Global NRS, thus the Global NRS knows the location of every OI in the network.

Published of OIs into the network is done by sources that are authorized to publish data in the NRS and the CRs. When a publisher wants to publish an OI it sends a PUBLISH(L) to its local NRS which stores the L and the sources' locator to its' mapping database (arrows 1-2). A publisher can also announce a name prefix to the CR as in NDN.

When a subscriber wants to retrieve the data of an OI it can either choose to send a GET(ni://A/L) message to its' local NRS(arrows 3) or send the GET(ni://A/L) message to its' default CR gateway (arrow a). In the first case, its' local NRS will contact the Global NRS which will return the locator of the OI to the subscriber (arrows 3-6) which in turn can create a GET(locator) message and use regular routing to retrieve the OIs' data from the publisher(arrows 7-12). In the second case, the GET(ni://A/L/) will be forwarded to the publisher like in NDN and the publisher will

respond with the OIs' data (arrows a-f). The main difference with the NDN routing of the data back to the subscriber is that instead of each CR having a PIT, the route is accumulated to the GET(ni://A/L) message and its' then added to the data message by the publisher, thus used later by the CRs to route the data message from the reverse path.

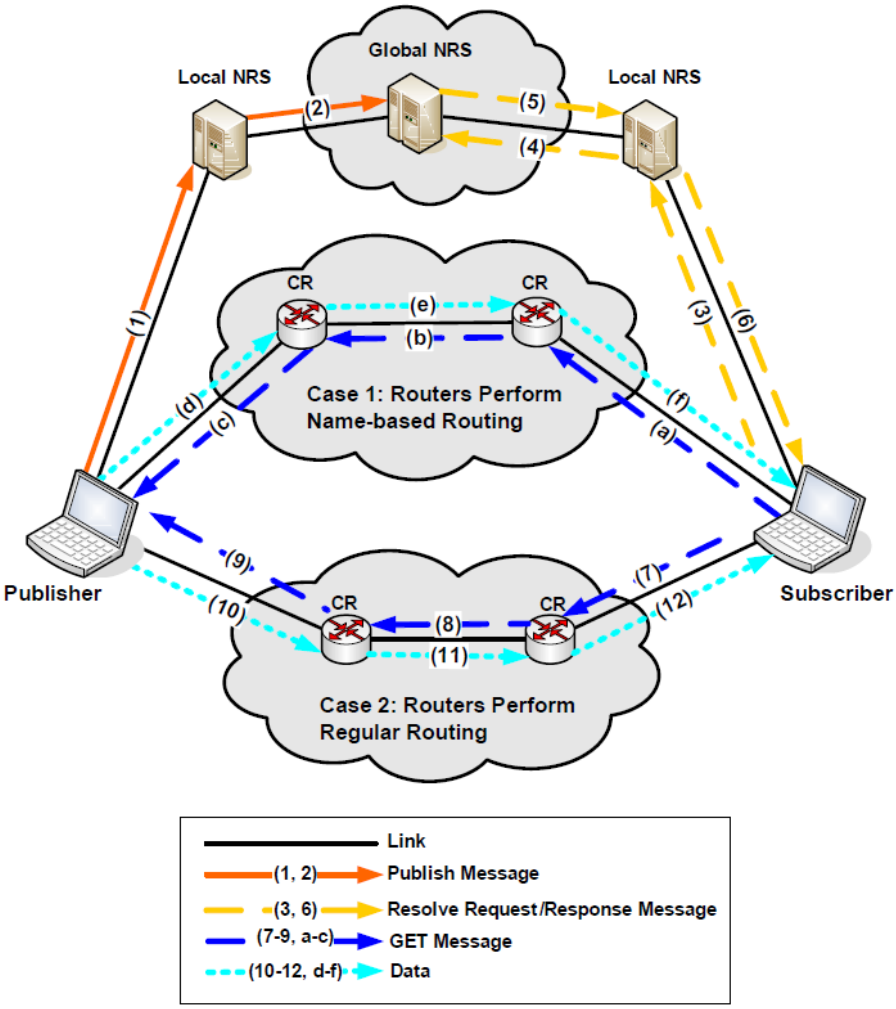


Fig. 4: The SAIL Architecture [8]

2.5.3 In-Network Caching

In network caching in SAIL is performed at the CRs like in NDN. Moreover, SAIL has replication mechanism that performs caching of data objects inside the NRS. Therefore, a GET(ni://A/L) request that is routed through the Global NRS can hit a cache copy that will be retrieve to the subscriber. However, caching in the NRS is done be establishing agreements between the different NRSs and is not embedded in the network infrastructure.

2.6 Summary of ICN approaches

Table 2 shows a summary of the ICN approaches characteristics reviewed in this research project. DONA and SAIL are both promising approaches in ICN, however there are no known software implementations available for evaluating the efficiency of those architectures. PURSUIT on the other hand offers a complete software implementation of the functionalities of each component in the architecture, yet its' inefficiency in-network caching strategy is considered inappropriate for using it for Big Data objects. On the other hand, based on our research we find NDN the most mature ICN approach. NDN offers a complete software implementation of the architecture and also a lot of open source NDN simulators [22] are available for evaluating the efficiency of in-network caching strategies under different scenarios. Furthermore, most the research papers [23] [24] [25] for evaluating the efficiency of in-network caching algorithms in ICN architectures uses the NDN approach as the reference ICN architecture. Finally, NDN is the only approach that has published a specification [14]. For the aforementioned reasons we decided to focus the remaining of our research in the NDN approach.

	DONA	**NDN**	**PURSUIT**	**SAIL**
Namespace	Flat with structure (P:L).	Hierarchical using URI based naming schemes ("/" separate hierarchy layers).	Flat with structure (<Scope><Rendezvous>) Scope part can be hierarchical.	Flat with structure (ni://A/L/). Both A and L can be hierarchical
Routing the OI Request message	OIs are used by the RHs to route the Request message through the Name Resolution Systems towards the source.	Request message is routed by the Content Routers (CRs) using longest prefix max between the OIs and the FIBs entries.	Rendezvous Network (RENE) is used to route an OI Request message to its' Rendezvous Node (RN) which has knowledge about the source location.	Case 1: Routed through the Name Resolution System(NRS) until a pointer to the OI and its' locator is found and returned to the requester. Regular IP routing is followed onwards. Case2: Routed by the Content Routers (CRs) like in NDN
Routing the OIs' Data message	Coupled: The routing of the Data message follows the reverse route of the Request message. Decoupled: The source uses regular IP routing to route the OIs' Data message.	Coupled: Content Routers (CRs) keeps the state of each Request message and use to return the Data message through the reverse route of the Request message	Decoupled: Source and Destination of an OIs' Data message is send from the RN to the Topology Manager (TM) which creates the route from source to destination.	Decoupled: Used when Case 1 is used for the routing the OI Request message. The Data message is returned by performing regular routing. Coupled. Used in Case 2 and follows the reverse path as in NDN.
In-Network Caching	Performed at Resolution Handlers (RHs).	Performed at the Content Routers (CRs).	Performed to the Routers among the Data delivery path. Not so efficient since Request message can follow a completely different route.	Performed at Content Routers(CRs). Also possible at NRS using agreements between NRSs.

Table 2: Summary of ICN approaches characteristics reviewed at this Research Project

3 Persistent Identifiers

At the beginning of the World Wide Web, its' creator Tim Bernes-Lee proposed in the Internet Engineering Task Force (IETF) the Uniform Resource Identifier (URI) as the naming scheme for describing identifiers for the Web contents. The IETF rejected the use of URIs as the identifier of Web contents due to the fact that they wanted to allow the WWW users to change the URIs of Web contents when they moved to another location. Therefore, Uniform Resource Locators (URLs) where chosen as the Web content identifiers. Although, this way of retrieving digital objects was found to be working fine in the early stages of the World Wide Web(WWW), several studies [26] [27] have found out that approximately 50% of the URLs in scholarly publications fail to retrieve the digital object after a period of seven to ten years. This problem is well known as 'link rot' and it can be caused by several reasons some of which are shown below:

- The digital object has moved to another server and is no longer accessible through the given URL (The 404 error 'Page not Found' does not gives any insight on what happened to the digital object).
- The URL is accessible but it now points to another digital object (The user may not be informed of that change).
- The website registration has expired.
- The web server hosting the digital object has been upgraded to a new machine with a new web address.
- The website directory structured is rearranged.

The 'link rot' problem has led to the need for persistent identification (PID) of the digital objects. A persistent identifier is a long lasting, globally unique identifier which can be resolved to a representation of the digital object that gives information about how to access it, meta-data about the digital object and even more. The PID resolved representation can be updated when the digital object changes location, or is no longer available so that it can continuously give appropriate resolve representation of the same digital object. From our research in the current PID standards, we found that all the PID standards have the same hierarchy in naming their digital objects. Specifically, each PID standard is consisted by three parts as shown in Figure 5 which makes the PID globally unique.

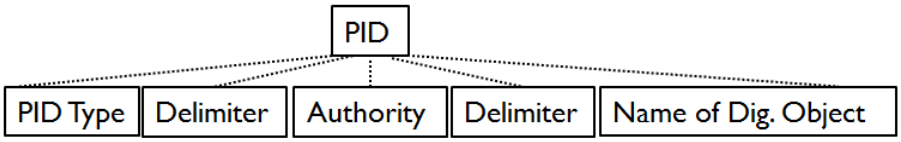


Fig. 5: PID hierarchical scheme

In this chapter we will review the current PID standards and services that were developed in order to overcome the "link rot" problem of the URL.

3.1 Uniform Resource Name (URN)

The Uniform Resource Name (URN) concept is a PID standard which was first introduced in RFC 1737. The URN syntax (RFC 2141) is based on the Uniform Resource Locator (URI) syntax and is shown below:

"urn:" <NID> ":" <NSS>

The first part specifies that the specific PID is a URN, thus it uses the naming syntax of the URN standard.

The second part <NID> is the Namespace Identifier which identifies the namespace or in other words the authority that publishes the specific URN.

The last part <NSS> is the Namespace Specific String which syntax depends on the authority identified by the NID. The <NSS> can be used from the authority for further delegation to sub-authorities.

URNs were developed to be independent of any one specific resolution service, thus although some resolution mechanisms were proposed (NAPTR[RFC 2168] ,Trivial HTTP [RFC 2169]) there is no universal resolution system for URNs. As a result different authorities that are using the URN to name their digital objects has established online services [28] for resolving a URN to a URL which it can be then used to retrieve the digital object.

3.2 The Handle System

The Handle System [29] is an infrastructure designed to provide naming services. The Handle System is composed by an open set of protocols, a name space and a reference software implementation. It uses handles to uniquely name a digital object. Its main functionalities are specified on RFC 3650. The naming scheme of the handle system is shown below:

"handle:"<Handle Naming Authority>"/"<Handle Local Name>

The Handle Naming Authority is a prefix that it is assigned by the Global Handle Service. It has hierarchical structure similar to the DNS domain names. Specifically, the Handle Naming Authority is sequence of decimals that are separated by the dot('.') character (e.g. 1.2.3). The path is read from the left to the right and the dot('.') character is used to define the hierarchy of the Naming Authorities. The hierarchy does not imply any technical implication. That is, a Handle Naming Authority '1.2.3' can be independent of the HNA '1.2'. The protocol specification ensures that in order to create a new HNA (e.g. '1.2.3'), an authorization from the higher HNA (e.g. '1.2') in the hierarchy is required but no further dependencies are implied.

The slash '/' is used to separate the Handle Naming Authority from the Handle Local Name

The Handle Local Name syntax is specified by its Handle Naming Authority policies. The only limitation is that it can only contain printable characters from Unicode's UCS-2 character set.

The resolution mechanism of the Handle System is based on the Handle System itself. The Global Handle System hosted by the Corporation for National Research Initiatives (CNRI) is the root server of the Handle System and has knowledge about all the Handle Naming Authorities. Each Handle Naming Authority can establishes its own resolution infrastructure, thus queries for handle resolution can be delegated by the Global Handle System to the corresponding Handle Naming Authority. Each Handle Naming Authority has a web accessed handle resolver (e.g. [30]) that can resolve a handle to a URL which it can be then used to retrieve the digital object.

3.3 Digital Object Identifier (DOI)

Digital Object Identifier (DOI) [31] is another PID service that is managed and controlled by the DOI Foundation (IDF). DOI uses the Handle System as the underlying communication technology for managing and resolving DOIs. It has been assigned the <Handle Naming Authority> value 10 in the Global Handle System. The DOI is mostly an administrative framework for assuring common practices and standards for publishing and maintaining handles between the Registration Agencies (RAs). An RA is an organization or institution that must fulfill specific quality standards in order to participate in the DOI project. The RAs are responsible for assigning DOIs to digital objects. The naming syntax of a DOI is as shown below.

"doi:10."<unique number"/"<name>

The "doi:" identifies the PID as a DOI and the <unique number> is an identifier for the RA which is locally unique. The <name> identifies the name of the digital object and is unique among the RAs' scope. The resolution of a DOI to the digital object is achieved by web accessed resolver. The DOI Resolver [32] is the higher in the hierarchy DOI resolver and it can resolve any RAs' DOI to a URL from which the digital object can be retrieved. A list of the current RAs can be found here [33].

3.4 Archive Resource Key (ARK)

The Archive Resource Key (ARK) [34] is another concept for persistent identification of digital objects. ARKs naming syntax is shown below:

"ark:/"<NAAN>"/"<Name>[<Qualifier>]

The Name Assigning Authority Number (<NAAN>) identifies the Naming Assigning Authority (NAA) that assigned the specific ARK. The NAAN is a string of five to nine decimals that uniquely identify each NAAN.

The <Name> is an identifier of the digital object which is unique among the NAAN scope. The <Name> is a string composed of printable ASCII characters and should be less than 128 bytes in length.

The Qualifier is an optional parameter of the ARK that specifies a variant of a digital object. It is added to the ARK identifier by appending a "." after the <Name>.

The resolution of an ARK is achieved by web accessed resolver called Naming Mapping Authority (NMAH). Each NAA has its' own NMAH that resolves the ARK to a URL which it can be then used for retrieving the digital object.

3.5 Persistent URL (PURL)

Persistent URL (PURL) [35] is another PID service that was developed from the Online Computer Library Center (OCLC) [36] The PURL naming syntax is shown below:

"purl:"<protocol><resolver address>"/"<name>

The <protocol> specifies the protocol used to access to resolver of the PURL and the <resolver address> the location of the resolver from which the digital object specified in the <name> field can be resolved.

A PURL is basically a URL that instead of pointing directly to the location from which the digital object can be retrieved, it points to a resolver that does the mapping between the <name> and the actual URL from which the digital object can be retrieved.

3.6 Summary of PID standards

From our research in the current PID standards, we found that all the PID standards have the same hierarchy scheme in naming their digital objects. Moreover, the resolution of each PID type to a locator is achieved by a web accessed resolver that resolves the PID to a URL through which the digital object can be retrieved. Table 3 shows the values that the PID standards reviewed in this research project have in their hierarchical naming scheme shown above in figure 5.

PID Types	PID Type Identifier	Delimiter	Authority	Delimiter	Name
URN	urn	:	<NID>	:	<NSS>
HANDLE	handle	:	<Handle Naming Authority>	/	<Handle Local Name>
DOI	doi	:	10.<Naming Authority>	/	<doi name syntax>
ARK	ark	:	/<NAAN>	/	<Name>[<Qualifier>]
PURL	Purl	:	<protocol><resolver address>	/	<name>

Table 3: Hierarchical scheme of PID standards review at this Research Project Values

The hierarchy scheme followed by the current PID standards was the key in creating the name-space implementation of our Mapping Architecture as we will show later.

4 Delivering PID objects via NDN

In this chapter we present our proposed architecture for mapping PIDs to NDN OIs. First, we define our Mapping Architecture Design Goals. Then we continue with the namespace-implementation of our architecture. The functionality of each component of our architecture design is then explained and the PID resolution procedure is specified. Finally, we show how a PID is resolved to an NDN name (OI) using our Mapping Architecture Design.

4.1 Schema mapping between PID and OI

Our main goal during the design of the Mapping Architecture was to provide a solution that will be:

- Generic: Can support many different PID types.
- Extensible: New PID Types could be easily added later on.
- Scalable: Can provide PID resolution for a large number of requests per time unit (size scalability) and across large geographical distances (geographical scalability).
- Availability & Performance.

In general there are two approaches for mapping a PID to an NDN Name.

The first approach could be based on rules that can be applied on a PID and transforms it to an NDN Name. For example if a PID is *urn:isbn:0-7645-2641-3* and the NDN name is */urn/isbn/0/7645/2641/3* a simple rule that will replace all the ":" and "-" to "/" will have successfully transform the PID to an NDN name. The rule could be installed to each client ndn-enable browser which given the PID would have applied the appropriate rule, forward the NDN name to the NDN network and eventually receive the digital object that corresponds to that PID. Although this approach of mapping PID to an NDN Name sounds simple it has a lot of disadvantages. Firstly, the mapping will be highly depended on the clients' ndn browser which will need to be updated every time new rule would be appeared or changed. Moreover, if for some reason the NDN Name that corresponds to a PID changes, the digital object could not be retrieved although the PID would be valid. Furthermore, since the name of the digital object field in the PID could be used for further delegation and each authority can choose to use any delimiter it wants to divide the hierarchy of its' sub-authorities, a vast number of rules could be needed in order for an ndn browser to be able to perform mapping for all the authorities that exists. Finally, each PID authority that wants to publish a PID for an NDN name it will also need to develop a rule for resolving the PID to the NDN name. Finally, due to the fact that some PIDs formats may not be easily transform to an NDN name by applying a rule, a publisher will not have the opportunity to choose any PID type it likes,

The second approach is a PID resolution service that will keep bindings between the PIDs and their corresponding NDN names. By using this approach, the clients ndn browser would not have to be updated with the rules of each authority, thus all the aforementioned disadvantages would not be present. Moreover, the authorities would

not have any limitations on which PID type to choose for publishing their NDN digital objects. This approach is more generic and extensible and for these reasons we decided to follow this approach for mapping PIDs to NDN Names.

4.2 NDN gateway for PID

The Mapping Architecture design goals specified above lead us to create a name-space implementation in which the PID resolution process along with the name-space management will be hierarchically distributed across multiple machines. The hierarchy layers on our Mapping Architecture name-space implementation was based in the hierarchical naming scheme of the PID standards reviewed in the previous chapter. Figure 6 shows the name-space implementation of our Mapping Architecture. As shown in figure 6, the name-space is hierarchically distributed in a tree structure. The tree is consisted of three layers, the Root PID layer, the PID Type layer and the Authority PID layer. Each layers' components have a unique NDN name.

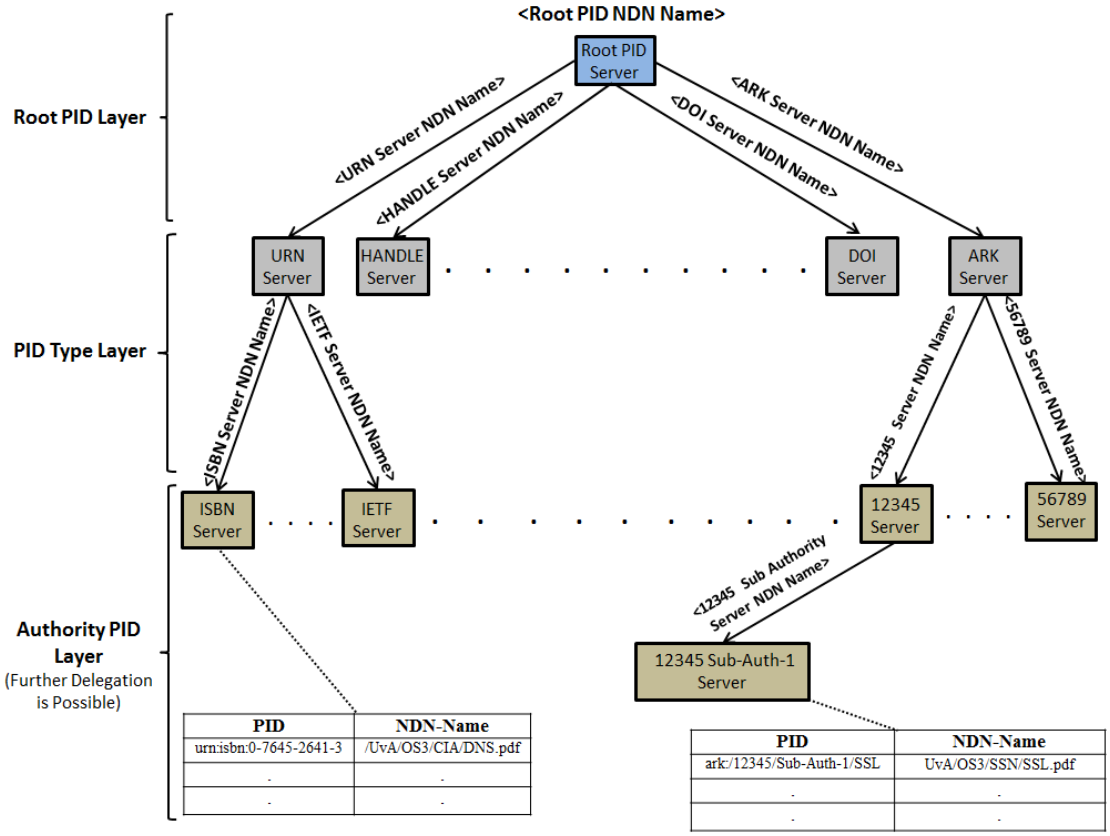


Fig. 6: Mapping Architecture Namespace Implementation

The Root PID layer is the top of the tree and the main component is the Root PID Server. A Root PID Server maintains a mapping table between PID types and their corresponding NDN name. Its' main functionality is to resolve the PID Type Identifier field of a PID to the NDN name of the server responsible for this PID Type.

The PID Type layer is the second layer of the tree structure and it consists of the server responsible for each PID Type. Each PID Type server maintains a mapping table between the authorities that uses the specific PID type to publish PIDs and their corresponding NDN name. The functionality of a PID Type server is to resolve the

Authority identifier field of a PID to the NDN name of the server responsible for this Authority.

The Authority PID layer is the last layer on our tree structure and it consists of the server responsible for each authority. Further delegation to sub-authorities is allowed on this layer. In this case, the parent authority server maintains a mapping table between their sub-authorities and their corresponding NDN name. The components that are leafs (authorities or sub-authorities) in the tree, maintain the mapping table between the PIDs and their corresponding NDN names.

Our name-space implementation ensures that each PID can be resolved to its' corresponding NDN name by following the path from the Root PID server towards the Authority (leaf) server which maintains the mapping between PIDs and their corresponding NDN names. Moreover, our name-space implementation meets our Mapping Architecture design goals defined in the previous section by the following:

The name-space implementation is generic since it can support all the current PID types reviewed in chapter 3. The naming scheme of all the reviewed PID types and specifically the delimiter used to separate the PID Type field with the Authority field can be used by the Root PID server to extract the PID Type from a PID and do the look up to its' mapping table.

Extensibility is achieved by allowing new PID types to easily be added to the name-space implementation. The only requirement for a PID type that wants to register to the mapping infrastructure is to have a PID schema from which the PID type can be easily extracted from the Root PID server. PID types that follow this requirement can be easily added to the Mapping Architecture.

From the availability and performance point of view, servers in each layer have to meet different requirements. Servers higher in the hierarchy are expected to have higher availability since if a server fails, a large part of the name space will be unreachable. Moreover, higher level servers' performance is also important since more request per time unit will be send to these servers. Since changes in the Root PID layer and the PID Type layer are expected to rarely occur (i.e. New PID Types and new Authorities are not every day registered in the Mapping System), replication of the servers at these layers can easily be deployed and maintained. Replicas should be geographically and efficiently distributed all around the world in order to overcome geographical scalability delays. Furthermore, since answer from look up operations at these layers remains valid for a long time, caching these answers to the client side (e.g. clients' resolver) can be effective and will significantly reduce the burden from the servers at these layers. On the other hand, servers at the Authority PID layer have completely different requirements from the availability and performance point of view. PIDs are expected to register to each authority in a daily basis, thus replication of the servers will introduce difficulties in keeping all the replicas consisted and should be avoided. Client resolver caching side is also considered inappropriate since it will burden the resolver with a vast amount of information. These lead us to the solution of high performance machines for running the servers at the Authority PID layer. Table 4 shows an overview of the availability and performance requirements for each layer.

Item	Root PID	PID Type	Authority PID
Geographical Scalability	Worldwide	Worldwide	Organizational
Update Propagation	Small	Small-Medium	High
Number of Replicas	Many	Many	None
Clients' Resolver Caching	Yes	Yes	No
Servers	High Performance is Recommended	High Performance is Recommended	High Performance is Required

Table 4: Availability and Performance Requirements for each Layer

4.3 Interoperable PID/OI handling

The hierarchical distribution of the Mapping architecture name-space across multiple servers defined above, affects the implementation of the PID resolution. In order to explain the resolution of a PID in a large-scale environment (Worldwide), we assume for now that there is no cache information in the clients PID resolver. A client PID resolver is the equivalent of a name resolver in the DNS system thus is responsible for resolving the PID to an NDN name. Moreover, the NDN name of the Root PID server is required to be known by the Clients' PID resolver and the Client is required to know the NDN name of its' PID resolver. Referring to figure 6, assume that the PID to be resolved is the ark:/12345/Sub-Ath-1/SSL. In general there are two ways to implement the PID resolution, iteratively and recursively.

In the iterative PID resolution the server that has received a PID is responsible to resolve this PID to its' corresponding NDN name and return it to the requester.

In the recursive PID resolution the server that has received a PID is responsible to resolve the PID as far as it can on the tree and return information about how to continue the resolve procedure to the requester.

In our implementation of the PID resolution we use both iterative and recursive PID resolution as shown on Figure 7.

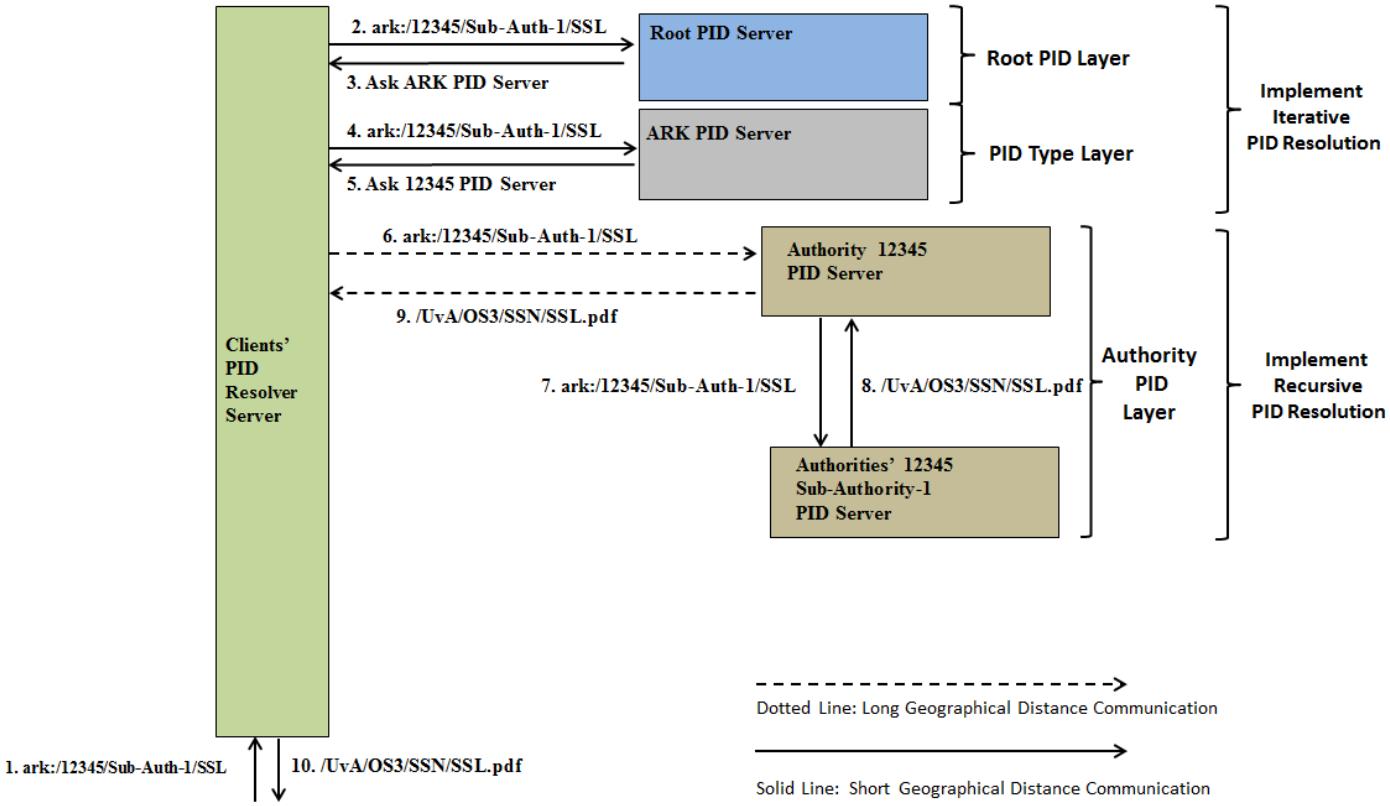


Fig. 7: PID Resolution Implementation

Specifically, the servers in the Root PID layer and the PID Type layer perform iterative PID resolution and the servers in the Authority PID layer implement recursive PID resolution. Our choice was based in the fact that the servers in the Root PID layer and the PID Type layer can have replicas efficiently geographically distributed, thus the communication with the Clients' PID Resolver will not introduce high time delays. On the other hand the servers on the Authority PID layer implement recursive PID resolution that is the parent Authority PID server from which the Clients' PID resolver will request the resolution of a PID will return the NDN name of that PID. The choice was based in the fact that as mentioned on the previous section of this chapter, Authority PID servers cannot easily manage geographically distributed consisted replicas and also to the expectation that parent Authority servers will be in a close geographical distance with their sub-Authorities servers (e.g. same country) thus the recursive PID resolution perform in this layer will not introduce high time delays. To finalize, the use of iterative PID resolution in the Root PID layer and PID Type layer and the recursive PID resolution in the Authority layer meets our design goal about geographical scalability of the Mapping Architecture.

4.4 Prototype

In the previous sections, we describe our name-space implementation, the implementation of the PID resolution and we analyze how our initial design goals can be meet using the proposed Mapping Architecture. In this section we describe how the Mapping Architecture can to be implemented in an NDN network infrastructure.

As mentioned in the review of the NDN architecture at chapter 2 the packet types defined in the specification are the INTEREST packet and the DATA packet. The forwarding of the INTEREST packet is achieved by the Content Routers (CRs) that perform longest prefix match between the NDN Name in the INTEREST packet and their Forwarding Information Base (FIB). In order to be able to forward INTEREST packet to the components of our Mapping Architecture, the Name inside the INTEREST packet has to contain the NDN name of the component to which the INTEREST message has to be delivered. Furthermore, the PID must be also forwarded to the component in order for the Mapping Architecture to work. Based on this requirements the NDN name of the INTEREST packets for the communication between the components of our Mapping Architecture must have the scheme shown in Figure 8.

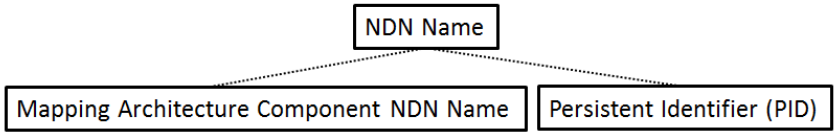


Fig. 8: NDN Name in the INTEREST packet

Since the NDN Name field of the INTEREST packet starts with the Components NDN Name it ensures that the INTEREST(<Components' NDN Name><PID>) packet will be forwarded to the appropriate component. Moreover, bearing in mind the each component knows its NDN name, when it receives an INTEREST(<Components' NDN Name><PID>) it can easily extract the <PID> field from the NDN Name. On the other hand, a DATA packet in NDN must contain the NDN Name of the INTEREST packet due to the coupled data routing used in NDN and a data field containing the data that corresponds to the INTEREST request, thus a DATA packet exchanged between the components of our Mapping Architecture will have the scheme shown in Figure 9.

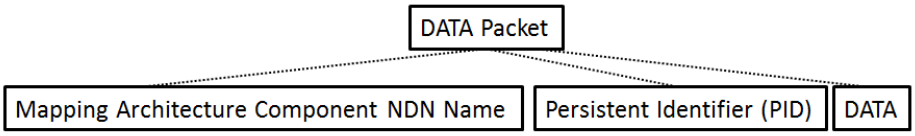


Fig. 9: DATA packet scheme for Mapping Architecture communication

The DATA field inside the DATA packet must contain information about what would be the next step in the PID resolution procedure. The possible values that the DATA field of a DATA packet can have are shown on table 5.

Value	Meaning
FOUND:<NDN Name of PID>	The NDN Name of the PID was found. <NDN Name of PID> contains the NDN Name of the PID
NOT_FOUND:<Description Message>	The PID could not be resolved. <Description Message> contains information about the failure
REDIRECTION:<NDN Name of Component>	A redirection to another Component of the Mapping Architecture <NDN Name of Component> is the NDN Name of the Component that needs to be queried

Table 5: Possible Values of DATA field of the DATA packet

Referring again to figure 6, assume that the PID to be resolved is the *ark:/12345/Sub-Ath-1/SSL* and that no caches exists in the Clients' PID Resolver. The client types *ark:/12345/Sub-Auth-1/SSL* to its' NDN enable browser. Its' browser under the hood, constructs the INTEREST(<Clients' PID_Resolver_NDN name>ark:/12345/Sub-Auth-1/SSL) and sends it to its' default NDN gateway. This INTEREST packet will be forwarded through the CRs to the Clients' PID resolver. The PID resolution procedure is shown on figure 10.

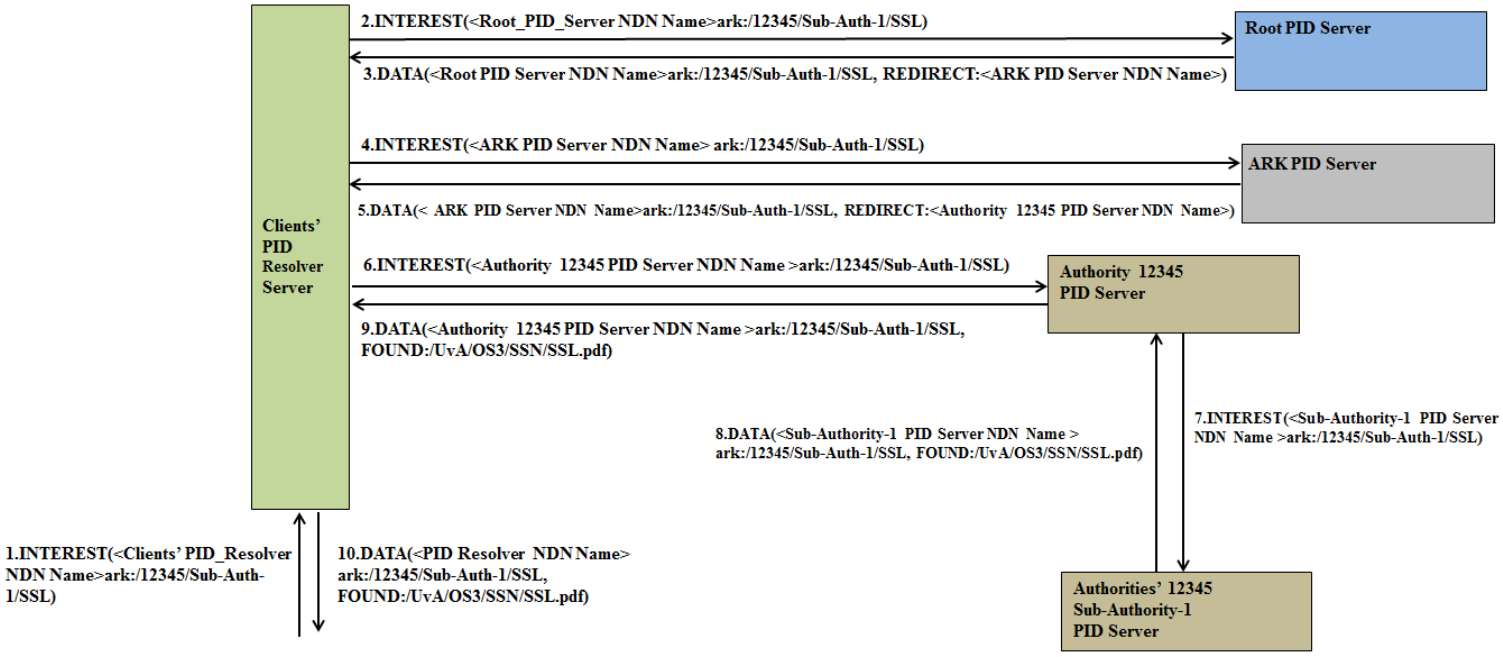


Fig. 10: PID Resolution Procedure in NDN

Upon arrival of the INTEREST packet in the Clients' PID resolver (arrow 1), it will extract the PID(*ark:/12345/Sub-Auth-1/SSL*) and will construct an INTEREST(<Root_PID_Server_NDN_Name>*ark:/12345/Sub-Auth-1/SSL*) packet and send it to its' default NDN gateway. This INTEREST packet (arrow 2) will be forwarded by the CRs to the Root PID Server which in turn will first extract the PID(*ark:/12345/Sub-Auth-1/SSL*) from the INTEREST packet and then the PID Type(*ark*) from the PID(*ark:/12345/Sub-Auth-1/SSL*). Extracting the PID Type from the PID is not trivial since the delimiter used (:) for separating the PID Type field from the Authority field in each PID standard defines where the PID Type field ends inside the PID. The Root PID server will then perform a look up on its' PID Type – PID Type NDN name mapping database and will construct the answer message REDIRECT:<ARK PID Server NDN Name> that will add in the DATA(<Root PID Server NDN Name>*ark:/12345/Sub-Auth-1/SSL*, REDIRECT:<ARK PID Server NDN Name>) and send it to the Clients' PID resolver server (arrow 3). This PID resolution procedure will continue (arrows 4 - 9) until eventually the Clients' PID resolver will receive the DATA packet (arrow 9) that contains the NDN name of the PID. At this point the Clients' PID resolver will construct the DATA packet (arrow 10) and send it to the client NDN enabled browser which can then construct an INTEREST(*/UvA/OS3/SSN/SSL.pdf*) packet, forward it to the NDN network and

receive the DATA(*/UvA/OS3/SSN/SSL.pdf*,<data>) packet that contains the data that corresponds to the PID(*ark:/12345/Sub-Auth-1/SSL*) requested.

In the aforementioned scenario the PID exists on the Mapping Architecture. In the case where the PID requested does not exists, the Clients' PID resolver will eventually receive a DATA packet that will contain in the <data> field the NOT_FOUND:<Description Message> that will in turn return to the Client.

Our Mapping Architecture porting in NDN ensures that if the NDN name of the PID requested for resolution exists, the clients' NDN enabled browser will eventually receive the DATA packet containing the NDN Name of the PID which in hence can use to get the actual data that correspond to the PID requested.

4.5 Summary of Gateway Architecture

In this chapter, we describe our proposed Mapping Architecture Design for resolving PIDs to their corresponding NDN names. Firstly, based on our design goals explained in section 1 and the hierarchical scheme of the PID standards, in section 2 we defined the name-space implementation of our Mapping Architecture. In section 3, we propose a hybrid PID resolution by the components of our Mapping Architecture in order to achieve the best geographical scalability. Finally, in section 4 we explained how the communication between the components of the Mapping Architecture can be achieved using the INTEREST and DATA packet specified in NDN.

5 Caching Strategies and Experimental Studies

Caching is an important mechanism in ICN for delivering data objects. In this chapter, we review the most well-known caching strategies implemented in NDN, and investigate how they influence the delivery of a Big Data object. The experiments in this chapter will be based on simulations.

5.1 Caching mechanisms

The key parameters of the caching mechanisms on an NDN network are the Forwarding Strategy (FS), the Decision Strategy (DS) and the Replacement Strategy (RS). Below, we briefly describe the functionality of each strategy and the most well-known algorithms used in each strategy.

5.1.1 Forwarding Strategy (FS)

The Forwarding Strategy (FS) is used to determine the route that an INTEREST packet will follow towards its' final destination in the network. Each router in the network that received an INTEREST packet for which it has no entry in his PIT will use the FS to decide in which output interface will send the INTEREST packet. There different FS algorithms for NDN networking some of which we briefly present below.

- Shortest Path Routing (SPR): In the shortest path routing FS the INTEREST packet is forwarded through the shortest path towards the repository that contains the requested content. SPR is the current FS that is used as the de facto FS in NDN.

- Nearest Replica Routing (NRR): In the Nearest Replica Routing FS the INTEREST packet is forwarded towards the nearest CR that has a replica of the requested content, or in the worst case scenario towards the server that hosts the permanent copy of the requested content. The implementation of the NRR requires the CRs to send meta-interest packets in order to discover a CR that contains the requested content.

5.1.2 Decision Strategy (DS)

The Decision Strategy (DS) is used to determine which router along the reverse path of an INTEREST request will cache the contents of the DATA packet. Each router in the reverse path of an INTEREST packet runs the DS to decide if it will cache or not the contents of the incoming DATA packet. There are different DS algorithms in the bibliography from which the most-well known and used are briefly described below.

- Leave Copy Everywhere (LCE): In the LCE DS, each router caches the contents of every DATA packet it receives. LCE is the DS currently used on NDN.

- Leave Copy Down (LCD) [37]: In the LCD DS, each router in the reverse path caches the contents of the DATA packet it receives only if the previous router in the reverse path had a replica of the contents on its' Content Store or if it had received the DATA packet from the permanent repository (the publisher itself) meaning that it was the first CR in the reverse path. Therefore, in LCD only one CR caches the content of the DATA packet on its Content Store (CS).

- Fix Probability (FIX(p)): In the FIX(p) DS, each CR on the network caches the incoming DATA packet on its' Content Store with a probability p(e.g. with p=0.5 each CR caches the half of the DATA packet it receives).

- Probabilistic In-Network Caching (ProbCache) [24]: In the ProbCache DS algorithm, the DATA packets are cached by each CR in the reverse path with a different probability. In general, in ProbCache, CRs that are closer to the node (CR or Server) from which the content of the requested object was retrieved have higher probability in caching the received DATA packet.

5.1.3 Replacement Strategy (RS)

Replacement Strategy (RS), is used to determine which object in the Content Store will be pulled out in order to make space for the new incoming object to be pushed in. The replacement algorithm runs on each router in the NDN network. There are many RS algorithms for NDN in the bibliography, from which the most-well known and widely used are briefly described below.

- First In First Out Cache (FIFO): In the FIFO cache RS, each router replace the Object that was first pushed in the Content Store.

- Random Cache (Random): In the Random cache RS, each router replaces an object randomly in order to make space for the incoming one.

- Least Recently Used (LRU): In the LRU RA, each router replaces the least recently used Content. The LRU, is the most used replacement algorithm within the literature and the replacement algorithm used for NDN.

- Least Frequently Used (LFU): In the LFU RA, each router replaces the least frequently used Content.

5.2 Simulation Methodology and Scenarios

In this section, we describe the simulation parameters used in our scenarios, the simulation metrics along with the method we use for collecting these metrics. Since our main focus is on Big Data objects our main simulation goal were to investigate how the Content Store size (C) of a CR to the Big Data object size (B) ratio (C:B) affect the performance of the caching mechanisms used in NDN. Moreover, we investigate if the segmentation of the Big Data objects to multiple equal sized sub-

objects (c) affects the performance of the caching mechanisms. The ccnSIM [9] caching simulator for NDN was used as the platform for performing all the simulation experiments.

5.2.1 Simulation setup

Table 6 shows an overview of the simulation parameters investigated in this research project. Afterwards, follows a brief description of each parameter.

Parameter	Description	Values
R	Big Data Repository Size	51.2TBytes
|R|	Num. of Big Data Objects in R	150
B	Size of Big Data Object	350GBytes
c	Num. of equal size sub Objects a Big Data Object is segmented	[1,2,4,6..20]
C	The Content Store Size in each Content Router expressed as Size of Big Data Object	[0.5B,1B,2B,6B,8B]
a	Zipf exponent	1
FS	Forwarding Algorithm	SPR
DS	Caching Algorithm	[LCE,LCD,FIX(0,5),FIX(0.25),ProbCache,NoCache]
RS	Replacement Algorithm	LRU
T	The Big Data Object Client has request and received so far	-
Network Topologies	The network topologies used in simulations	String(figure 11), Binary Tree (figure 12)

Table 6: Overview of Simulation Parameters Investigated in this Research Project

- **Big Data Repository(R,|R|,B,c,a)**

In order to determine a realistic size for our Big Data Repository we investigate different public Big Data repositories available in the Open Data Science Cloud(ODSC) [38].Table 7 shows an overview of the public Big Data Sets available on OSDC.

Big Data Repository	Size (TBytes)	Num. of Big Data Objects	Average Size of Big Data Objects (GBytes)
Complete Genomics Public Data	50.4	150	350
Earth Observing-1 Mission	80.5	Unknown	Unknown
ASTER	23.7	Unknown	Unknown

Table 7: Public Data Sets Available in ODSC

From our investigation on public Big Data sets we find out that Big Data Repositories can have a size of 25-100TBytes (e.g. Complete Genomics Public Data [39], Earth Observing-1 Mission [40]), and contain Big Data objects of approximately 350GByte(e.g. In Complete Genomics Public Data each Genome has approximately 350 GBytes of information [41]). Based on these observations in our simulation experiments we consider a Big Data Repository size equal to 51.2TBytes(R), 150 Big Data objects (|R|) each one with 350 GBytes size (B). Due to the fact that Big Data Objects consumes a big amount of memory to be saved as one piece of data, we wanted to investigate how the number of equal size sub-Objects a Big Data Object is segmented can affect the performance of the caching strategies. Therefore, in our simulations, we investigate different number of equal size sub-Objects (c) a Big Data Object is segmented. In the case where the Big Data Object is consisted of multiple sub-Objects a Client that wants to receive a Big Data Object, starts sending request for the first sub-Object and when it received the content it proceeds with a request for the second sub-Object and so on until it has received all the sub-Objects of the Big Data Object.

Finally, as we could not have any insights on the distribution of the popularity of the Big Data Objects, in our experiments, the popularity of the Big Data Objects is calculated as a single sequence using Zipf law [42] with exponent (alpha) parameter set to 1.

- **Content Routers (C)**

As we wanted to investigate how the Content Store size (C) of each CR affects the efficiency of the caching strategies, we used variable Content Store size starting with CS size of half of the Big Data Object size and we doubled it each time up to the point where the Content Store size is eight times bigger the Big Data Object size.

- **Caching Strategies(FS,DS,RS)**

For the caching strategies:

  - SPR described earlier in this chapter as the FS algorithm. Our choice was based in the fact that SPR is the current FS algorithm used in NDN.
  - LRU also described earlier in this chapter as the RS algorithm. Here our choice was based in the fact that is the most-well known and used RS.
  - For the DS algorithms we used LCE, LCD, FIX(0.5), FIX(0.25), ProbCache and NoCache. As we wanted to see how the different DS algorithms proposed in the bibliography perform when they are used for Big Data Objects we choose to evaluate all the well-known DS algorithms in our simulations. NoCache was used in order to have a comparison of the other DS algorithms with CRs that do not cache any objects.

In our simulations when the Big Data Object is segmented to multiple equal size sub-objects, each sub-object of a Big Data Object is treated by the Caching

strategies independently of all the others. That means that a sub-object of a Big Data Object may be cached by a CR and another may be not cached

- **Clients Request (T)**

The parameter T indicated how many Big Data Objects a client has request and received so far. In the case of multiple clients each client in the simulation have to finish the request and the reception of the Big Data Object T before any other client proceeds in requesting the T+1 Big Data Object.

- **Network Topologies**

String and binary tree as shown in figure 11 and figure 12 are the network topologies investigated in our simulations. In both topologies the distance of each client from the repository is 5 hops (4 CRs and 1 Hop for the Big Data Repository).

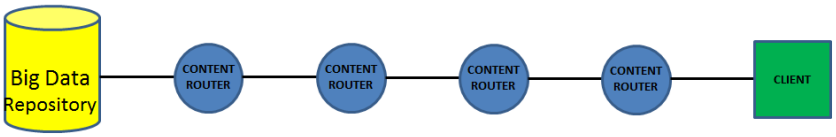


Fig. 11: String Topology

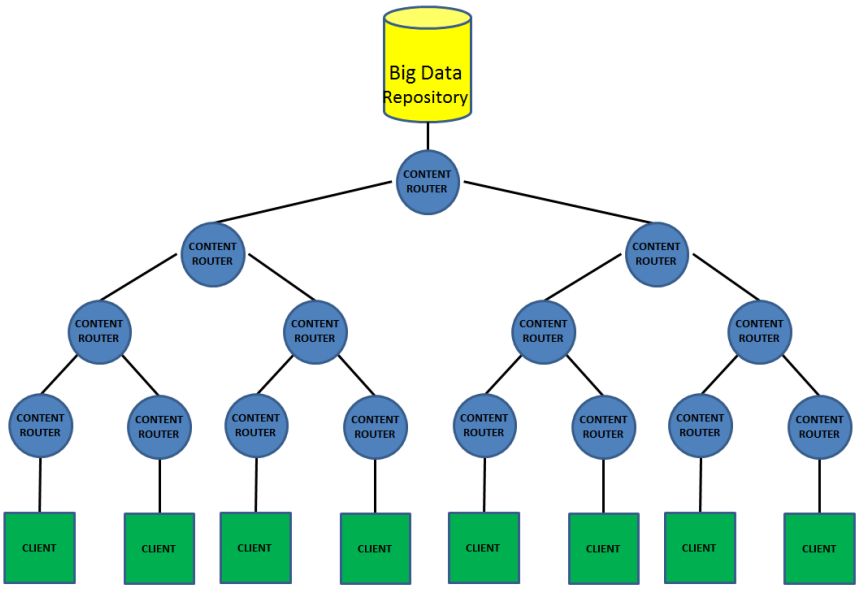


Fig. 12: Binary String Topology

5.2.2 Performance Metrics

The main goal of the in–network caching in ICN can be described by the following three aspects:

- From a customer point of view, ICN aims to reduce the average time for downloading a requested content.
- From the provider point of view, ICN aims to reduce the number of requests that the provider needs to serve.
- From the network point of view, ICN aims to reduce the network traffic.

All the above aspects can be express as the average number of Content Routers(Hops), a customer request need to travel before finding a temporary(in a routers Content Store) or a permanent copy(In the Providers Repository) of the requested object. Therefore, in our simulation the metric that we used to describe the benefit for all the aforementioned aspects is the average number of hops a request needs to be routed in order to find the requested object.

5.2.3 Collection of measurements

As described above the main metric that we use to evaluate the performance our simulation scenarios is the average number of Hops that the request from the clients need to be routed before hitting the Big Data Object requested. One of the issues to take in to account when taking measurements within a network of caches is when to start collecting the metrics. In our simulation scenarios we start to collect the average number of hops metric when the average number of hops converges. More specifically, each simulation scenario runs until the average number of hops metric is stable for at least 50T as shown on figure 13.

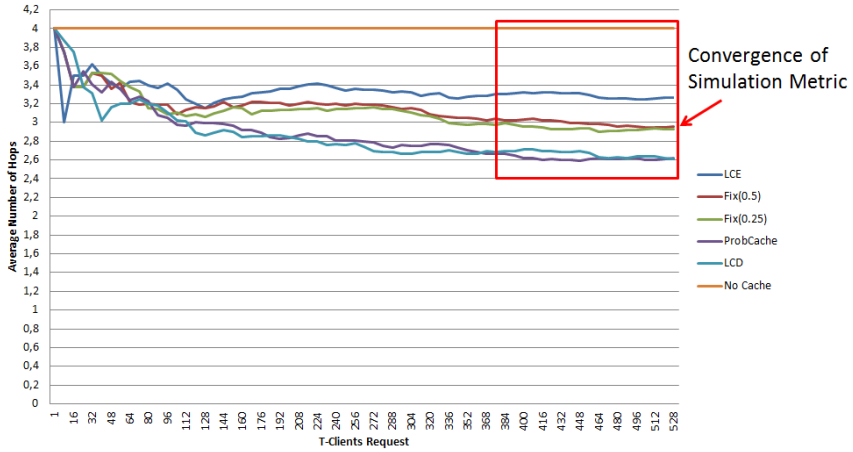


Fig. 13: Collection of measurements at convergence point

By performing the simulations described above, we can have valuable results for evaluating the performance of in-network caching for the most well-known caching strategies, the effectiveness of segmenting the Big Data Object in multiple equal size sub-Objects and the impact introduce by the CR Content Store size (C) to the Big Data Object size (B) ratio (C:B).

5.3 Results and Evaluation

In this chapter, we present and analyze the results of our simulations. First we present and analyze the result for the string topology and then we proceed with the results of the binary tree topology. For each topology, we first evaluate how the cache size (C) of a CR to the Big Data object size (B) ratio (C:B) effects the performance of the caching strategies and then we proceed on how the number of equal sized sub-objects a Big Data object is segmented, effects this metric. Finally, for each topology we explain which caching decision (DS) algorithm gives the best performance results.

5.3.1 String Topology

The results gathered from the simulations of the string topology are shown on figure 14. X-axis shows the ratio of the Content Router Cache Size to the Big Data Object Size (C:B). Y-axis shows the average number of hops metric collected at the convergence point as described on the previous section. The dotted points in the graph shows the average value of the different number of equal size sub-Objects a Big Data Object is segmented and the error bar indicates the standard deviation.

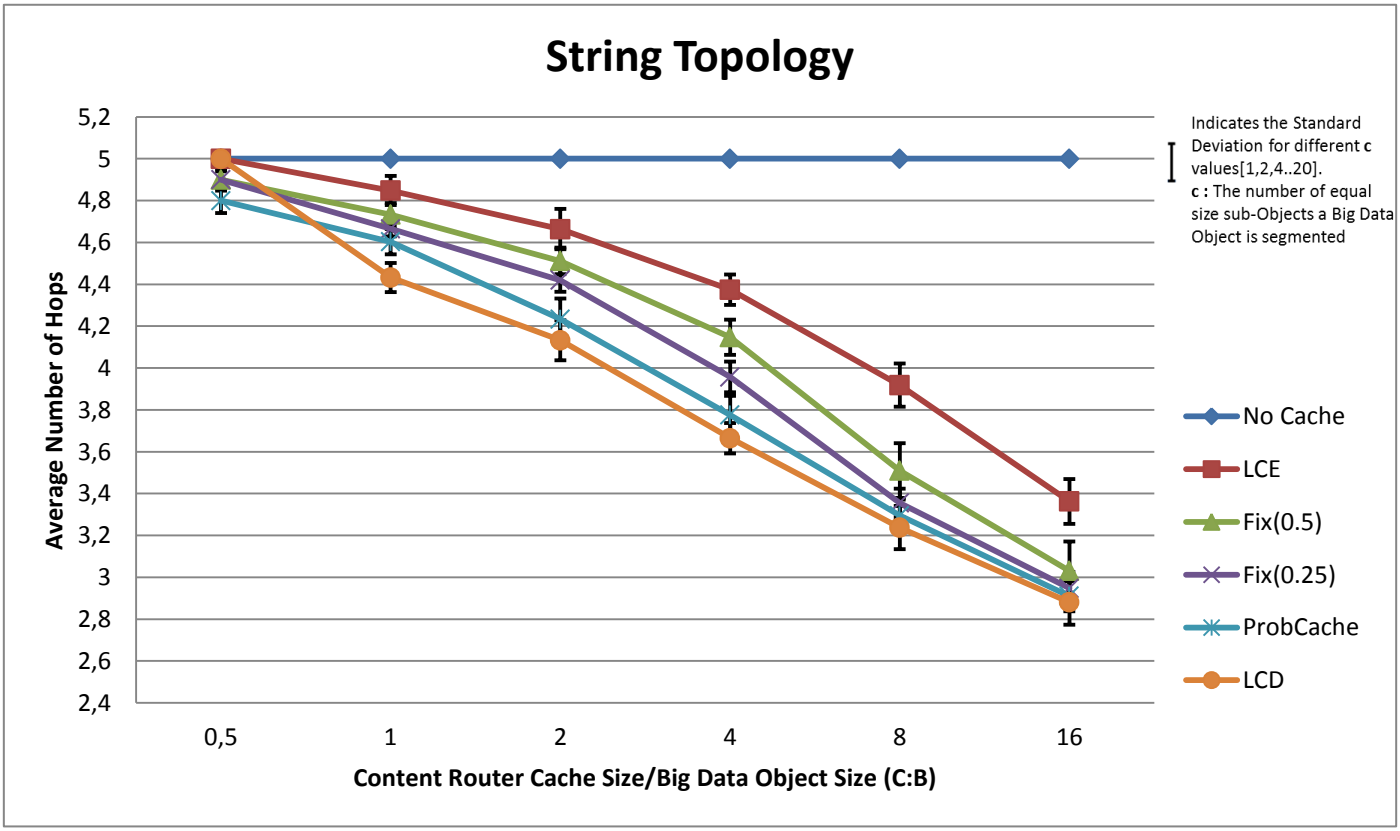


Fig. 14: String Topology Simulation Results

- CRs' cache size (C) to Big Data Object size(B) ratio (C:B):

  As we can clearly see from the graph the ratio of Content Router Cache Size to Big Data Object Size (C:B) significantly effects the performance of the Caching Algorithms. More specifically, for C:B $\leq$ 1 the performance of the Caching Algorithms does not give any significant improvements in the

average number of hops metric. On the other hand for C:B ≥ 2 we can see that for the LCD and ProbCache DS algorithms the average number of hops decreases by approximately 18%, thus significant gains on the performance metrics can be gained from that point and onwards.

- Number of equal sized sub-objects a Big Data object is segmented (c):

  Referring to the graph of figure 14 we can see that the error bar that indicates the standard deviation for all the different values of c has a small variation for all points in the graph. From this result, we can clearly see that the number of equal size sub-objects a Big Data Object is segmented does not significantly affect the performance of the CD algorithms.

- Performance of different Caching Decision Strategy (DS) algorithms.

  Referring again to the graph of figure 14 we can see that there is a significant difference on the performance of the DS algorithms investigated in our simulations. More specifically, the LCE algorithm that is the current DS algorithm used in NDN gives significant performance benefits at the point where C:B ≥ 8. On the other hand, the ProbCache and the LCD DS algorithms are found to give the best performance gains since they manage to give approximately the same performance results with the LCE at the point where C:B ≥ 2. Finally, the Fix(p) DS algorithms also perform better than the LCE.

5.3.2 Binary Tree Topology

The results gathered from the simulations of the binary tree topology are shown on figure 15. Again in this graph the axis, the dotted points and the error bar has the same meaning as in the string topology.

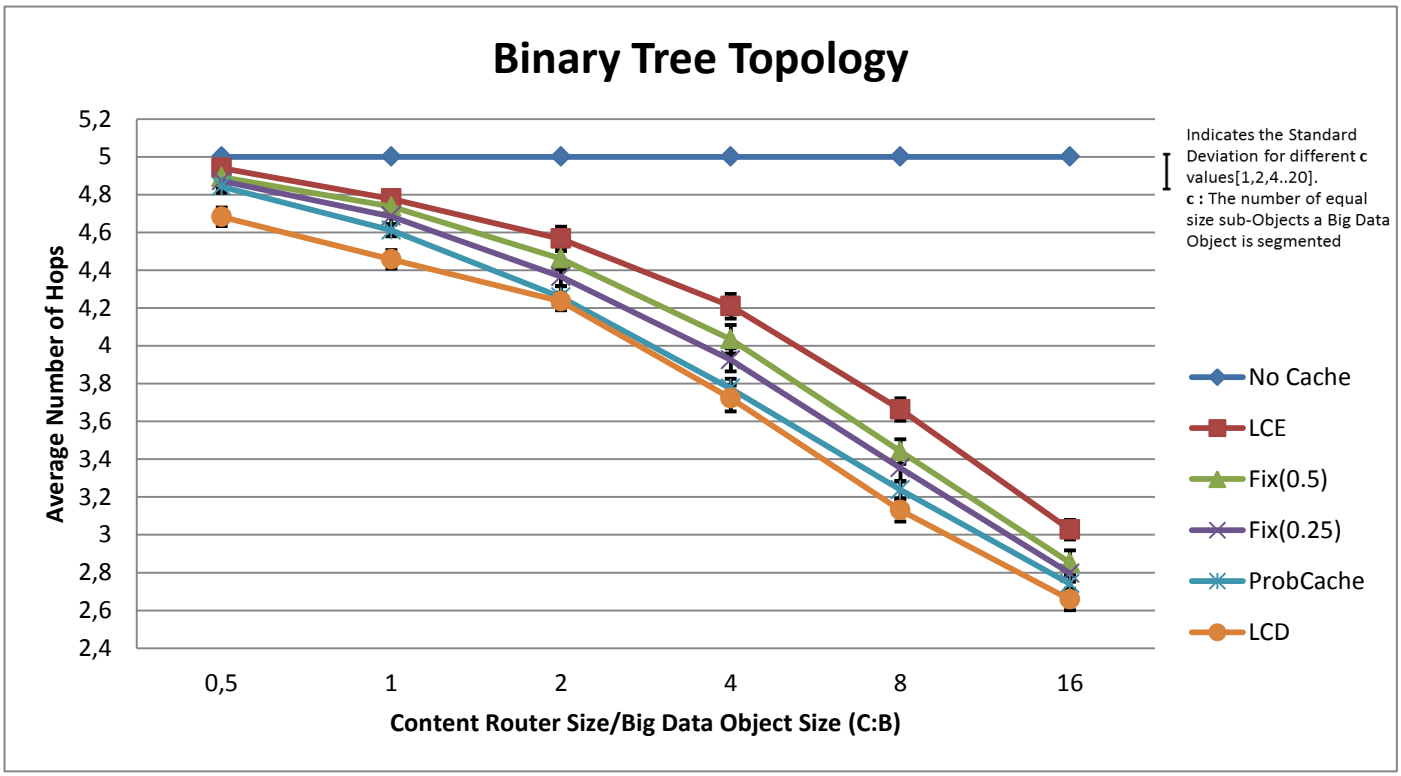


Fig. 15: Binary Tree Topology Simulation Results

- CRs' cache size (C) to Big Data Object size(B) ratio (C:B):

  As we can see from the graph the results are the same as in the case of the string topology. More specifically, the performance of caching algorithms give significant benefits at the point where C:B $\geq$ 2 while for C:B $\leq$ 1 the gains are insignificant.

- Number of equal sized sub-objects a Big Data object is segmented (c):

  Again as in the case of the string topology, the number of equal size sub-objects a Big Data object is segmented (c) does not affect the performance of the CS. The only significant difference that can be observed is that the standard deviation values are smaller compared to the case of the string topology.

- Performance of different Caching Decision (CS) algorithms:

  The performance of the CS algorithms for the case of the tree topology is again approximately the same as in the case of the string topology. The only significant difference that can be observed compared to the results of the string topology is the performance of the LCD when the C:B = 0.5. At this point the LCD in the binary tree topology has an average number of hops of approximately 4.6 while at the string topology the average number of hops is 5. This difference is based on the fact that in the binary tree topology, there are multiple clients requesting Big Data objects, thus some clients are benefit for Big Data objects that are already cached in the CRs because other clients has already requested them. This result gives motivation for evaluating the performance of the in-network caching for more complex networks

To finalize, our results for both topologies, showed us that in order to have significant benefits for In-Network caching in NDN the Content Router Cache Size has to be at least twice the size of the Big Data Object size and that the number of equal size sub-Objects a Big Data is segmented does not affect the performance of the caching algorithms. Moreover, we find out that the LCE which is the current CS algorithm used in NDN needs at least a C:B $\geq$ 8 to give significant performance benefits. On the hand, LCD and ProbCache are more promising CS algorithms for delivering Big Data objects since they give significant performance benefits from the point where C:B $\geq$ 2 and onwards.

6 Discussion

In this section, we discuss the advantaged and disadvantages of the proposed Mapping Architecture, and the challenges with respect of the efficiency of in-network caching for delivering Big Data in NDN.

During this research project a generic, extensible, scalable and efficiency Mapping Architecture was proposed for resolving PIDs to NDN names. Although we show the requirements that our solution needs in order to meet our design goals there are other issues that can arise during the implementation of the Mapping Architecture. More specifically, since each PID standard is allowed to use different delimiters for separating the different fields (PID type, Authority, Sub-Authority, Name of Digital Object) the PIDs it publishes, it is not possible to have a universal software (like BIND in DNS) that could be installed for every new PID type, Authority or sub-Authority. As a result, each PID type and authority must develop its' own software that would be in compliance with the scheme that they use for publishing their PIDs in order to be able to enter our Mapping System. One way to over-come this burden is to specify a framework in which a general scheme with some elasticity will be provided in order to limit the big number of possible different PIDs schemes that could be chosen by the authorities. Based on this framework, a universal software could be developed and each authority that use this framework can also use this software instead of developing its' own.

After the proposed Mapping Architecture was introduce, we conducted experimental studies by performing simulations in order to investigate what benefits can Big Data objects have from the in-network caching of NDN. Our results showed that significant benefits can be gained from the point where the cache size of the Content Routers is double the size of the Big Data Objects. However the amount of cache size that a Content Router need in the case of Big Data objects would be more than 1TBytes. Adding 1TBytes in a Content Router cache size that needs to precede data at line speed is a hardware limitation even in todays' state of the art Routers. However, the technology of fast SSD is rapidly developed and we believe that in the near future implementing TBytes of cache size in a Content Router will be possible.

7 Conclusions

This research project is a preliminary study on how Information Centric Networking (ICN) can efficiently be used for delivering Big Data with Persistent Identifiers (PIDs). Firstly, we reviewed the state of the art approaches in ICN and the current PID standards, we chose the most mature ICN approach (NDN) and we proposed a Mapping Architecture for resolving PIDs to NDN names. After, we evaluate the efficiency of the in-network caching mechanisms proposed for NDN when delivering Big Data objects.

The main design goals of our Mapping Architecture was to provide a solution that support all the current PID standards, allow new PID standards to be ported in and scale for a vast number of PIDs. Based on these goals, we design a name-space implementation in which the PID resolution process along with the name-space management is hierarchically distributed in a tree structure across multiple components. We showed that the hierarchy layers defined in our name-space implementation can support all the current PID standards and allows new PID standards to be easily ported in. Moreover, we showed that by assigning specific requirements for each hierarchy layers' components, the implementation of the PID resolution can efficiently scale for a vast number of PIDs. Finally, we showed how the Mapping Architecture can be efficiently ported in NDN.

In order to investigate the efficiency of the NDN in-network caching mechanisms when delivering Big Data objects, we performed experimental studies based on simulations. Our results showed that the cache size of the Content Routers (CR) to the Big Data object ratio (C:B) plays an important role in the efficiency of the in-network caching mechanisms in NDN. Specifically, for $C{:}B \leq 1$ the in-network caching in NDN has no significant performance benefits, while for $C{:}B \geq 2$ significant performance benefits can be gained for delivering Big Data objects. Furthermore, the number of multiple equal sized sub-objects a Big Data object is segmented does not affect the performance of the in-network caching mechanisms in NDN.

To finalize, based on the work done in this research project we can say that ICN is a promising approach for delivering Big Data with Persistent Identifiers (PID) that should be taken into account and further researched.

8 Future Work

Future work can be done both on improving the Mapping Architecture proposed and evaluating the efficiency of caching mechanism for delivering Big Data objects. For the Mapping Architecture future work can be done in specifying the exact protocol for the communication between the components. Moreover, security mechanisms for the Mapping Architecture were not taken into account during the design phase, thus implementing secure communication between the components is another topic for future work. For the efficiency of in-network caching mechanisms when delivering Big Data objects further research is need to be done. More specifically, during our experimental studies the only forwarding strategy investigated was the shortest path routing towards the Big Data repository and experimental studies were based on two simple network topologies (string and binary tree). The efficiency of in-network caching mechanisms for more forwarding strategy algorithms (e.g. NRR) and more complex network topologies must be also researched and evaluated.

9 References

[1] S. Sagiroglu and D. Sinanc, "Big Data: A Review", *in IEEE International Conference on Collaboration Technologies and Systems(CTS)*, May 2013.

[2] Environmental Research Infrastructures (ENVRI) project. [Online] ENVRI project. Available: http://envri.eu/.

[3] TCP Throughput Over Long Fat Networks. [Online] ITPerformanceManagement. Available: http://itperformancemanagement.blogspot.nl/2010/04/tcp-throughput-over-long-fat-networks.html, April 2010.

[4] TCP Tuning. [Online] Wikipedia. Available: http://en.wikipedia.org/wiki/TCP_tuning, June 2014.

[5] TCP Extensions for Multipath Operation with Multiple Addresses. [Online] IETF. Available: http://tools.ietf.org/html/rfc6824, January 2013.

[6] Information-Centric Networking Research Group (ICNRG). [Online] IRTF. Available: https://irtf.org/icnrg.

[7] Persistent Identifer. [Online] Ariadne. Available: http://www.ariadne.ac.uk/issue56/tonkin, July 2008.

[8] G. Xylomenos et. al. "A Survey of Information-Centric Networking Research", *in IEEE Communications Surveys & Tutorials (Volume:16,Issue: 2)*, July 2013.

[9] R. Chiocchetti et. al. "ccnSim: an Highly Scalable CCN Simulator" *in IEEE International Conference on Communications (ICC)*, June 2013.

[10] T. Koponen et. al. "A data-oriented (and beyond) network architecture", *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communication*, October 2007.

[11] Named Data Networking (NDN). [Online] NDN. Available: http://named-data.net/.

[12] Content Centric Networking Project. [Online] CCNx. Available: http://www.ccnx.org/.

[13] Named Data Networking (NDN) project. [Online] PARC. Available: https://www.parc.com/publication/2709/named-data-networking-ndn-project.html.

[14] NDN Specification Documentation. *[Online] NDN. Available: http://named-data.net/wp-content/uploads/2013/11/packetformat.pdf*, March 2014.

[15] Uniform Resource Identifier (URI): Generic Syntax. [Online] IETF. Available: http://tools.ietf.org/html/rfc3986, January 2005.

[16] L. Wang et. al. "OSPFN: An OSPF Based Routing Protocol for Named Data Networking", *Technical Report, NDN-0003*, July 2012.

[17] PURSUIT. [Online] FP7. Available: http://www.fp7-pursuit.eu/PursuitWeb/.

[18] PSIRP. [Online] PSIRP. Available: http://www.psirp.org/.

[19] J. Rajahalme et. al. "On name-based inter-domain routing", *Computer Networks (Vol. 55, no. 4)*, March 2011.

[20] SAIL Project. [Online] SAIL. http://www.sail-project.eu/.

[21] FP7 4WARD Project. [Online] FP7. Available: http://www.4ward-project.eu/.

[22] Deepali D. Ahir, Sagar B. Shinde. "Caching Simulators for Content Centric Networking", *International Journal of Science and Research(IJSR)*, May 2014.

[23] D. Rossi, G. Rossini. "Caching performance of content centric networks under multi-path routing (and more)", *Technical report, Telecom ParisTech*, November 2011.

[24] I. Psaras et. al. "Probabilistic In-Network Caching for Information-Centric Networking", *Proceedings of the second edition of the ICN workshop on Information-centric networking*, August 2012.

[25] W. K. Chai et. al. "Cache Less for More in Information-Centric Networks", *Proceedings of the 11th international IFIP TC 6 conference on Networking*, May 2012.

[26] F. MvCown et. al. "The Availability and Persistence of Web References in D-Lib Magazine", *5th International Web Archiving Workshop (IWAW05)*, November 2005.

[27] D. Spinellis. "The decay and failures of web references", *Magazine Communications of the ACM(Volume: 46, Issue: 1)*, January 2003.

[28] URN:NBN RESOLVER FÜR DEUTSCHLAND UND SCHWEIZ. [Online] Available: https://nbn-resolving.org/.

[29] The Handle System. [Online] Handle. Available: http://handle.net/.

[30] UW-Madison Libraries Handle System.[Online] Madison, University of Wisconsin. Available: http://digital.library.wisc.edu/webhandle/.

[31] The DOI System. [Online] DOI. Available: http://www.doi.org/.

[32] DOI Resolver. [Online] DOI. Available: http://dx.doi.org/.

[33] DOI Registration Agencies. [Online] DOI. Available: http://www.doi.org/registration_agencies.html.

[34] ARK (Archival Resource Key) Identifiers. [Online] California Digital Library. Available: https://wiki.ucop.edu/display/Curation/ARK.

[35] PURL. [Online] PURL. Available: http://purl.oclc.org/docs/index.html.

[36] PURL. [Online] PURL. Available: http://www.oclc.org/research/activities/purl.html?urlm=160105.

[37] N. Laoutaris et. al. "The LCD interconnection of LRU caches and its analysis", *in Performance Evaluation Journal (Volume: 63, Issue: 7),* July 2006.

[38] Public Data Sets. [Online] Open Data Science Cloud(ODSC). Available: https://www.opensciencedatacloud.org/publicdata/.

[39] Complete Genomics Public Data. [Online] Open Data Science Cloud (ODSC) Available: https://www.opensciencedatacloud.org/publicdata/complete-genomics-public-data/.

[40] Earth Observing-1 Mission. [Online] Open Data Science Cloud (ODSC). Available: https://www.opensciencedatacloud.org/publicdata/earth-observing-1-mission/.

[41] Public Genome Data Repository. [Online] Open Data Science Cloud (ODSC). Available: http://media.completegenomics.com/documents/PublicGenomes.pdf.

[42] Lada A. Adamic, Bernando A. Huberman. "Zipf's law of the Internet", *Glotometrics 3:143-150,* 2002.