UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODOLFO STOFFEL ANTUNES

Leveraging Relations among Objects to Improve the Performance of Information-Centric Networks

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Marinho Pilla Barcellos

Porto Alegre
February 2016

CIP — CATALOGING-IN-PUBLICATION

Antunes, Rodolfo Stoffel

Leveraging Relations among Objects to Improve the Performance of Information-Centric Networks / Rodolfo Stoffel Antunes. – Porto Alegre: PPGC da UFRGS, 2016.

141 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2016. Advisor: Marinho Pilla Barcellos.

1. Content Distribution. 2. Information-Centric Networks. 3. Performance Evaluation. I. Barcellos, Marinho Pilla. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Luis da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“It is impossible for a man to learn
what he thinks he already knows.”
— Epictetus

“Books are not meant to be believed, but to be subjected to inquiry.
When we consider a book, we must not ask ourselves what it says,
but what it means.”
— Umberto Eco

“What is the worst thing that can happen in a pressure cooker?
Science.”
— Randall Munroe

ACKNOWLEDGEMENTS

First of all, I must thank my parents, Antonio and Magdalena, for the support they provided while this thesis was developed. I am well aware that I often was a grumpy creature in the last few months due to my worries about this study. Nevertheless, they understood that this challenge would eventually end and never ceased to encourage me to get it done.

Family support aside, this thesis proposal is a result from collaborations with various people I met during my Ph.D. I do not claim that those named here are the most important. Rather, these are the names I remember at the moment of this writing. Since I cannot guarantee that all deserving people will be mentioned, I leave here an acknowledgement to those people not listed but that in some way contributed to this thesis.

The first person that must be acknowledged here is my Ph.D. advisor, Prof. Marinho Barcellos, for his utmost effort in providing me with the research background and expertise required to conduct this study. Despite my stubbornness to understand some of his views about my work, he was many times successful in making me see the flaws on my research methodology. Among the various lessons learned from him, one worth mentioning is to never argue in favor of an idea if you are half-hearted about it and half-read the literature about the subject. I would like to thank Prof. Luciano Gaspary for his contributions to this thesis and the opportunity to contribute to GT-ICN project, which was an invaluable opportunity to acquire practical knowledge about my research topic. Last but not least, I would like to thank Prof. Christian Rothenberg for his contributions to my paper published at IM 2015.

I would like to dedicate a small space here to thank the staff at the UFRGS's Informatics Institute. They may have not directly contributed to this thesis development, but their efforts certainly helped ease my routine to go on with my work. Namely, I would like to thank Elisiane for helping students go through all the bureaucratic nonsense from Brazilian educational institutions. Also, I would like to thank Luis Otávio for his utmost efforts in keeping the Institute infrastructure working as best as possible.

Among my colleagues, I would like to thank Matheus Lehmann and Rodrigo Mansilha for their uncountable efforts, including discussions and reviews, that undoubtedly contributed to the impact of this research. I also would like also thank Matheus for never letting the lab fall into boredom due to his controversial sense of humor and political views. To Mansilha, I extend my gratitude for his philosophical teachings about how to

structure an argument hierarchically using two points, two issues and two aspects. To Bruna Fiorentin I would like to thank for using my thesis as background for her graduation work and resulting collaboration with my study. To the remaining of my lab mates, I would like to thank for the companionship in the last few years and the many insights about: research, life, the universe, grumpy owls, high tension, relationships, mental illnesses, good music, not so good music, and everything else. The gang from lab 208 includes: Barata, Bátimã, Bays, Daniel, o Faixa, Lucas, Miguel, Rodrigo Oliveira, Rafael Coelho, Rafael Fonte, Rafael Hansen, Weverton. I would like to name everyone in the Computer Networks Research Group, but the list would be too long, so I will summarize it thanking the inhabitants from labs 210, 212 and 221.

Outside UFRGS, I would like to thank a small group of friends that tried their best to stay in touch, even after our graduation from college, with the ultimate goal of not letting the nonsense fade away. Namely: Ana Paula, César, Denise, Dormento, Fabiane, Hisham, Jezer, Klaser, Mierlo, Mikito, Raisa, Renata, Reverendo, Sandra. A special mention goes to a select group that introduced me to the fine art of beer appreciation at Porto Alegre.

Last but not least, I would like to show a bit of gratitude to two tireless servants that spent days and nights doing everything they could to help me test my ideas, get my texts written and provide me some entertainment along the way. These are my notebook, Wanderer, and my desktop, Asgard. Without them, this text would not have been written or, in the best scenario, would have been written in a typewriter.

ABSTRACT

Information-Centric Networking (ICN) is a communication paradigm created to align the network infrastructures to the needs of content distribution systems. ICN employs routing and caching mechanisms tailored to fulfill requests for uniquely identified *data objects* not associated to a fixed locator. So far, research about ICN focused primarily on evaluating architectural aspects, such as the performance of different routing and caching schemes. However, the method applied to distribute data using the concept of objects can also impact communications in an ICN. In this thesis, we explore a model that enables the distribution of contents as multiple data objects. We employ the concept of *relations*, defined as links between two objects indicating that the data from one complements in some way the data from the other. Our model based on relations enables clients to identify and retrieve the data pieces required to reconstruct a content. It is application agnostic, supports different relation structures, and is backward-compatible with current ICN specifications. We also discuss the main design aspects related to the implementation of the model in the Named Data Networking (NDN) architecture. To evaluate how relations impact network and application performance, we perform a series of experiments with two case studies based on relevant scenarios from the current Internet, namely: multimedia content and Web pages. The multimedia case study explores a favorable scenario in which relations present a negligible overhead in contrast to the high volume of content data. Results from this case study show that, compared to the standard NDN implementation, relations can reduce download times by 34% and network traffic by 43%. In turn, the Web pages case study explores a scenario in which relations generate a non-negligible impact on the network and applications. The analysis of this scenario shows that, even with the additional overhead incurred by relations, the mechanism can reduce on average 28% client download time, and 34%, global network traffic.

Keywords: Content Distribution. Information-Centric Networks. Performance Evaluation.

Utilizando Relações entre Objetos para Melhorar o Desempenho de Redes Orientadas a Conteúdo

ABSTRACT

Redes Orientadas a Conteúdo (*Information-Centric Networks*, ICN) são um novo paradigma de comunicação criado para aproximar as infraestruturas de rede às necessidades de sistemas de distribuição de conteúdo. ICN utiliza mecanismos de roteamento e cache projetados para atender requisições por *objetos de dados* unicamente identificados e desassociados de um localizador fixo. Até o momento, pesquisas sobre ICN focaram principalmente na avaliação de aspectos arquiteturais, tais como o desempenho de diferentes esquemas de roteamento e cache. Entretanto, o método aplicado para distribuir dados utilizando o conceito de objetos também pode impactar a comunicação em uma ICN. Esta tese explora um modelo que permite a distribuição de um conteúdo através de múltiplos objetos de dados. Emprega-se o conceito de *relações*, definidas como elos entre dois objetos indicando que os dados de um complementam de alguma forma os dados do outro. Tal modelo baseado em relações permite que clientes identifiquem e recuperem os objetos necessários para a reconstrução do conteúdo. Ele é agnóstico ao formato de dados das aplicações, suporta diferentes estruturas de relações e é retrocompatível com especificações atuais de arquiteturas ICN. Também discute-se os principais aspectos de projeto relativos à implementação do modelo na arquitetura NDN. Para avaliar o impacto de relações no desempenho da rede e aplicações, foi realizada uma série de experimentos com dois estudos de caso baseados em cenários relevantes da Internet atual, sendo eles: conteúdo multimídia e páginas Web. O estudo de caso sobre conteúdo multimídia explora um cenário favorável, no qual relações apresentam uma sobrecarga negligível em contraste ao grande volume de dados dos conteúdos. Os resultados deste estudo de caso mostram que, em comparação com a implementação padrão do NDN, o uso de relações pode reduzir os tempos de download em 34% e o tráfego de rede em 43%. Por sua vez, o estudo de caso sobre páginas Web explora um cenário no qual relações geram um impacto não negligível na rede e aplicações. A análise deste cenário mostra que, mesmo com a sobrecarga adicional gerada pelas relações, o mecanismo pode reduzir, em média, o tempo de download dos clientes em 28% e o tráfego de rede em 34%.

Keywords: Redes Orientadas a Conteúdo, Distribuição de Conteúdo, Análise de Desempenho.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AS	Autonomous System
BGP	Border Gateway Protocol
CCN	Content-Centric Networking
CDN	Content Distribution Network
CIM	Common Information Model
CR	Content Routers
CS	Content Store
DHT	Distributed Hash Table
DNS	Domain Name System
DO	Data Object
FIB	Forward Information Base
FN	Forwarding Node
ICN	Information-Centric Networks
IO	Information Object
HTML	Hypertext Markup Language
LCE	Leave Copy Everywhere
LRU	Least Recently Used
NDN	Named-Data Networking
NDNrel	Named-Data Networking with Relations
NFD	Named-Data Networking Forwarding Daemon
NRS	Name Resolution Service
NetInf	Network of Information
PIT	Pending Interest Table
PSIRP	Publish Subscribe Internet Routing Paradigm

RN	Rendezvous Node
rID	Rendezvous Identifier
sID	Scope Identifier
TM	Topology Manager
URL	Uniform Resource Locator

CONTENTS

1 INTRODUCTION

The current Internet Protocol stack was designed to enable remote clients to access expensive computational resources that were available in a select number of institutions (CLARK, 1995). Since then, the Internet and its usage scenarios evolved beyond initial expectations. Nowadays, it is a global network mainly used for distribution of a wide variety of digital contents. For example, video distribution applications alone account for 66% of global network traffic in recent years (CISCO, 2016).

The original Internet Protocol stack requires additional functionality to cope with the scale and demand from current content distribution systems. This fact is evidenced by the advent of application layer mechanisms for high-scale content distribution, such as Peer-to-Peer (P2P) systems (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004) and Content Distribution Networks (CDNs) (PALLIS; VAKALI, 2006). In particular, CDNs are the most widely used mechanism for global-scale content distribution, generating nearly 36% of the global network traffic (CISCO, 2016). These technologies helped content distribution reach its current scale. However, both P2P systems and CDNs also present drawbacks. For example, P2P systems are known for their high inter-domain traffic generation due to lack of knowledge about underlying network infrastructure. In turn, CDN infrastructures quickly become expensive and complex to manage when used to distribute an extensive catalog of popular contents.

The facts presented above combined with the efforts to design networking architectures for the Future Internet led to the conception of the *Information-Centric Networking* (ICN) paradigm (AHLGREN et al., 2012). Originally, ICN proposes the change from the current *host-centric* network layer to a *data-centric* one, consequently modifying basic elements of communication. Unique *names* identify each available object and are used as main routing information by network mechanisms. A client issues a request for a specific name to the network, which routes the request to a *publisher* and, later, the data back to the client. ICN also proposes the extensive use of *caching* in order to improve content distribution performance.

The focus of ICN is on content-centric based services, but a full-fledged Internet architecture must support every type of application that communicates at a global scale, including those tailored to a connection-centric model. Early work on ICN, such as from Jacobson et al. (2009a), shows the feasibility of porting connection-centric applications to ICN. However, such ports will incur trade-offs with performance and traffic, for example

due to the higher number of messages necessary to implement connection-centric application protocols in ICN architectures. Recent studies suggest that a more plausible reality would be the coexistence of both IP- and ICN-based infrastructures (CHEN et al., 2012).

Albeit the debate over the applicability of ICN, various studies have shown that ICN infrastructures have the potential to improve current content distribution systems (XYLOMENOS et al., 2014). These studies focus on developing the architectural aspects of ICN to improve network performance, for example routing and caching strategies. However, few studies focus on the impact caused by the characteristics of the content distributed in ICN infrastructures. More specifically, the concept of uniquely identified data objects can be used in distinct ways with different types of information, resulting in varying network and application performance.

In this thesis, our goal is to explore the use of multiple objects for the distribution of contents. More specifically, we allow publishers to describe a content as a set of individual objects, by establishing *relations* among them. In a nutshell, a relation is a link between two objects indicating that the data from one complements in some way the data from the other. Following this concept, a content becomes a collection of multiple objects and the respective relations that characterize the interactions among their data. We employ these definitions to formulate a model that publishers can use to distribute contents as sets of related objects. Our study analyzes the implementation aspects of such concepts in an ICN architecture and evaluates their impact on application and network performance.

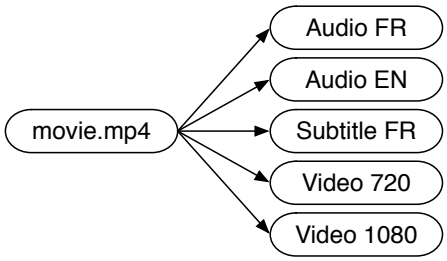
1.1 Problem and Hypothesis

An *object* is the basic unit of data distribution and is uniquely identified according to a naming scheme defined by the ICN architecture (JACOBSON et al., 2009b). The unique *name* of an object is the primary information used by clients to request data from the network. ICN architectures can also further divide an object into multiple *chunks*. This division is agnostic to the content structure and usually considers aspects of the underlying network infrastructure. It is used to facilitate the transmission of data and the implementation of flow control mechanisms. Our study does not focus on chunks; they are considered an implementation aspect of our mechanism when required by the architecture. Rather, our work considers an object as the basic data unit that is uniquely identified in the network.

The naïve method to employ an object to publish a content is to store all of its

data in a single object. In other words, one object encapsulates all the required components a client must obtain to use the content. However, current contents available on the Internet are better described as collections of distinct elements that can be separated and distributed as individual data objects. One example is a multimedia content, such as the one depicted in Figure 1.1. As illustrated, this type of content presents distinct components, such as video channels in different qualities, audio channels in different languages, and subtitles.

Figure 1.1: Video content separated in multiple objects.



Source: Authors (2016).

Some applications on the current Internet employ multiple objects to distribute a single content. HTML is one of the most notable examples. However, these applications employ content description models tailored to their specific requirements. In our work, we are interested in a generic model to describe contents as multiple objects, so any application in an ICN environment can apply it. The widespread use of relations results in two main advantages to the network architecture. First, it allows clients to select only the parts required to reproduce the content according to personal preferences, such as a particular audio language for a movie. As a result, there is a reduction in the traffic needed to distribute the content. Second, parts already available in the network can be reused in the publication of new contents, such as a stylesheet employed in multiple Web pages. As a result, the redundancy from published data is lowered.

A content distributed as multiple objects requires means to identify the needed parts and respective identifiers. Such a *description* of a content must be available through metadata that can be accessed and parsed by any client that desires to retrieve it. In our work, this metadata describes the various *relations* established among individual objects that constitute a content. In the example from Figure 1.1, each object would contain one of the content distinct parts, that is, a video or audio channel, or a subtitle. In turn, each relation establishes how the data from one object acts on data from another to result in the original content. For example, a relation between an object containing an audio channel

and another with a subtitle indicates that the latter contains the transcription of the audio from the first. The type of data contained in each object and the semantic of respective relations will directly depend on the type of content modeled with these concepts.

Current ICN architectures present two main methods that enable the publication of a content relations structure: *hierarchical names* and *link objects*. Hierarchical names are one type of object identifier employed by some ICN architectures (JACOBSON et al., 2009b). They are similar to URLs from current HTTP systems. Each element of a hierarchical name has implicit relations with its parent and child components that enable the description of a content based on multiple objects. For example, the name "</ufrgs/ppgc/video.mp4/video_h264/720p>" allows clients to infer that the "video.mp4" content has a video channel encoded on the h.264 format with a 720p quality. Albeit hierarchical names allow a straightforward way to implement relations, two limitations must be considered. First, all objects that belong to a content must be published with identifiers contained in the corresponding hierarchical structure. Consequently, it is not possible to reuse objects from different contents without adding complexity to the naming scheme. Second, changes in the content structure may reflect on the names of the respective parts. Consequently, object names would be changed after publication, creating instabilities in the network routing mechanisms.

Link objects are a feature implemented in the NetInf architecture (SAIL PROJECT, 2014). In summary, a link object contains a list of names to other data objects that are related in some way. A hierarchical structure of link objects with additional metadata can describe a content based on multiple objects. However, a content with a complex structure will result in a high number of link objects because these can only maintain information about one hierarchy level. Consequently, clients must obtain a large number of metadata objects before requesting the actual content.

Regarding the goal of avoiding data redundancy, Perino, Varvello and Puttaswamy (2012) propose the use of redundancy identification algorithms to scan objects for bitwise identical chunks. These redundant chunks are isolated and published once to avoid data duplication and increase cache performance. Results from the study show potential performance gains from isolating bitwise identical data and publishing it as shared chunks. However, this advantage is limited by the use of an automated redundancy identification algorithm that explores contents from a single publisher. They do not consider the possibility of different publishers distributing contents shared among them. Further, the proposed algorithm only uses bitwise comparison to identify redundancy. It does not con-

sider cases in which the redundancy goes beyond such a comparison, like a video with the same content but different encoding settings.

Based on the previous observations, we conclude that:

1. ICN architectures lack a general model that enables a content to be represented as a complex structure of multiple data objects;

2. There are no studies about the behavior of network and applications when different types of contents are distributed using multiple objects.

Following these conclusions, we formulate the following hypothesis:

If a model to establish a complex set of relations among data objects is available in an ICN, publishers can employ new methods to describe contents as sets of data objects. Such methods would result in better application performance and usage of network resources, because popular objects present a higher request rate and redundancy on published data is lowered.

A *relation model* can be used to implement solutions to known ICN challenges, such as content versioning or collection of network monitoring data as described by Kutscher et al. (2016). Thus, in this thesis, our goal is to present a model to establish relations among objects for ICN and to explore the applicability of this model in relevant scenarios for Information-Centric Networking.

1.2 Goals and Contributions

The study here proposed has three main goals: (*i*) present a model that enables publishers of an ICN to describe a content as a set of data objects linked through arbitrary relations; (*ii*) analyze the design aspects from the implementation of the model in a relevant ICN architecture; and (*iii*) evaluate the impact of relations on applications and ICN infrastructure. We meet these goals, respectively, with three main contributions of our work, described below.

The first contribution is the proposal of a model to establish relations among objects. We design the model to *be application agnostic*, *support different relation structures*, and *be backward-compatible with current ICN specifications*. The model is application agnostic and enables different systems to share a content without converting relations

between different formats. For example, an application would not need to support various specifications of multimedia containers (such as MKV or MP4) to be able to identify an audio channel that belongs to a movie. Applications only need access to a description of relation semantics to interpret the data from each linked object. To enable such a description, the proposed model allows the use of attributes, which are key-value pairs that can be added to relations. In turn, the model is designed to support different relation structures to avoid restricting applications when designing the distribution of data among objects. The examples presented in our work focus on hierarchical structures because these are found in common and relevant applications, such as Web pages and multimedia content. However, other structures can be designed depending on content characteristics, for example, a graph that includes cycles. Finally, the model is designed with backward-compatibility in mind to enable its implementation in current or new ICN architectures without impacting their specifications. As an additional contribution regarding the model, we present a mathematical analysis of possible gains related to the space required to maintain a content catalog. This analysis demonstrates the effects of relations on the elimination of data redundancy from the network.

The second contribution of our work is an analysis of the main design aspects of the model implementation in the Named-Data Networking (NDN) ICN architecture (ZHANG et al., 2010). We analyze the requirements and how to fulfill them with features available in NDN. We focus our analysis on the use of manifests, which are meta-objects that describe the relations among the objects that carry content data. We discuss the trade-offs related to relation storage, organization, management, authenticity, and representation regarding application and network performance.

The third contribution of our work is to evaluate the impact of the relations model on two relevant content distribution scenarios, namely: *multimedia content* and *Web pages*. Each scenario is analyzed in an individual case study. The multimedia case study assesses the behavior of relations with objects that contain a high volume of data. Multiple objects enable the client to obtain only the necessary parts to reproduce the content according to specific quality and regional preferences. The results of this case study show that the use of relations can reduce the download times in an average of 34% and the network bandwidth usage in 43%. In turn, the Web pages case study explores the gains obtained in a scenario that presents a non-negligible overhead caused by the use of relations. In this case study, relations enable the publication of Web pages through composition, which allows the reuse of already existing objects to avoid redundancy. Because

HTML objects have a small size in average, it is possible that the overhead resulting from relations metadata would negatively impact the performance of the network and applications. Consequently, this scenario evaluates in more detail the gains obtained with the use of relations in comparison to the resulting overhead. Our results show that, despite the additional traffic and latency from relations, the download time of clients can be reduced in an average of 28% and the network traffic in 34%.

This document presents the proposed relations model, the analysis of implementation aspects on NDN and the evaluation methodology and results from the case studies. The results from the multimedia case study were thoroughly explored in a paper published at the IFIP/IEEE IM 2015 symposium. In turn, results from the Web pages case study are described in a paper submitted to the Elsevier Computer Networks Journal, which is currently undergoing revision.

1.3 Organization

The remaining of this thesis is divided in 5 chapters:

- **Chapter 2** presents an overview of ICN fundamental concepts and relevant proposed architectures. The chapter also explores the shortcomings of these architectures that are the motivation for the present work.

- **Chatper 3** focuses on the relation model proposed in this thesis. It first presents the fundamental concepts used in the model. Next, it demonstrates how relations can be used to structure different types of contents that may be distributed in an ICN. Lastly, it explores the design aspects to be considered to implement the concept of relations in a mechanism for the NDN architecture.

- **Chapter 4** presents the case study of multimedia content distribution. It describes how the case study is implemented in one ICN architecture, both in a standard implementation and another with the use of relations. Next, it explores the results obtained from a series of experiments that compare content distribution with relations against the results from the standard implementation. The results focus on client performance and network resource usage.

- **Chapter 5** describes the case study of Web content distribution. This case study focuses on the use of relations to improve distribution of HTML content in ICN

based on the concept of micro-caching. The chapter describes the case study concepts, how it is implemented in an ICN architecture, and the proposed evaluation methodology.

- **Chapter 6**, in turn, presents the final considerations about the studies developed in the context of this thesis and directions of future work that may be followed. The chapter also presents additional academic contributions originated from this work.

2 INFORMATION-CENTRIC NETWORKS

This chapter describes the fundamental concepts of information-centric networks. Section 2.1 explores the three essential elements of ICN: object naming, routing, and in-network caching. Section 2.2 presents the three ICN architecture proposals with higher relevance to the research community: NDN, NetInf, and PSIRP. Finally, Section 2.3 describes architectures limitations regarding content publication.

2.1 Fundamental Elements of ICN

ICN architectures employ three basic elements (AHLGREN et al., 2012). The first element is a *naming scheme* that allows unique identification of data objects. The second is a *routing mechanism* based on content names that forwards requests towards content sources and, later, the corresponding data back to clients. Finally, the third element is an *in-network caching mechanism* that enables routers to act as temporary content sources using data received from previous requests. The correct interoperation of these three elements is the key to the efficient content distribution in ICN. Each element is explored in more detail next.

2.1.1 Object Naming

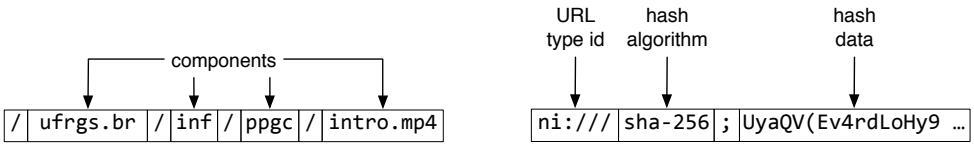
The goal of an ICN naming scheme is to identify with a unique name each content available in the network (BARI et al., 2012). Current IP address schemes focus on identifying hosts because end-to-end connections are the basis for all communication. In contrast, transmissions in ICN are driven by client requests for specific content objects. Thus, the focus of an ICN naming scheme is to identify objects independently of their location in the network. The separation of identifiers and locators is a relevant feature to ICN because it allows any network device that stores content copies to fulfill requests. An ICN naming scheme must also enable a routing mechanism to forward requests and objects efficiently in a global scale. In other words, the algorithms employed in routing mechanisms must be able to use the identifiers generated with the naming scheme to forward requests and data at line-rate speeds on large-scale networks.

ICN architecture proposals explore two main types of naming schemes: *flat* and

hierarchical (XYLOMENOS et al., 2014). Flat names are bit sequences with fixed length. In turn, hierarchical names present a variable number of string components and resemble current Internet URIs. These two schemes diverge in regard to the properties of readability, self-verification, and aggregation.

Human-readable names allow network mechanisms to work directly with identifiers given by users to applications. In turn, non-human-readable names require a DNS-like mechanism to translate user-readable identifiers into actual network names. A priori, a hierarchical name is human-readable while the bit sequence from a flat name is non-human-readable. Hierarchical, human-readable schemes have the additional flexibility of including metadata about the object directly into names, as explored by Sollins (2012). Regarding representation, hierarchical names have a form similar to current URIs, as illustrated in Figure 2.1(a). Flat names are random sequence of bits that do not contain any readable information about the content they identify. To enable their representation in a readable form, Farrell et al. (2013) propose the named identifier (NI), a URL-style form for flat names. Figure 2.1(b) illustrates an example of a name in NI form.

Figure 2.1: ICN naming schemes.



(a) Hierarchical. (b) Flat (NI form).

Source: Authors (2016).

Regarding self-verification, a name generated by a hash function with the necessary security properties can be used for data authentication. In principle, flat names based on hash functions are self-verifiable while human-readable names are not. It is possible, however, to add the output of a secure hash function as an element of a hierarchical name to enable self-verification at the cost of degraded human-readability.

Finally, aggregation can be trivially used to combine hierarchical names by matching common prefixes. It is necessary to enable efficient name-based routing mechanisms for ICN (further discussed in Section 2.2). Flat names cannot be aggregated *a priori*. However, some ICN architectures divide their flat address space in a fixed number of components, enabling aggregation to a certain degree. For example, DONA (KOPONEN et al., 2007) uses a flat namespace divided into two parts: the first identifies the object and the second, the scope that contains the object.

In summary, the choice of a naming scheme influences various characteristics of an ICN architecture, such as self-verification of contents and the possibility to aggregate identifiers. The latter is highly relevant to ICN routing mechanisms, which are further discussed next.

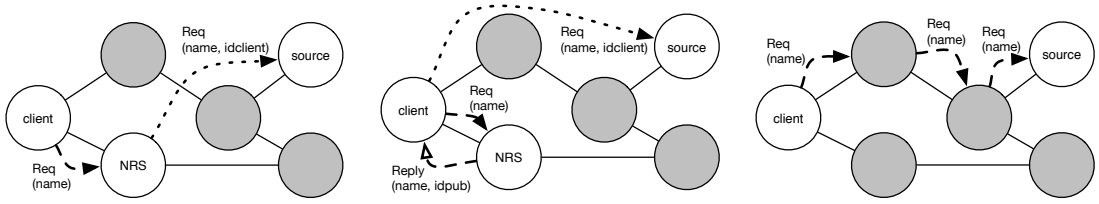
2.1.2 Routing

The routing process in ICN comprises two main steps (AHLGREN et al., 2012). The first, known as *request routing*, encompasses the forwarding of a request from a client to a content source. In turn, the second, known as *content routing*, comprises the forwarding of the content back to the original requester. Each step can employ different approaches depending on architecture design decisions.

Two main techniques are used to implement request routing in ICN: *name resolution service* (NRS) and *name-based routing*. NRS-based routing employs a service similar to the current the DNS service that converts names into lower-level network identifiers (e.g. the IP address of a publisher). A client first forwards its request to the NRS, which will translate the name. Next, the NRS forwards the request directly to the content publisher (Figure 2.2(a)) or returns the translated identifier to the client (Figure 2.2(b)). In the latter case, the client will directly contact the publisher after receiving its network identifier.

A name-based routing scheme forwards requests from clients to sources using only objects names (Figure 2.2(c)). Routers maintain a table that maps names to network interfaces. Each entry of a name-based routing table stores a name prefix. A longest prefix match on the routing table entries identifies the forward interface of a request.

Figure 2.2: Request routing.



(a) NRS direct. (b) NRS indirect. (c) Name based.

Source: Authors (2016).

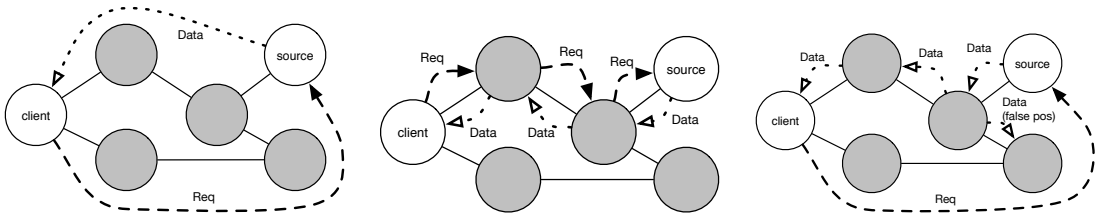
The scalability of name-based routing mechanisms depends on a naming scheme

that allows aggregation. Ghodsi et al. (2011) employ data from Web indexing mechanisms and estimate that a name-based routing table would be roughly three orders of magnitude bigger than the size that guarantees line-rate forwarding speeds for current implementations. Consequently, the naming scheme used for name-based routing must support aggregation to reduce the size of global routing tables and achieve line-speed data forwarding.

The efficiency of NRS-based routing directly depends on the performance of the resolution service available in the architecture. To achieve line-speed forwarding of data, the NRS must be capable of managing an object catalog with a size equivalent to that of the current Web. DHT-based systems for P2P overlay networks are known to achieve good performance when routing flat names over a large-scale network. Thus, they are a possible alternative to implement an NRS service for ICN that does not require name aggregation. Work from Dannewitz, D'Ambrosio and Vercellone (2013) evaluates the scalability of two DHT-based NRSs. They explore results from analytical and simulation experiments with millions of nodes and demonstrate that a global-scale NRS is capable of achieving resolution speeds smaller than 100 ms. However, such results are yet to be validated by experiments with actual implementations in real network environments.

Concerning content routing, three techniques are employed, namely: *direct*, *reverse path* and *Bloom filter*. With direct forwarding (Figure 2.3(a)) the publisher simply sends the requested content back to the client using a lower-level network connection (e.g. TCP/IP). Reverse path forwarding (Figure 2.3(b)) forwards the content back through the same path taken by the request using traces (or "breadcrumbs") left on routers. Finally, in the Bloom filter forwarding (Figure 2.3(c)) the publisher generates a Bloom filter that describes the path that should be used to forward data packets.

Figure 2.3: Data routing.



(a) Direct. (b) Reverse path. (c) Bloom filter.

Source: Authors (2016).

Direct routing uses an underlying network infrastructure (such as IP) to route contents. Consequently, an infrastructure with global internetworking capabilities must be

maintained in parallel with the ICN architecture. In turn, reverse path and Bloom filter routing schemes require only end-to-end links among ICN routers in the path from publishers to clients. Reverse path routing requires routers to maintain state information about unfulfilled requests to identify the forward interfaces of each content. Under high traffic, state information can rapidly consume router resources and degrade forwarding performance (WÄHLISCH; SCHMIDT; VAHLENKAMP, 2013). In their turn, Bloom filters do not require state information in routers because they are kept directly into data objects. However, to generate the Bloom filter, the content source must have access to a service that provides updated network topology information. Such a service might be a complex distributed system when considering a global scale. Bloom filters also have an inherent false-positive chance depending on the hash function used in its implementation. This probability presents a clear trade-off between wasted network bandwidth and computational requirements of routers.

In summary, there are different routing schemes that may be used to implement request and content forwarding in an ICN. No consensus exist regarding the "best" choice because each routing scheme presents particular trade-offs on different scenarios. ICN architecture proposals (further explored in Section 2.2) employ different routing schemes depending on their design choices. The routing scheme also impacts the design of in-network caching, which is described next.

2.1.3 In-network Caching

Caching is one of the fundamental aspects of ICN architectures because it greatly influences the performance of communication among clients. Many ICN studies focus on in-network caching aspects (ZHANG; LI; LIN, 2013; KUROSE, 2014) such as content placement, admission and eviction policies, and analytic models to evaluate caching performance. These aspects are already explored in studies about caching in the current Internet. However, the unique characteristics of ICN in-network caching require a review of current results.

ICN in-network caching presents unique characteristics that distinguish it from current caching systems such as proxies and CDNs. Zhang, Li and Lin (2013) summarize the three main distinguishing features of ICN in-network caching: *transparency, ubiquity,* and *fine-granularity.* ICN caches are transparent in the sense that neither publishers nor applications are aware of their presence in the communication process. In contrast,

currently employed caching systems, such as CDNs, require the implementation of additional mechanisms on the publisher side to properly function over the Internet. On an ICN, caching applies to any transmitted content *a priori*. Thus, publishers do not need to manage additional mechanisms. This characteristic is intrinsic to ICN principles: it does not matter where a client request is fulfilled as long as the content is delivered.

On the one hand, the openness provided by cache transparency implicitly optimizes the communication of any application. On the other hand, the policies used to control caches have a higher complexity. Resource conflicts may arise because different applications that share the infrastructure may have incompatible caching goals. For example, a live-streaming system requires caching of the most recent chunks in detraction from earlier ones, which can be evicted. In turn, a video-on-demand application requires the caching of the earlier content chunks since viewers are prone to watch only the first minutes of the content. As a result, a single caching policy cannot fit the requirements of all applications that communicate over the network. Ultimately, these policies may degrade the overall application performance due to disputes over caching resources (FRICKER et al., 2012). Also regarding performance, caches in ICN are assumed to operate in line-rate. Such assumption requires the development of new technologies to optimize cache operation (PERINO; VARVELLO, 2011).

Regarding cache ubiquity, ICN distinguishes itself because any router can be a caching element. Consequently, ICN infrastructures result in a general network of caches with high content volatility (ZHANG; LI; LIN, 2013). Such characteristics are fundamentally different from those of current systems, that position caching elements in well-known locations and employ complex algorithms to control content placement. Consequently, management techniques from current systems cannot be directly applied to ICN caches.

Finally, ICN caches present a finer granularity in comparison to current systems, which operate maintaining copies of entire objects, for example an image required by a Web page. In contrast, ICN usually divides objects in smaller pieces (named *chunks*) to facilitate the implementation of transmission control mechanisms. In turn, these chunks are the basic unit employed by cache storage in ICN routers. Studies from current caching systems assume the storage of complete objects and do not consider the use of chunking. ICN requires new analytic models for caching because different chunks from an object may present distinct popularities. Also, studies from current systems frequently use the independent reference model (IRM), which states that requests to a cache are probabilistically independent. Results from these studies might not hold in ICN because there are

correlations in requests for different chunks of a single content (ZHANG; LI; LIN, 2013).

An ICN architecture may implement two types of caching depending on its design choices: *off-path* and *on-path* (ZHANG; LI; LIN, 2013). An off-path cache is positioned outside the path between a client and a publisher. It can only serve contents if the routing mechanism is aware of it and forwards requests to it instead of the original publisher. Consequently, off-path caches can become expensive to the routing mechanism if contents stored in caches change too frequently. In turn, on-path caches are restricted to the path between a client and a publisher. An on-path cache does not announce its contents to routers outside the path between the publisher and a client that requested a content. It only fulfills a request if the path from a request intersects with that of a previous request for the same content. On the one hand, on-path caches do not impact the routing mechanism because their contents are not announced. On the other hand, global caching performance can be degraded because close content copies outside the request path are not explored (LEE et al., 2015).

Different architecture proposals found in the literature employ the three fundamental ICN components with particular design choices. Next, we further explore the proposals that received more attention in ICN studies.

2.2 Current ICN Architectures

We now describe how current architectures instantiate the fundamental concepts of ICN. More specifically, we focus on *Content Centric Networking* (CCN) (ZHANG et al., 2010), *Network of Information* (NetInf) (KAUFFMANN; PELTIER; TRUONG, 2013), and *Publish-Subscribe Internet Routing Paradigm* (PSIRP) (FOTIOU; TROSSEN; POLYZOS, 2012). Table 2.1 summarizes the main characteristics of the architectures according to the fundamental elements previously described. Next, each architecture is explored in more detail.

2.2.1 Content Centric Networking

The basic architecture from CCN was first described by Jacobson et al. (2009b). There are two main prototypes of the architecture currently available. The first, known as NFD (name forwarding daemon) (NDN PROJECT, 2014), is maintained by the Named-

Table 2.1: Summary of architecture characteristics.

Architecture	**Naming scheme**	**Request routing**	**Data routing**	**Caching**
CCN	Human-readable, hierarchical	Name-based	Reverse path	On-path
PSIRP	Flat, two components	NRS-based	Bloom filter	Off-path or on-path
NetInf	Flat, two components	Name-based or NRS-based	Reverse path or Bloom filter	Off-path or on-path

Source: Authors (2016).

Data Networking (NDN) Project (ZHANG et al., 2010). The second, known as CCNx (PARC, 2015) is maintained by the Palo Alto Research Center (PARC). Both prototypes follow the main architecture principles with few divergent design decisions. Both NDN and CCNx present prominent contributions to the CCN architecture design.

CCN uses a hierarchical naming scheme with identifiers similar to current Web URLs (e.g. </ufrgs.br/inf/ppgc/intro.mp4>). This hierarchical structure enables aggregation of names with common prefixes for routing purposes. Although ICN proposes the separation of content identifiers and locators, CCN names may present location dependence because of design decisions to improve the scalability of the routing mechanism. More specifically, the aggregation of prefixes in routing tables depends on the distribution of name prefixes in the network topology.

Communication in CCN follows a name-based request routing and a reverse path content routing strategies. It is consumer-driven and employs two main packets, namely *interests* and *data objects*. The former is used to transmit requests from clients while the latter, the data from objects that fulfill a request. Routers maintain two structures to forward these packet types: the *forward information base* (FIB) and the *pending interest table* (PIT). The FIB is a name-based routing table used to forward interests from clients towards content sources. It stores name prefixes announced by publishers connected to the network. In turn, the PIT is used to forward data objects back to clients. It is a list of received interests and corresponding interfaces. Each CCN router also maintains a *content store* (CS) that keeps copies of recently forwarded contents. The CS enables routers to fulfill new interests for previously requested data without the need to propagate the interest to the source. The CCN routing process interacts with these structures as described next.

When a router receives an interest, it will first check if the request can be fulfilled by data from the CS. If not, the router will check if the PIT has an entry for a request

for the same name. If the record exists, the new request and the corresponding interface are merged with the ones already in the PIT. If a PIT record does not exist, it is created, and the interest is forwarded upstream based on a longest prefix match on the FIB entries. If the interest does not match any FIB entry, by default the content is considered non-existent and the interest is dropped. This behavior can be changed depending on network characteristics, for example the router can broadcast the interest to all of its neighbors in an attempt to find a content copy. However, such strategy may create a high network overhead due to interest flooding. When a router receives a data object, it will forward the content to the interfaces listed in the PIT entry with an exactly matching name. The forwarded content is optionally validated and added to the CS. If a PIT entry does not exist the router discards the received data. The router also drops the data if it already exists in the CS.

The forwarding of requests based on longest prefix match poses an engineering challenge because of the expected size of a FIB and the requirement to lookup entries in line speed (VARVELLO; PERINO; ESTEBAN, 2012). Furthermore, prefix aggregation is a crucial element to keep routing table sizes tractable. Without proper levels of aggregation, the size of routing tables would incur in non-line-rate seek times. Due to the above circumstances, the CCNx project is working on a new design for the CCN routing mechanism based on flat names. The flat names used in routing, in this case, are generated with hashes over the original hierarchical object name. Albeit documented by Mosko (2014), this new routing proposal is not yet implemented and actively evaluated.

Regarding the content store, CCN adopts an on-path caching strategy due to the name-based and reverse path routing mechanisms. The CCNx prototype adopts *leave copy everywhere* (LCE) as admission policy and *least recently used* (LRU) as eviction policy by default. This design decision stems from the low implementation complexity of these policies. Nevertheless, as surveyed by Zhang, Li and Lin (2013) work on in-network caching has shown that these policies do not offer the best performance in regards to content distribution. Such work proposes caching policies that try to minimize object redundancy among caches while moving popular contents closer to clients. Although these caching policies offer better content distribution performance, their implementation may not be feasible assuming the requirement of line-rate cache operation (PERINO; VARVELLO, 2011).

Recent work on the CCNx architecture proposes the use of Manifests for the specific purpose of fast chunk authentication through precompiled hashes. This is docu-

mented in a recent ICNRG IRTF draft from Mosko et al. (2015). Precompiled hashes are contained in a manifest together with chunk identifiers. Once the manifest is authenticated through traditional methods (publisher signature), all chunks can be quickly verified with manifest hashes. It is important to note that these manifests are not used for generic purposes, such as to enable the description of complex contents based on a structured set of published objects.

2.2.2 Publish Subscribe Internet Routing Paradigm

PSIRP is a product of two EU Framework 7 projects, the first named after the architecture (PSIRP PROJECT, 2011) and the second known as *Publish Subscribe Internet Technology* (PURSUIT PROJECT, 2013). The PSIRP architecture proposes to replace the current Internet stack with a new one, based on a publish-subscribe model. A working prototype of PSIRP, named Blackadder, is available on the project website.

PSIRP adopts a flat naming scheme with two elements, namely the *scope ID* (sID) and *rendezvous ID* (rID) (TROSSEN; PARISIS, 2012). Both sIDs and rIDs are flat bit sequences of arbitrary size. The rID is the unique identifier of a piece of information. In turn, the sID identifies the scope that maintains a piece of information. The sID can be used to group objects in sets according to an arbitrary information, such as their publisher. A hierarchical structure is used to organize different scopes (e.g. employee scopes belong to a higher level enterprise scope). As a result, an object can belong to multiple scopes and the sID part of the name must list these scopes in hierarchical order. In other words, the name of an object in PSIRP is composed by one or more sIDs and one rID.

The PSIRP routing mechanism is NRS-based and employs the hierarchical structure of scopes (RAJAHALME et al., 2011). The name resolution infrastructure uses a hierarchical DHT that operates on a set of *rendezvous nodes* (RNs). A scope is associated to at least one RN that resolves rIDs of objects that belong to it. A publisher inserts a new object in the network with an announcement to the RN that manages the scope that will contain the object. The RN matches each resolution request to an announced name and forwards the request to the respective publisher.

Content forwarding in PSIRP employs Bloom filters. Consequently, a publisher must know the path to the client to include it in the data object. PSIRP uses special nodes known as *topology managers* (TMs) to manage the required topology information. TMs use a distributed routing protocol to discover *forwarding nodes* (FNs) and links among

them. Each link between two FNs is identified by a bit string generated by hashing. The TM uses these link identifiers to create Bloom filters with the path between two hosts. Paths are encoded on-demand when a TM receives a request from an RN and are sent to publishers of matched contents. The publisher attaches the received path to each object that belongs to the matched content when transmitting it to the network. When an FN receives an object, it applies the Bloom filter to the names of its links and forwards the packet to those with a positive match. The described method only allows a publisher to send data to a client. To enable a two-way channel between the two hosts, the TM sends to the client a *reverse path*. This two-way channel allows clients to send further data requests directly to the publisher, reducing the overhead in the NRS.

The routing strategy employed in PSIRP does not require routers to maintain complex state information to transfer data from publishers to clients. Such complexity is transferred to the size of network packets, which are augmented with routing information for the entire path. In other words, the computational overhead in routers is exchanged by another, in network resource consumption. Another issue with PSIRP routing model is the architecture employed in TMs, which requires unique identification for all links existing in the network. The size of link IDs directly influences the chance of *false positives* occurring in the routing process due to Bloom filters, resulting in wasted network bandwidth. However, longer link IDs result in higher network overhead due to the size of path data attached to content objects. This trade-off must be considered when evaluating a global scale deployment of PSIRP.

Regarding in-network caching, PSIRP is capable of supporting both on- and off-path caching (XYLOMENOS et al., 2012). Off-path caching can be implemented with "mirror" publishers that copy objects from others and announce them to RNs, similarly to the process adopted by a CDN. In turn, on-path caching can be used in each FN to fulfill recurrent requests for popular objects. However, the effectiveness of on-path caches in PSIRP may be reduced because different requests for the same data can result in entirely different paths depending on results from topology managers.

2.2.3 Network of Information

The NetInf architecture was developed in the context of two EU Framework 7 projects: *Architecture and design for the future Internet* (4WARD) (4WARD PROJECT, 2008) and *Scalable and Adaptive Internet Solutions* (SAIL) (SAIL PROJECT, 2013).

Both projects present a wide scope that influences the complexity of specific components of the NetInf architecture, such as routing. A NetInf prototype implementation is available for experimentation (SAIL PROJECT, 2014).

NetInf adopts the flat naming scheme described in (FARRELL et al., 2013). The namespace is divided into two distinct components: an *authority* (A) name and a *local* (L) name. Authority and local names are semantically similar to PSIRP's scopes and resources, respectively. That is, local names can be grouped according to their authority part. NetInf allows both parts of the name to be hashes, enabling self-certification, or an arbitrary string, similar to current URLs (SAIL PROJECT, 2011). For comparison purposes, NetInf names are considered flat, that is, a request is only fulfilled if there is an exact match to a published object name. In turn, depending on the employed routing scheme, NetInf names are interpreted as hierarchical, and routers can employ longest prefix matching.

NetInf allows different routing mechanisms to be "plugged" in the network infrastructure to improve its adaptability to new network technologies. Currently, the architecture specifies two routing mechanisms for its reference implementation, one NRS-based and the other, name-based. The NRS-based approach employs a multi-level DHT similar to PSIRP. A publisher announces its contents to a *local NRS*, which controls the match with requests from clients. A local NRS is considered the *authority* over all local names it stores. In summary, the local NRS creates a Bloom filter that encodes all names it stores and sends it to the global NRS. The latter, in turn, forwards requests to local NRSs based on the authority part of an object name. Clients send requests to their local NRS. If the local NRS cannot resolve the name, it forwards the request to the global NRS. When a content match occurs in an NRS, it sends back to the client the publisher network locator (currently an IP address).

In the name-based routing approach, publishers advertise their objects to *content routers* (CRs), which populate their routing tables based on longest-prefix matching, similarly to CCN. The forward method also follows the same principles of CCN: a client issues a request to a CR, which in turn forwards it hop-by-hop to a publisher or a cache. When a hit occurs, the data is sent back on the reverse path of the request. Differently from CCN, however, routers do not store pending request information in their memory. The reverse path is maintained as hints in the request message and is attached to the returned data.

The two routing methods described above can be combined into a hybrid mechanism with two operation modes. In the first mode, a request is first sent to an NRS, which

will return a *routing hint* for the content publisher. This hint is the network locator of a CR that can receive the client request and forward it to a content source. In the second operation mode, a request is forwarded with name-based routing until it reaches an NRS that can translate it. The NRS, in turn, resolves the request and sends the resulting network locator back to the client.

The fact that NetInf does not specify its routing mechanisms has the advantage of enabling innovation in face of new networking scenarios and environments. However, it may be hard to integrate multiple, possibly non-compatible routing mechanisms that may be employed by different ASs. In such a case, a global standard is necessary for routing among ASs, similarly to BGP in current networks. Regarding the proposed name- and NRS-based routing approaches, the same issues observed in CCN and PSIRP are in effect. Additionally, in the name-based approach, the accumulation of routing information in request messages can incur in high network overhead depending on path and message sizes. Despite that, this name-based approach eliminates the need to maintain state information in routers, reducing their complexity.

The use of in-network caching in NetInf depends directly on the employed routing scheme. The name-based approach enables deployment of on-path caches on every CR, similar to what is proposed by CCN with its content stores. In the NRS-based approach, the project proposes the use of a hierarchical infrastructure of off-path caches, similar to current CDN infrastructures.

2.3 Modeling Contents in an ICN

The ICN architectures above described employ the concept of *data object* (or simply *object*) to define the basic unit of data publication in the network. In general, an object is defined as an individual block of data identified by a unique name, derived from the scheme adopted by the ICN architecture. Current work on ICN assumes that an object carries all data from a content, for example a document with all images associated to it. Additionally, for the purpose of data transmission, ICN architectures establish a subdivision of objects, referred as *chunks*. Each chunk has a small, fixed size (e.g. 4 kB), and the way it is identified in a name varies according to ICN architecture. For example, in CCN, chunks are identified by an additional name segment containing an incremental counter. There is no particular meaning in the way an object is divided into chunks besides creating small units to enable features such as client-side flow control based on chunk request

rate.

The basic concept of object allows a content to be distributed as an individual block of data identified by a unique name. This abstraction resembles the concept of files in operating systems and is sufficient to distribute simple contents. However, a content with a complex structure may require a higher flexibility than that offered by a single, isolated object. Some services on the current Internet already explore such a concept, most notably HTML, that employs multiple objects to create a complete content. Each service that employs multiple objects has a particular method to model and describe the content. Considering the advantages created by the advent of ICN, it is beneficial to evaluate a generic model that enables widespread usage of multiple objects in different types of content. Each ICN architecture presents additional features that can offer such a flexibility. We explore these features below.

The hierarchical naming scheme available in NDN can be used to establish relations among objects. Objects that present a certain prefix within the namespace can be considered to be related in some way. The segments of the name are used to identify how each object relates to another with the same prefix. For example, as proposed by Kulinski and Burke (2012), the components of a video "</ufrgs/inf/intro.mp4>" may be aggregated under a single prefix that represents the entire content. Objects including such a name as prefix would be components of this content. For example, the name "</ufrgs/inf/intro.mp4/h264-1080p>" would represent a simple video channel with a 1080p quality from the above content. This strategy can be used if the content presents a hierarchical structure among its components. However, if the content components are related in a non-hierarchical form, additional mechanisms, besides those enabled by the naming scheme, must be used to represent the structure. The length of names can also become a concern. If segments are used to characterize how each object is related to the content their length will cause excessive processing overhead in network components.

The object model from the NetInf architecture can also be used to establish links among related objects. NetInf defines an object type named *information object* (IO). In summary, an IO is an object that contains one or more links to other objects, which can be other IOs or data objects (DOs). The concept of IOs in NetInf enables modeling of complex structures among objects, such as an artist discography, composed of multiple albums and songs. IOs can be used to model contents with diverse link structures. However, the IO model presents design decisions that can result in high network overhead to retrieve the content structure. More specifically, a content that presents a structure with

multiple levels of relations requires a recursive retrieval of multiple IOs, resulting in a high latency and network overhead.

The naming structure adopted by PSIRP enables objects to be related according to the structure of scopes in the network. In turn, scopes allow objects to be organized according to publisher entities, not related contents. The scope structure could be used to specify an entity that represents a content, encompassing all required objects. However, such objects would have to be distributed by the same publisher, limiting the flexibility of relations.

The architectures described above do not provide a standard and flexible method for a publisher to explicitly identify that the data in different objects is related in some way. A common format to express *relations* among objects would enable publishers to model contents in ways that could potentially benefit the network and clients in terms of performance. For example, the work from Wang, Krishnamurthy and Wetherall (2014) (when applied in the context of ICN) would benefit from a common standard to describe a content as multiple objects linked through relations. Authors only evaluate the potential gains of decomposing and micro-caching HTML data. They argue that current Web systems require modifications to support such proposal. In turn, an ICN architecture that supports the use of relations would natively support micro-caching and decomposition, thus achieving the gains demonstrated by authors.

A thorough discussion about relevant design choices is needed to implement a relation mechanism for ICN that does not create unnecessary overheads in the network architecture. This discussion is the focus of the next chapter.

3 A MODEL FOR RELATIONS AMONG OBJECTS

This chapter introduces a model that enables publishers to describe relations among objects in an ICN and the respective implementation, henceforth called NDNrel. Section 3.1 presents the fundamental concepts employed in the model. Next, Section 3.2 describes several usage scenarios that illustrate how to use the model in current and future applications implemented with ICN. Section 3.3 presents a mathematical formulation that measures possible network gains achievable with relations. Finally, Section 3.4 explores relevant design aspects to implement NDNrel in current ICN architectures.

3.1 Fundamental Concepts

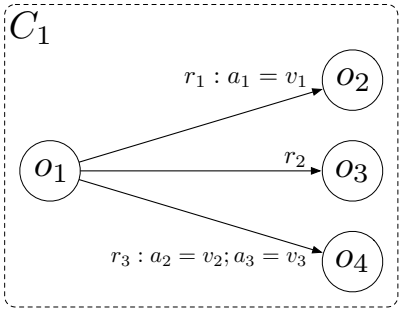
The proposed NDNrel model employs three fundamental elements: object, relation, and content. An *object* represents an individual piece of information that is uniquely identified in the network. The model also assumes that objects are the basic data element that clients can request to the network. This concept of object should not be mistaken with the concept of chunk, employed by some architectures to divide objects in smaller pieces to facilitate the data transmission over the network. This representation is equivalent to the one adopted by current ICN architectures (AHLGREN et al., 2012) and guarantees compatibility with applications that opt not to use the proposed model. In turn, the characteristics of the data contained in objects will vary according to the application that publishes contents and how it employs the relations model.

A *relation* from an object a to another, b, is a link indicating that the data of a can be complemented by the data of b in some way. Each relation has a semantic that is directly related to the data contained in the linked objects. The semantic depends on the way a specific type of content is modeled using relations. Nevertheless, the NDNrel model enables the semantic of a relation to be explicitly presented to applications using it. To that end, each relation can be associated with one or more *attributes*, which are key-value pairs with additional information to describe it. Applications can use attributes to further analyze a relation and take different actions depending on the content. For example, a publisher can use attributes to identify objets that are optional for the reproduction of a content. These objects can be ignored by clients that are not interested in the specific data marked as optional by publishers.

Finally, a *content* represents some information that a publisher wishes to make

available to clients in the network. Currently, ICN architectures adopt a direct relation when mapping contents to objects. That is, an object o encapsulates all the information required by an individual content C. With relations, the content C can now be represented as a tuple $\langle O, R \rangle$. O is the set of objects that carry the content data. In turn, R is the set of relations that describe how to use the data from objects in O to reconstruct the content C. The described concepts are illustrated in Figure 3.1. In the example, the content C_1 is composed by the object set $\{o_1, o_2, o_3, o_4\}$, which are linked by the relation set $\{r_1 : o_1 \rightarrow o_2, r_2 : o_1 \rightarrow o_3, r_3 : o_1 \rightarrow o_4\}$. Some relations also present attributes that characterize the interaction of the linked objects data. More specifically, r_1 has the attribute $a_1 = v_1$ and r_3 has the attributes $a_2 = v_2$ and $a_3 = v_3$.

Figure 3.1: Illustration of fundamental concepts.



Source: Authors (2016).

ICN architectures do not make assumptions about the semantic of contents stored in objects. In other words, routing and caching mechanisms only see an object as an arbitrary block of data. We employ the same assumptions regarding relations. That is, a priori, the semantic of relations is opaque to the ICN infrastructure and is known by the applications that publish and request the content. We maintain this design decision to keep the network core simple. Semantic inferences over relations and similar tasks are left to the application level, allowing the network to focus on storage and transmission. The above concepts can be used in a myriad of ways to model contents published in an ICN. The next section demonstrates the flexibility of NDNrel by exemplifying different features that can be implemented in ICN with it.

3.2 Usage Examples

Recent discussions from the ICN research community suggest that ICN architectures require additional mechanisms for applications to implement advanced features such

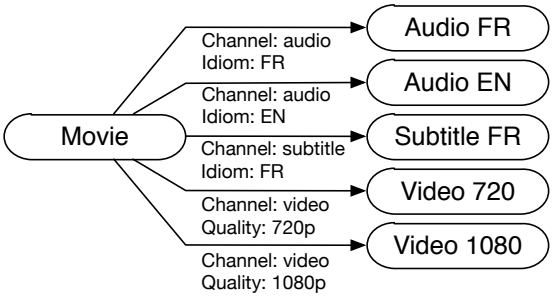
as: content personalization and internationalization, aggregation of related content data, and data control for versioned contents. To demonstrate the flexibility of the proposed relations model, we focus on three particular usages for relations that enable the implementation of the features mentioned above. These are: (*i*) *decomposition:* publish a content as multiple unique objects; (*ii*) *composition:* share previously published objects to create a new content; and (*iii*) *versioning:* modify an object data without creating an entirely new object copy. We present examples that demonstrate these modelings and the advantages of using NDNrel, namely: a multimedia content, a Web page, and a cumulative log.

3.2.1 Decomposition

Decomposition enables a publisher to divide contents into multiple parts, which are individually made available in the network. More specifically, the process of decomposition assumes that a publisher wishes to distribute a content contained in a mass of data that may be separated into distinct parts. Each part is published as an individual object and relations are used to describe how the data in objects should be used to reconstruct the content. The use of decomposition allows redundant parts that may exist in the original data to be eliminated, reducing the required storage space and network traffic needed for distribution. Further, decomposition allows clients to obtain only specific content parts, also reducing network traffic and download times.

Figure 3.2 illustrates an example of a content decomposed using relations. In the example, the multimedia content has a set of options regarding video quality (720p or 1080p), audio language (French or English), and subtitles (French). Relations allow the data of each option to be published as an individual object related to the main content "Movie". The attributes of each relation indicate how the data of each object relates to the

Figure 3.2: Multimedia modeling example.



Source: Authors (2016).

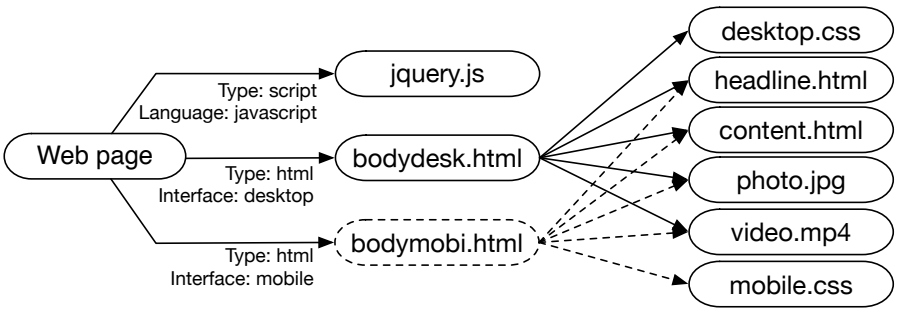
main content and the option it represents. A client can access the relations of the “Movie” object to identify the available audio, video and subtitle options and then download only the objects of interest. Consequently, the client only consumes the network resources necessary to obtain the requested content options instead of the entire content data.

3.2.2 Composition

Composition enables the creation of new contents based on the information of already published objects. As such, the basic assumption for the process of composition is that a set of objects is available before the creation of contents. The publisher employs relations to create a new content based on the data contained in a subset of objects already available in the network. The publisher may also include newly created objects with additional data of the composed content. The main advantage of composition lies in the lower data redundancy achieved with the reuse of previously published data. Such reuse can also result in higher caching performance depending on the popularity of objects used in composed contents.

Figure 3.3 illustrates the reuse of objects in the context of Web pages. The example depicts an object that contains the desktop version of a page (`bodydesk.html`). In turn, this object is related to other components required to the page, such as stylesheets, images, and other HTML data. The publisher wishes to add a new version of the page specifically for mobile devices. Instead of creating a new object with all page components, the publisher simply creates objects for the specific elements of the mobile version and reuse the objects already available from the desktop version. In the example, the publisher adds the `bodymobi.html` object, which contains the specific information of the mobile version while linking previously existing objects, such as images and additional HTML data. The

Figure 3.3: Web page modeling example.



Source: Authors (2016).

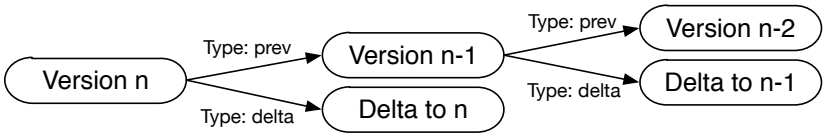
publisher also adds a stylesheet specific to the mobile version (`mobile.css`) and ignores the one from the desktop version. Both page versions can be accessed through the web page relation structure, from which the client can identify and obtain the objects specific to the desired version. As previously mentioned, the objects common to both versions are also benefited from improved caching potential since they will receive requests that would be directed to two different copies of the same data.

3.2.3 Versioning

Versioning enables the data of a previously published content to be updated without breaking the ICN principle of unique naming. The process of versioning assumes that a content is already published using relations (e.g. using decomposition) and must be modified (e.g. adding a new audio channel to a multimedia content). Thus, the process of versioning involves the publication of an updated version of a content relation set, which describes the state of the content after the update. It is important to note that the process of versioning is different from that of composition because the latter assumes the creation of an entirely new piece of content while the first assumes an already existing content is modified. Further, the process of versioning assumes that the new version may exclude objects from a relation structure while composition only assumes the possibility of inclusion. The naming of a content with multiple versions is a known problem in ICN (KUTSCHER et al., 2016) with some solutions available and under study (e.g. the use of name component in a hierarchical scheme to identify object versions).

Figure 3.4 illustrates the use of relations to enable the versioning of a log file distributed in an ICN. The log is updated with new entries according to a division criteria (e.g. all entries from a particular hour). Each version n has two relations. The first relation points to the previous version ($n - 1$). In turn, the second relation points to the object containing the new entries, that is, the delta from the previous version (Δn). Redundancy is avoided with this simple model because only the changes from previous versions are

Figure 3.4: Log file modeling example.



Source: Authors (2016).

added with each update. Furthermore, previous versions of the content are still accessible to applications. It is interesting to note that this simple model of versioned content can benefit from the use of techniques employed in modern content versioning systems. For example, the *delta-skip* algorithm employed in Subversion (ASF, 2014), which reduces the chain of delta objects that must be processed for the reconstruction of a content with a large number of published versions.

The implementation of the above features using relations provides performance advantages to content distribution in ICN. In the next section we provide a straightforward mathematical analysis of such advantage.

3.3 Analytical Evaluation

The use of relations adds flexibility to content publication and enables the creation of structures based on the combination of data from any object available in the network. Also, the use of relations lowers the data redundancy in published contents, positively impacting the use of network resources. Such a reduction occurs because data that would be published multiple times becomes available in a single object accessed by contents. However, a complex relation structure can also result in performance issues, such as high network overhead to retrieve relation information. Thus, the use of relations is justifiable if the performance gains in content distribution surpass the additional overhead generated by the model. More specifically, we are interested in the space required to maintain the catalog because it reflects the amount of redundancy that is eliminated from data in the network. To assess this impact, we elaborate a straightforward mathematical model, which is described next.

Assume a content catalog C with an arbitrary number of contents $|C|$. Each content $c \in C$ presents *variants*, which are modifications of c that present at least one block of *shared* data among themselves. Each variant also presents at least one *individual* block of data that is unique and not present in the remaining variants. We identify the set of variants of a content $c \in C$ as V_c, a specific variant as $v \in V_c$, and the number of variants of c as $|V_c|$. The shared part of variants in a content c has size S_{s_c} and the individual part of a variant $v \in V_c$, size S_v. We consider that, without the use of relations, the shared part of a content is replicated by each of its variants. Thus, the total size of the catalog without the use of relations S_{Cd} is given by Equation 3.1:

$$S_{Cd} = \sum_{c=1}^{|C|} \sum_{v=1}^{|V_c|} (S_{s_c} + S_v) \tag{3.1}$$

When relations are used, we consider that the shared part of the content is separated from the variants and distributed as an individual object through relations. In this case, we must also consider the size employed to describe the relations of content c, which is given by S_{r_c}. In this case, the size of the content catalog when relations are used is given by Equation 3.2.

$$S_{Cr} = \sum_{c=1}^{|C|} \left(S_{s_c} + S_{r_c} + \sum_{v=1}^{|V_c|} S_v \right) \tag{3.2}$$

The manipulation of these two equations enables us to define a formula for the expected reduction in the space required to store the content catalog. We define this reduction as the ratio ρ_C between the space of the catalog when relations are used and the catalog without relations. This is represented by Equation 3.3:

$$\rho_C = \frac{\sum_{c=1}^{|C|} S_{s_c} + S_{r_c}}{\sum_{c=1}^{|C|} |V_c| \times S_{s_c}} \tag{3.3}$$

Equation 3.3 shows that *the use of relations is advantageous if the size required to describe relations is smaller than the space required by redundant copies in content variants*, that is, $\sum_{c=1}^{|C|} S_{r_c} \leq \sum_{c=1}^{|C|} S_{s_c}(|V_c| - 1)$. This relation is true in cases such as the multimedia content illustrated in Figure 3.2, in which the size of relation information is greatly surpassed by the size of video and audio data. In this case, the overhead added by relations is minimal in comparison to the potential of traffic and retrieval time reduction. In turn, a content with data of small size and relations of higher complexity, such as the Web page illustrated in Figure 3.3, may present a non-negligible overhead because the space required to represent relations may surpass the size of objects data.

To further evaluate the performance impact of relations on content distribution we elaborate two case studies based on different content scenarios. We evaluate these scenarios both in a default case, without the use of the proposed model, and another in which relations are used to distribute contents through object composition. To elaborate these scenarios, the proposed model must be implemented as a mechanism in an ICN architecture. We choose a currently developed and scientifically relevant architecture, namely NDN, to implement such a mechanism. In the next section we explore the design aspects to be considered in an implementation of the relations model in the NDN architecture. In

turn, the case studies and their evaluation are discussed in the next chapters.

3.4 Design Aspects

The model presented in the previous section can be used to publish contents if it is available as a mechanism in ICN architectures. In this section, we detail the design of NDNrel, a relations mechanism tailored for NDN. Our goal is to implement the relations model using the NDN architecture concepts and features. We identified five main design aspects of the mechanism:

- **Storage:** What type of object is used to store information about relations.
- **Organization:** How can a publisher organize the relation structure to describe the content.
- **Management:** How clients can create, modify and remove relation information from the network.
- **Authenticity:** What methods are used to verify that relation information and if the linked objects are authentic regarding the content.
- **Representation:** What format is used to represent the relation structure and data objects of a content.

These aspects are explored in depth next. We present the main possibilities for each of them and, in the end, a description of the design decision assumed for NDNrel. As we progress through the aspects, we consider our previous design decisions to narrow the issues explored in our analysis.

3.4.1 Relations Storage

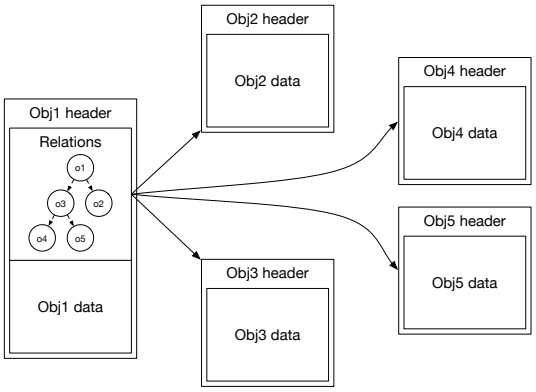
The first aspect to consider is what type of object the mechanism uses to store relation information. The characteristics of the object used to store relations should not limit the flexibility of the mechanism. We consider two main options: *internally* or *externally* to objects with content data. In the first case, relation information is kept in the same objects that store the content data. In the second, information is stored externally to

the content object. Next, we detail the storage options, discussing their advantages and limitations.

3.4.1.1 Internally

The first option is the most trivial and adds relations into the objects with the content data. Figure 3.5 illustrates it with an example of five objects with embedded relations. These objects can have up to three components: header, relation structure (optional), and content data. The object header must identify the chunks that contain relation information, enabling the clients to retrieve them and parse the relations before requesting the actual content data. The relation structure (when available) describes the content organization whereas the data contains the actual content.

Figure 3.5: Relations stored within the data object.



Source: Authors (2016).

This storage option has a limitation when updating the relation structure. Since ICN principles forbid updates on existing objects, the entire data part would be replicated to edit the relations of an object. Although a versioning control could alleviate the update overhead, the content data would still be replicated in multiple objects differentiated only by their relation structure.

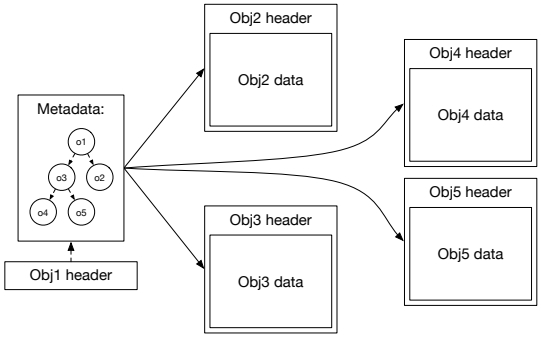
3.4.1.2 Externally

The second option is to store the relation information separately from content data. It reduces the overhead to manage relations because there is no need to replicate the content object when an update of the relation structure is done. Two alternatives exist to implement this option: metadata or manifest objects. We discuss each one next.

Metadata is a piece of information that provides additional information about a

published object and can be retrieved independently of the content data. Figure 3.6 exemplifies the use of metadata to store the relations of the virtual part “Obj1”. A virtual part does not contain content data (e.g. the Movie object presented in Section 3.2.1) and results in an object with a header and associated metadata, but no payload. NDN offers a metadata feature that enables publishers to add metadata to objects. These can be accessed using the reserved name component %C1.META and a metadata identifier.

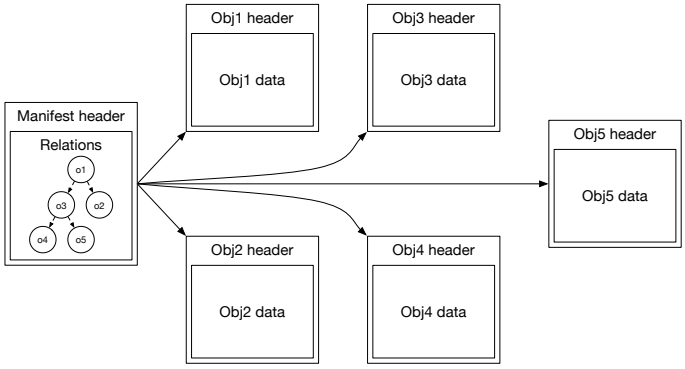
Figure 3.6: Relations stored using metadata.



Source: Authors (2016).

Manifests, in turn, are separate and self-contained objects that maintain the relation structure of a given content. Figure 3.7 demonstrates the relation structure of the content stored as a manifest. This approach is simpler and more flexible than metadata because it does not need to be associated with an existing object.

Figure 3.7: Relations stored using manifests.



Source: Authors (2016).

This storage option could be implemented using the CCNx manifest specification (MOSKO et al., 2015), which is a result of an ongoing work of exploring manifests to improve content distribution. The specification defines a simple structure that describes a collection of named objects. However, CCNx manifests fail to satisfy requirements of the relations model (presented in Section 3.1), as we will explain later in Section 3.4.5.

3.4.1.3 NDNrel design choice

We choose to store relations using manifests because it is the most flexible option. Storing relations directly into data objects would limit the mechanism ability to manage the relations due to the update overhead. In turn, the use of metadata requires every information to be associated with a published object. This restriction reduces the flexibility of the mechanism, particularly when using "virtual" content elements. Next, we address the retrieval methods available to recover relations stored as manifests.

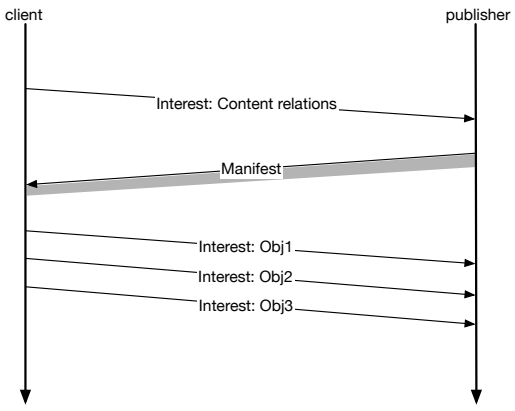
3.4.2 Relations Organization

The second aspect regards how the publisher organizes the relation structure of content: *flat* or *hierarchical*. The publisher should consider the trade-off between size and complexity of the relation structure to organize the contents. Next, both approaches are discussed.

3.4.2.1 Flat approach

The flat approach centralizes the complete relation information of content in a single manifest. This characteristic allows the clients to find the entire content structure in one location and retrieve it with a single request, as shown in Figure 3.8. On the negative side, this approach does not provide modularity for sub-parts of the content, which prevents their reusability to define other contents. In general, the flat approach is better suited for contents that have simpler descriptions or that need low retrieval delay.

Figure 3.8: Relation retrieval using flat organization.

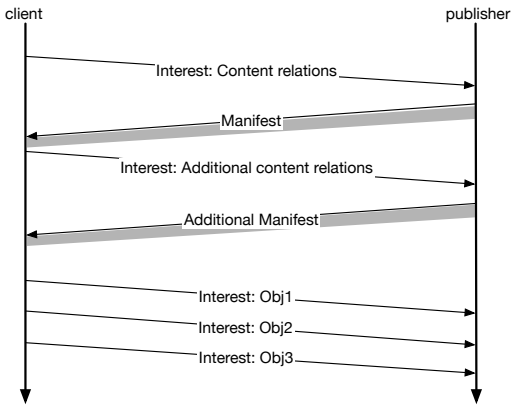


Source: Authors (2016).

3.4.2.2 Hierarchical approach

The hierarchical approach allows the publisher to divide the relation information into multiple manifests. Sub-parts of content are structured as modules, and they can be reused to compose other contents, simplifying the description of complex contents. Figure 3.9 illustrates the retrieval process, in which clients can access specific relations. However, they need to issue multiple requests to obtain the entire content structure since the relations are spread in different manifests. This approach is recommended for contents that are complex or popular because they benefit more from the modularity and reusability of manifests.

Figure 3.9: Relation retrieval using hierarchical organization.



Source: Authors (2016).

3.4.2.3 NDNrel design choice

The use of the approaches described above depends directly on the type of content to be described with relations. Thus, we design NDNrel to support both approaches, increasing the flexibility of the mechanism. Publishers should use their knowledge of the content to organize the relations in the best way. Next, we explore the alternatives to implement operations to manage relations of contents.

3.4.3 Relations Management

The management aspect refers to *create*, *modify*, and *remove* relations of contents. Relation mechanisms have different capabilities according to operations they support. In NDN, the creation of manifests is natively supported, whereas the other two operations

are not and may require additional mechanisms. We discuss the details of each operation next.

Creating a relation is the simplest operation, requiring only the publication of a set of manifests that describes the content structure. A publisher can use any object available in the network (including those from third-party publishers) in new contents because it does not affect the data object. It is supported natively by NDN and does not require any extension.

To modify a published manifest, the publisher needs to create a new one with the updated information and mark the previous manifest as obsolete. Both actions are required to preserve the unique identification of objects in ICN. NDN supports a simple version control mechanism by adding a name segment containing an incremental counter to the object (MOSKO, 2015).

The third and last operation can be achieved using two methods. The first one is relying on the modification operation. In the extreme case that all relations are removed, an empty manifest would be published. The second method is to delete the manifest object from the network, which is still an open challenge in ICN (XYLOMENOS et al., 2014). One of the biggest concerns is how to eliminate copies of the object located in clients (other from the publisher) and caches.

3.4.3.1 NDNrel design choice

To improve the capabilities of NDNrel, we include all three operations. The relation modification is enabled through the versioning segment of the object name whereas its removal is done using the modification operation. Next, we consider issues related to the authenticity of relations.

3.4.4 Relations Authenticity

The fourth aspect is the authenticity of the relation structure created by the publisher. The mechanism needs to guarantee that the *relation structure* (manifest) and the *linked objects* retrieved by the clients are those specified by the publisher.

Clients can authenticate a relation structure by verifying the authenticity of the manifest objects that describe it. NDN uses a standard public key signature mechanism to support object authenticity, in which every object is signed with the private key of its

publisher. The signature information (algorithm, public key, and key locator) are sent with the manifest object to the clients, enabling them to verify its authenticity.

The objects listed in a manifest benefit from the same NDN mechanism but require an extra step to guarantee that their data is what should be used in the content. The publisher can add the signature information of related objects in the manifests (as relation attributes) (MOISEENKO, 2014). This information allows clients to verify the retrieved objects, including those from third-party publishers.

3.4.4.1 NDNrel design choice

The NDN mechanisms are sufficient to guarantee the authenticity in NDNrel. The manifest (and relation structure) are authenticated without any modification whereas the linked objects have an extra step. We add the signature information of those objects as attributes in the manifests, enabling clients to verify their authenticity. Next, we elaborate some considerations about the description format employed in NDNrel manifests.

3.4.5 Relations Representation

The fifth and last aspect of the mechanism design is the format to represent the relations. It should support the model features (presented in Section 3.1): (*i*) a hierarchical structure that represents complex contents with multiple levels of relations; and (*ii*) the attributes included in the description of relations. We consider two possible formats: the *CCNx manifest specification* (MOSKO et al., 2015) and the *structured description formats* (e.g. JSON or XML).

3.4.5.1 CCNx manifest specification

The CCNx manifest enables the description of an object collection formatted as a list of names and their corresponding signature value (MOSKO et al., 2015). The specification fails to satisfy the requirements of our proposed model for two reasons. First, the manifest does not support the inclusion of attributes about relations (except its signature value). In other words, publishers can not give any semantics to the relations between the objects as required by our relations model, limiting the description richness of the mechanism. Second, the CCNx manifest uses a simple list of objects to define the content, forcing publishers to use multiple manifests when describing hierarchical (and complex)

contents. Not only the specification prevents the use of a single manifest (flat approach) for hierarchical contents, it also restricts the organizational flexibility of relations by requiring a manifest for each level. These limitations reduce the richness and flexibility capabilities to describe contents, narrowing the type of content that the mechanism can represent, and could add significant overhead due to poorly structured content.

3.4.5.2 Structured description formats

JSON and XML are current standards applied for various uses, especially in Web applications. Both formats satisfy the model requirements by enabling the description of a hierarchical structure of elements and the association of attributes to these elements. The combination of these properties grants flexibility for the mechanism to represent different types of contents.

3.4.5.3 NDNrel design choice

Due to the limitations identified in the CCNx manifest specification, the implementation of NDNrel encodes the manifests with a structured description format. Specifically, JSON was chosen over XML because it is less verbose, reducing the mechanism overhead. We may change this decision in the future if the CCNx manifest specification evolves to a design that supports the proposed relations model.

In the following Chapters, we evaluate the concepts and design decisions of NDNrel to model and distribute content using relations. We develop two case studies considering relevant applications on the Internet: multimedia and HTML content. In the next two chapters, we present the methodology and results from each of these case studies.

4 CASE STUDY: MULTIMEDIA CONTENT

Multimedia content distribution is a relevant application on the current Internet. The network traffic generated from such applications is constantly growing (CISCO, 2016) and is estimated to encompass from 80% to 90% of the total network traffic in 2018. This tendency shows the importance of understanding and optimizing the methods employed to distribute multimedia content.

Current multimedia distribution services, such as YouTube, employ Dynamic Adaptive Streaming over HTTP (DASH) (ISO/IEC, 2014) for data transmission. Contrary to previous findings related to video distribution over computer networks, HTTP has proven to be an efficient protocol to transmit multimedia content (LEDERER et al., 2016). The efficiency of HTTP media streaming depends on dedicated mechanisms such as CDNs optimized for the distribution of this type of content.

Information Centric Networking has been evaluated as a potential tool for multimedia content transmission since its conception (KULINSKI; BURKE, 2012). This study shows that the ICN-based video distribution can surpass the performance of DASH-based infrastructure from the current Internet in terms of network traffic and quality of the delivered content. To achieve such a performance, ICN-based multimedia distribution requires a mechanism that enables functionalities similar to those available in DASH-based systems, such as the distribution of contents with multiple quality settings. Relations enable such a functionality because they can be used to decompose the multimedia content into multiple objects, each containing a specific content option. As explained in Section 3.2.1, the division enables clients to obtain only the necessary parts to reproduce the content according to specific quality and regional preferences. Consequently, the client download times and network traffic required to distribute the content are reduced because of ICN in-network caching.

This section explores the case study developed to evaluate the potential advantages of using decomposition through relations to distribute multimedia content in ICN. It first presents an overview of multimedia content distribution over ICN based on previous literature. Next, it explores the use of decomposition based on relations to model the multimedia content to be distributed. In turn, the final part of the section presents the parameters and metrics employed to evaluate the network and application performance in content distribution.

4.1 Multimedia Content over ICN

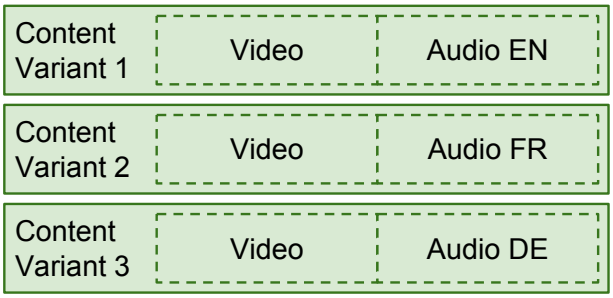
The work from Kulinski and Burke (2012) employs multimedia content distribution as a case study for ICN. The authors overview possible methods to enable the distribution of video over NDN while taking advantage of the architecture mechanisms. They focus mainly on the use of NDN naming scheme capabilities to address different chunks of a multimedia content in order to enable random-access streaming for video-on-demand (VoD) and live streaming. The authors also show that the use of naming to enable the distribution of multimedia content decomposed in multiple parts could achieve performance gains due to: the in-network caching mechanism; and the existence of multiple network paths to content sources. However, they do not further explore such an idea.

A multimedia content may comprise different video and audio channels, each one with varying characteristics such as quality or idiom. Modern multimedia container formats, such as MP4 (ISO/IEC, 2015), support the distribution of multimedia content with different audio, video and subtitle channels. When such a content is reproduced, one option of each channel is selected according to the user preferences. Albeit modern formats support multiple channels, it is not hard to find examples on the current Internet of multimedia contents comprised of multiple files, each one for specific audio variants from the same video. Consequently, redundancy exists in multimedia content, especially for data from different providers that distribute copies of the same movie.

The baseline scenario of this case study, which is illustrated in Figure 4.1, is based on the above discussion. More specifically, we consider that a content publisher creates a movie with different variants, each one characterized by an audio channel of a specific idiom. The publisher proceeds to distribute each variant as a monolithic file containing the video channel and one of the audio channels. Clients interested in a specific variant discover its name and request the respective data object to the network. The client requests subsequent chunks of the content, which contain both audio and video information with the goal of obtaining the content as fast as possible. This behavior is compatible with applications for offline viewing and VoD, as observed in studies from YouTube (CHA et al., 2007).

It is important to note that, in the current Internet, a client interested in a specific variant of a content first has to discover its location through a search engine or a multimedia service interface. Content distribution based on relations also requires some sort of discovery mechanism because clients need a name to request the relations metadata prior

Figure 4.1: Multimedia content baseline scenario.



Source: Authors (2016).

to request the content itself, independently of the method employed to store relations. A mechanism or service that allows clients to search and discover content names in an ICN is an open research issue. Thus, in this case study we assume that such mechanism is available and clients know the name of contents to be requested *a priori*. We leave the issue of content names discovery with the use of relations as future work. We also assume the content is divided in individually addressed chunks and that the client will request them in order. The request rate of each client will vary according to its network conditions and type of application (e.g. live streaming, VoD, or offline viewing).

The process to distribute multimedia over ICN described above assumes that each variant of a content is encapsulated in a monolithic object containing one audio and one video. Next, we explore the use of relations and decomposition to publish a content with multiple variants as a set of objects that carry individual parts. The use of relations eliminates the need to publish each variant as an individual object, lowering network data redundancy.

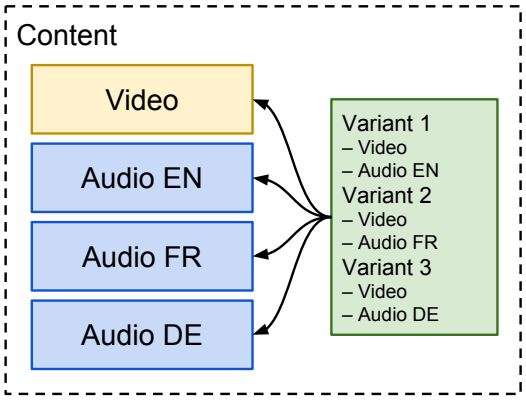
4.2 Using Relations to Distribute Multimedia Content

In this case study we assume the publisher intends to use relations to decompose the movie that will be distributed. That is, instead of distributing each variant as a monolithic object, each audio and video channel becomes available as a single object. These objects are linked through relations that describe to the client the available options. We assume, as previously discussed, that the multimedia content employs a container format that supports multiple audio and video channels. Further, we assume that the container format allows the client to reproduce the content when at least one specific audio and video channel options are downloaded. Figure 4.2 illustrates the content structure after

decomposition.

The client first obtains the relation metadata that describes the movie. As stated earlier, we assume the client employs a mechanism such as a multimedia service interface to discover the object name of the movie, which will point to the content relation structure. After analyzing the relations and selecting the desired audio and video options, the client proceeds to request the objects in parallel to enable immediate content playback as in VoD systems.

Figure 4.2: Multimedia content scenario with relations.



Source: Authors (2016).

The main advantage of employing decomposition in this case study is the lower redundancy in the content data compared to the use of monolithic objects, because video channels are not duplicated in each variant of a movie. Further, a client only needs to download the audio and video of the respective idiom choice, ignoring the remaining data that would be unused and would become a waste of network resources. The lower redundancy also leads to better cache performance, consequently reducing the download time of clients due to the smaller distance to the content. To verify these claims, we propose to experimentally evaluate this case study through simulation, as described next.

4.3 Evaluation Methodology

This section describes the methodology employed to model our case study and its evaluation. The basic scenario is the distribution of multimedia content over NDN. We employ relations as foundation for content decomposition, with the goal of eliminating the data redundancy in published objects. To guide our analysis we define two main research questions to be answered by our experiments:

Q1: How much network user performance is improved with relation-based decomposition? The goal is to evaluate the improvement in clients experienced quality and publishers' request load.

Q2: How much more efficiently network resources are used when relation-based decomposition is employed? The goal is to understand the benefits on the network, specially in the overall traffic and cache utilization.

We implemented a relations mechanism based on the design decisions described in Chapter 3 over the current publicly available version of CCNx (0.8.2 at the time of this writing). However, to achieve a higher scale in our experiments, we decided to use a simulation environment to conduct our analysis. In the remainder of this section, we describe the characteristics of the scenario employed in our experiments.

4.3.1 Simulation environment

We extended the well-known ndnSIM simulator (AFANASYEV; MOISEENKO; ZHANG, 2012) to include support for relations and decomposition. We use the simulator to evaluate scenarios where a multimedia content is modeled as a set of decomposed objects, which represent a video and different audio channels for it. We name this simulation scenario **NDNrel**. We also employ an unmodified version of the simulator as a baseline scenario, in which each content variation (choice of audio and video channel) is published as a unique object. This baseline scenario is named **default NDN**. The results presented in Section 4.4 focus on the performance difference between NDNrel and default NDN. Additionally, we explore the impact of different content popularity distributions by using two different parameter values, which will be discussed later. So, there are four different scenarios to be evaluated in the experiments.

4.3.2 Content and workload

The requested multimedia content follows the characteristics from common VoD systems. We model each content as an HD video file with a content length of 25 min. The content stream rate is 5 Mbps, out of which 192 kbps are related to audio. The content catalog is composed of a set of 10.000 movies published by a single producer. Each movie, in turn, has a fixed number of versions, equivalent to the available options of

audio channel, which is set to the number of different geographical regions according to the topology trace. Without loss of generality, objects distributed to clients are divided in 50 kB chunks.

Requests for contents are generated continuously at each network node with intervals defined by a Poisson process. Content selection is made according to a Zipf popularity distribution with its α parameter set to 0.7 and 1.2. Such values are similar to those employed in current literature (ROSSINI et al., 2014), and encompass the popularity curve known to model contents in a VoD system ($\alpha = 1.0$) (CHOI; REAZ; MUKHERJEE, 2012). The content version, in turn, is selected by users according to their location in the network, which is the same of the network node they are connected to. Each node, in turn, is assumed to have a locality distinct to each other in the network. We employ such a behavior to simulate the effects of locality in object requests (for example, audio files tend to be chosen according to country).

4.3.3 Network infrastructure

We use topologies based on real traces obtained from the Internet Topology Zoo (KNIGHT et al., 2011). We experimented with multiple topologies and found that different configurations yielded results with congruous behavior. Thereby, the results presented later are based on one representative topology, namely the British Telecom Latin America. This topology presents a total of 45 nodes and 50 links among them. Routers employ a shortest path routing model that forwards interests according to the shortest path to the publisher. Consequently, from the 50 topology links, 44 are actually used for content distribution.

Regarding routers cache, previous studies argue that the available space will be small with respect to content catalog in order to maintain line rate lookup speeds (ZHANG; LI; LIN, 2013). Thus, our evaluation uses cache with space equivalent to 1% of the content catalog size of the default NDN case. The admission policy used in caches is leave copy everywhere (LCE), while the eviction policy is least recently used (LRU). Finally, regarding the content publisher, we position it on a randomly selected node, which varies in each experiment runs.

4.3.4 Execution

The execution starts with all contents published and empty caches. A *warm up* time is employed to stabilize network conditions prior to observation. It is comprised by the period since the beginning of execution until the moment caches are completely filled, as in (ROSSINI; ROSSI, 2013). After the warm up period, we let the network execute for 60 minutes. The results presented in Section 4.4 consider values after warm up.

Our simulation campaigns are comprised of multiple executions of both scenarios (NDNrel and default NDN). The results presented in the next section are based on central tendencies from multiple executions. We summarize the parameters of our simulation in Table 4.1.

Table 4.1: Simulation parameters.

Parameter	**Value**
Content catalog size	10,000
Content popularity	Zipf with $\alpha = \{0.7, 1.2\}$
Content length	25 min
Content versions	44
Chunk size	50 kB
Total content bitrate	5 Mbps
Audio content bitrate	192 kbps (each channel)
Video content bitrate	4,808 kbps
Topology	45 nodes, 44 active links
Cache size	1% of catalog (default NDN)
Cache policies	LCE, LRU
Simulation time	60 min

Source: Authors (2016).

4.4 Evaluation Results

We now focus on a thorough evaluation of the proposed case study. Prior to each result, we describe the computed metrics and any particular simulation configuration where appropriate. As expected, the obtained results will provide detailed evidence on how relation-based decomposition (*i*) improves the performance of network clients, and (*ii*) contributes to the better utilization of network resources.

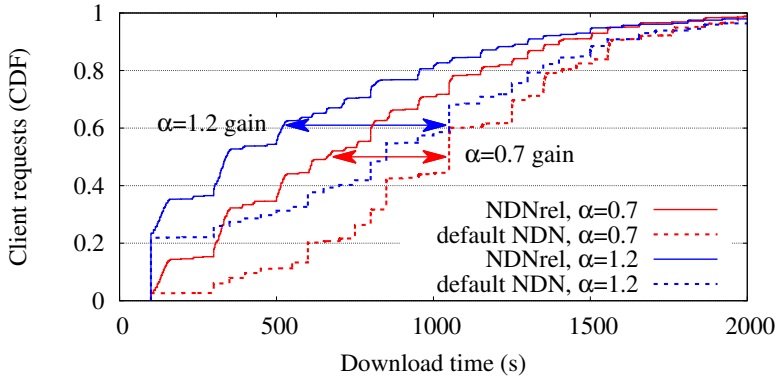
4.4.1 User Performance

Our analysis begins with a comparison of user performance between NDNrel and default NDN. In the evaluation, besides the client requesting contents, we also consider the publisher as a network user. Thus, we focus firstly in two metrics: the *client download time* and *data volume served by the publisher* (or publisher load). The first metric reflects the QoE perceived by clients. The second metric, in turn, is an indicative of the resources required by a publisher to distribute content. With the proposed mechanism we expect both clients download time and publisher load to be reduced. Finally, to complement the results from the aforementioned metrics, we focus on the *request hop distance*. This last metric indicates how far requests are forwarded before fulfilled and, consequently, the efficiency of in-network caching.

4.4.1.1 Client download time

Download times observed in our experiments are presented in a CDF, depicted in Figure 4.3. The figure has two pairs of curves, presenting the results for scenarios with either $\alpha = 0.7$ or $\alpha = 1.2$. Each pair, in turn, compares the performance of NDNrel and default NDN.

Figure 4.3: CDF of clients download time.



Source: Authors (2016).

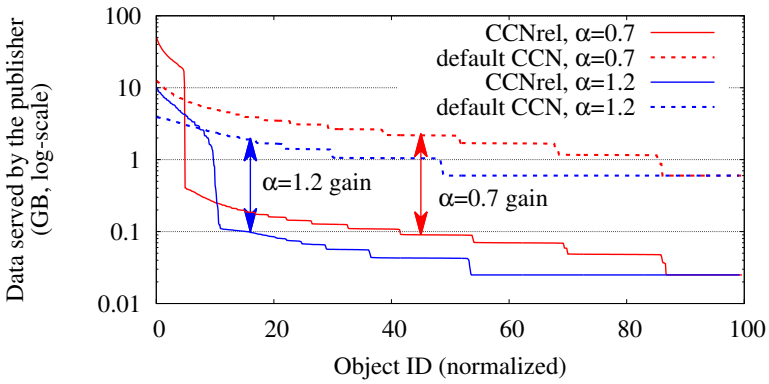
Figure 4.3 shows an improvement of average client download times when NDNrel is used in both content popularity scenarios. When popularity is configured to $\alpha = 0.7$, NDNrel reduces download times 29.2% on average. This occurs because the proposed mechanism eliminates redundancy of objects through decomposition. As result, requests previously scattered among duplicated objects become concentrated in less chunks, in-

creasing their distribution performance. With $\alpha = 1.2$ the achieved reduction of download times is of 34.3% in average. This is explained by the increased concentration of requests to already popular objects, which amplifies the benefits of the mechanism (as demonstrated later in our analysis). In the curves from scenarios with $\alpha = 1.2$ the first 20% of requests present very similar download times, indicating a negligible impact. We verified that these requests are directed to objects with high popularity, which were stored in caches at 1-hop distance from clients. Consequently, these requests are fulfilled with very small latency, independent of NDNrel usage.

4.4.1.2 Publisher load

The publisher load, in turn, is depicted in Figure 4.4. Its horizontal axis presents the objects ordered by their popularity while the vertical axis (which is in logarithmic scale), the volume of data transferred by the publisher for each object (the lower, the better). Similarly to the previous graph, Figure 4.4 depicts two pairs of curves to compare the performance of NDNrel and default NDN under different popularity configurations.

Figure 4.4: Publisher load.



Source: Authors (2016).

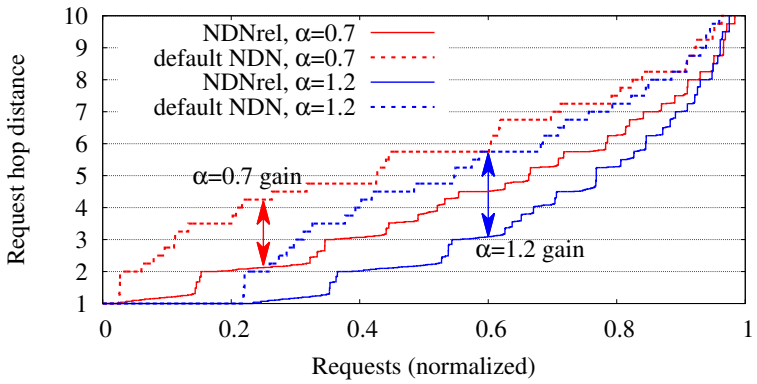
Results show that, for both values of alpha, there is a group of objects (10% of the catalog) that generate a higher data volume with NDNrel. This occurs because requests previously scattered among different copies of duplicated data are now concentrated in one object due to redundancy elimination. Because of the higher request ratio, objects with higher popularity generate more traffic volume in the entire network, including the publisher. However, the remaining objects present a reduction on the data volume served by the publisher. This happens because these objects become smaller with redundancy elimination (as explained later in our analysis). Looking at the overall data volume served

by the publisher, we observe a difference of 45.7% when $\alpha = 0.7$ and 55.7% when $\alpha = 1.2$.

4.4.1.3 Request hop distance

The gains perceived by users are a consequence of the positive effects of NDNrel over the network infrastructure. This is evidenced by the *hop distance of requests*, which we explore next. We consider as request hop distance the number of hops traversed by a request prior to reaching a content copy. Its value depends on the topology properties and the shortest path between clients and the publisher. In simulations, the distance between clients and the publisher had a minimum of 2 hops, a maximum of 10, and an average of 5.3. Shorter hop distances indicate that requests are fulfilled by copies from caches closer to clients (the lower the curve, the better). We expect the use of NDNrel to result in a reduction on the overall hop distance. Figure 4.5 depicts the values of this metric observed in the experiments. Its horizontal axis presents the normalized number of requests ordered by their respective hop distance.

Figure 4.5: Request hop distance.



Source: Authors (2016).

The request hop distance corroborates the results observed in user performance. That is, we observe a general reduction on the hop distance when NDNrel is used. This result demonstrates that in-network caching is fulfilling more client requests in cache units closer to requesters. Consequently, the server load and the download time of clients are reduced, improving overall distribution performance. Moreover, the value of α also affects the hop distance of requests. In average, when $\alpha = 0.7$, the hop distance reduction offered by NDNrel is of 26.3% and, when $\alpha = 1.2$, the hop distance reduction is of 34.3%. This indicates that the gain provided by NDNrel depends on the value of α. In

the curves from scenarios with $\alpha = 1.2$ the first 20% of requests are fulfilled within 1 hop. As previously explained, these requests belong to very popular objects that tend to constantly remain in cache.

So far, we found out that *relation-based decomposition improves user performance because it reduces both client download times and requests served by the publisher.* This is a consequence of smaller request hop distance, which is a result of better network resource utilization due to less data redundancy. Next, we explore how NDNrel influences network resources, such as bandwidth and cache space.

4.4.2 Network Resources Utilization

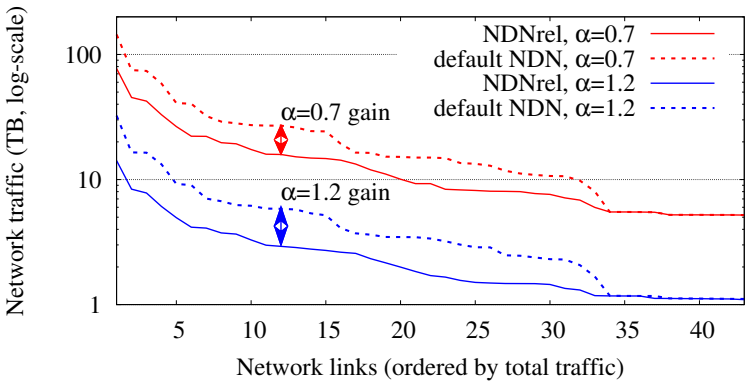
Next we explore the impact of NDNrel on the behavior of network resources efficiency. Thus, we first explore the metrics of *network traffic* and *cache hit ratio*, which indicate if the available network bandwidth and cache space, respectively, are efficiently used. To further explain the behavior of network resources efficiency, we explore the *number of caching operations*, which indicates how frequent content is substituted in caches. Finally, we focus on the *object request distribution*, which indicates how the object catalog popularity is influenced by NDNrel and the consequent impact on the behavior of in-network caching.

4.4.2.1 Network traffic

A consequence of the smaller request path size is the reduction of the *network traffic* generated by content distribution. Requests are satisfied by caches closer to the clients, thereby reducing the number of links through which data is transmitted and, consequently, the overall network traffic. We illustrate in Figure 4.6 the traffic observed in our experiments. The horizontal axis presents topology links used for content transmission, while the vertical axis, the network traffic in logarithmic scale.

Considering the overall network traffic, NDNrel offers a reduction of 33.6% on transmitted data when $\alpha = 0.7$, while with $\alpha = 1.2$ the reduction is of 42.4%. There are two distinct behaviors depending on the distance of the link to the content publisher. Links closer to the publisher transmit requests (and data) for multiple content versions because they are hubs from multiple network regions. Since NDNrel enables objects to be shared among different versions, cache is used more efficiently than in NDN, resulting in objects

Figure 4.6: Total network traffic: (average gain of 34% and 42%, shown in log scale).



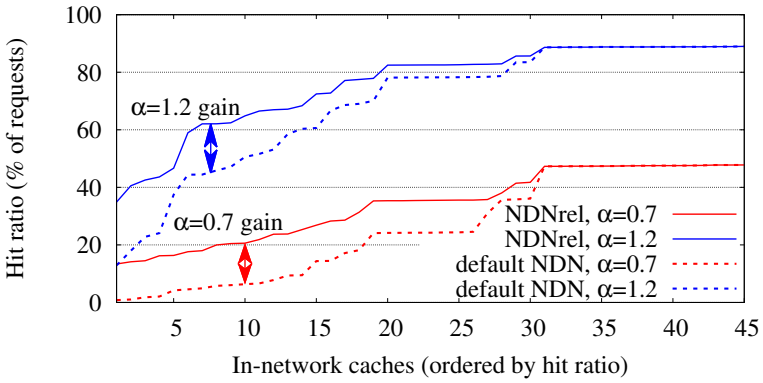
Source: Authors (2016).

stored closer to the edge of the network. Therefore, the traffic reduction on these links is higher. The other behavior concerns links connected to leaf routers. They only transmit data from a single version to clients, causing most (or all) of the popular contents to be in cache, in both NDNrel and default NDN. Consequently, these links present small traffic difference when NDNrel is used.

4.4.2.2 Cache hit ratio

The reduction of the network traffic and requests path occurs because in-network caches present a higher *hit ratio*. Figure 4.7 depicts the hit ratios observed in our experiments. Its horizontal axis presents the caches from each router ordered by observed hit ratios.

Figure 4.7: Cache hit ratio.



Source: Authors (2016).

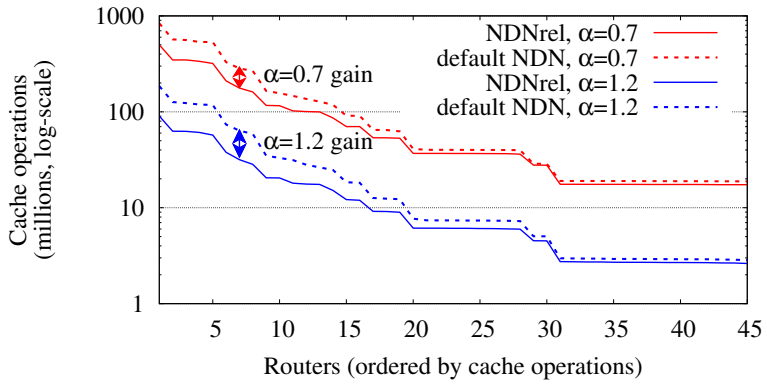
Results show that NDNrel offers an average improvement of 33.6% in the cache hit ratio when $\alpha = 0.7$ and 9.8% when $\alpha = 1.2$. NDNrel presents a smaller hit ratio gain

with $\alpha = 1.2$ because caches in the default NDN have a considerable efficiency gain with such a content distribution. As result, the average NDNrel gain is reduced in comparison to $\alpha = 0.7$. Regarding the behavior of NDNrel, similar to network traffic, routers have their performance based on topological position. Routers closer to the publisher (or core routers) may serve several different versions of the content. With NDNrel, a smaller number of chunks must be cached to distribute all content versions, allowing more objects to be stored in each router. By covering a higher percentage of the content catalog than NDN, core routers in the NDNrel scenario have higher cache hit ratio. Edge routers (leafs of the spanning tree) need to serve only one version. Consequently, NDNrel will have negligible impact to the cache hit ratio in these routers.

4.4.2.3 Caching operations

Caches are more efficient with NDNrel because their content changes less frequently. This effect can be evaluated by the *number of operations performed by caches*, as measured in (CHAI et al., 2013). This metric presents the number of storage and eviction operations performed in each network cache. When less content substitution occur there is a higher chance that a content will remain available to fulfill new requests. We compute the number of cache operations from our experiments and depict it in Figure 4.8. The horizontal axis presents the topology routers in order of cache operations.

Figure 4.8: Caching operations: (average gain of 32% and 44%, shown in log scale).



Source: Authors (2016).

Results show that NDNrel reduces cache operations in 32.1% when $\alpha = 0.7$ and 43.9% when $\alpha = 1.2$. This phenomenon is an effect of the reduction in the object catalog data redundancy. As result, the cache space available for other contents to be stored is increased and the the substitution of contents due to cache eviction policies is reduced.

Consequently, when a content is stored it will remain longer in caches because there is a smaller chance that it will be substituted. Values of cache operations follow the same behavior from results of traffic and cache hit ratio: the gain in the number of operations is higher in routers closer to the publisher. The higher performance difference with $\alpha = 1.2$ occurs because requests become even more concentrated, further reducing the cache operations due to less distinct objects forwarded by routers.

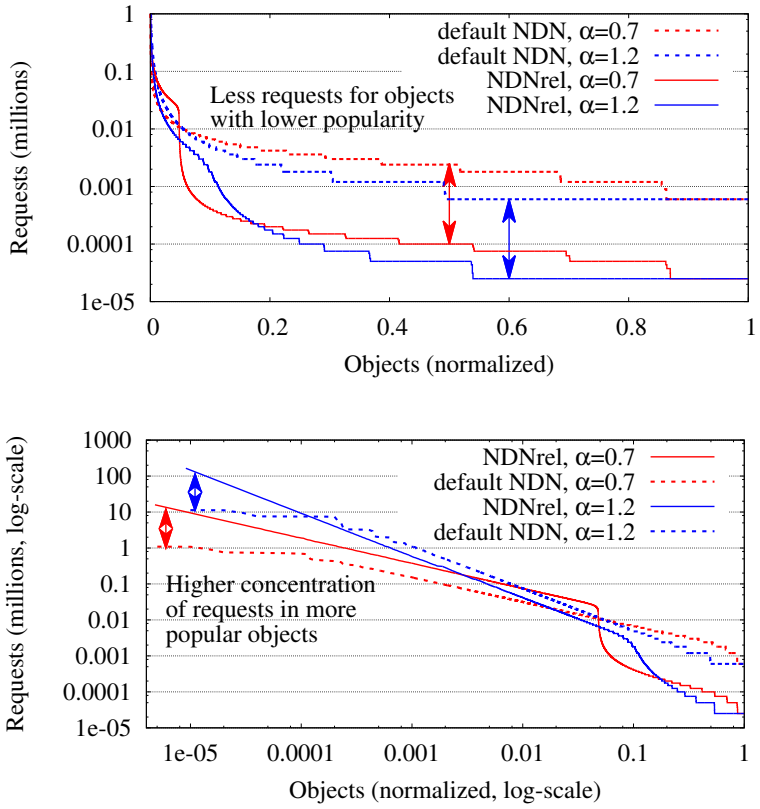
4.4.2.4 Distribution of requests

NDNrel reduces cache operations because it alters the *distribution of requests* for different objects. Recall from the previous section that clients select (*i*) a content to request based on a Zipf distribution and (*ii*) an audio version based on the locality of the network node they are connected to. These two factors and the use of NDNrel will directly influence the distribution of requests to the available object catalog. We expect NDNrel to concentrate requests even more into a smaller group of objects in comparison to default NDN. The request distributions observed in our experiments are depicted in Figure 4.9. All graphs present on the horizontal axis the objects ordered by their popularity. Also, the vertical axis of all graphs are in logarithmic scale to ease the visualization of low popularity contents. We present results in two graph versions for the sake of clarity (horizontal axis are either in linear or logarithmic scale).

Results show that NDNrel reduces the request rate of the least popular objects in approximately 96% for both values of α. Also, the data request volume of the most popular objects with NDNrel is 13.6 times higher than NDN with $\alpha = 0.7$ and 14.6 times higher with $\alpha = 1.2$. This occurs because the long tail of the distribution is mostly composed of objects carrying audio information. These objects are requested by clients interested in a specific content version and are composed by a smaller number of chunks. Video objects, in turn, will have their popularity increased, being concentrated on the beginning of the curve. This occurs because the requests from redundant data will be concentrated on a common (smaller) set of chunks that are the result from content NDNrel.

The behaviors observed in the request distribution and cache efficiency occur due to the number of published objects and their sizes. Recall that in our evaluation scenario 10,000 contents are available. In the default NDN each content version results in a complete object with audio and video. Therefore, considering 44 content versions, there are 440,000 objects available to clients. Taking into account content length and chunk size,

Figure 4.9: Object request distribution.



Source: Authors (2016).

the above object catalog results in approximately 412 billion chunks. In the NDNrel case, however, contents generate a total of 450,000 objects, a 2.3% increase compared to default. However, the number of chunks is decreased to 16.5 billion, a 96% decrease in the catalog space. This reduction occurs because, with NDNrel, only 10,000 objects represent the video channel. The remaining 440,000 are only audio data, which is 96% smaller than video.

Summarizing, we observed that *relation-based decomposition improves the utilization of network resources, reducing network traffic and increasing the hit ratio of caches*. This occurs because *relation-based decomposition reduces the catalog storage size and modifies the distribution of requests among objects in a way that improves cache usage and stability*.

The case study presented in this chapter demonstrates the potential advantages of using relations to distribute contents. However, it did not present a detailed analysis of the overhead caused by the relations mechanism. To fill this gap, we developed a new case study, which is explored in the next chapter.

5 CASE STUDY: WEB CONTENT

In this case study we focus on Web content, another popular application that significantly contributes to global Internet traffic (CISCO, 2016). Data from Web systems presents significant differences from those explored in the previous case study, especially regarding the size of data objects. According to our observations from real Web traffic (presented in more detail later in this Chapter), the size of a HTML object can vary from 10 kB to up to 3 MB, but approximately 95% of objects are smaller than 100 kB. Because of the small size of data objects, the use of relations to enable composition in Web pages generates a higher latency for clients and traffic for the network due to the acquisition of relation information for each accessed page. However, the reuse of objects lowers data redundancy, resulting in less space required to publish pages and higher cache hit ratio, potentially contributing to client performance.

This section details the case study developed to evaluate the use of composition based on relations to distribute Web content in ICN. It first presents an overview of Web content distribution in ICN based on previous literature. Next, it explores the use of composition based on relations to publish and distribute such content. Then, the section presents the parameters and metrics used to compare the use of relations against a baseline scenario.

5.1 Web Systems in ICN

The distribution of Web content in ICN can be trivially implemented with two steps: convert the files stored in a current Web server to individual data objects and the HTTP URLs contained in HTML files to identifiers of the ICN namespace. Such an analysis is naïve (PENTIKOUSIS et al., 2015a) because it considers only static content. It ignores dynamically generated content, which is highly relevant to the current Web.

Moiseenko, Stapp and Oran (2014) analyze the challenges related to the implementation of URL parameters from current Web systems in an ICN environment. They explore two possible alternatives to implement parameter transmission in ICN queries: (*i*) direct inclusion in the name or (*ii*) creation of additional fields in ICN request messages. The first method does not require modifications on current ICN architectures but incurs a higher overhead to routing mechanisms because of larger names. In turn, the second method requires the addition of parameter fields in ICN request messages. Request

parameter fields also must be considered by in-network caching because a single name with different parameters may result in different data. Consequently, their use increases the complexity of in-network caching.

We argue that the method employed to send parameters to publishers is an orthogonal issue to the use of relations to describe Web contents in ICN. The only restriction is that a data object must be uniquely identified for caching and request aggregation purposes. This restriction is in line with the fundamental requirement that a name cannot represent two distinct blocks of data in ICN. In our analysis of Web systems in ICN, we assume that parameters are added directly to the requested name. We assume the usage of this method because it is natively supported by current ICN architectures.

Regarding the composition of HTML content, our case study assumes that the structure of the HTML content does not change in comparison to the current Web. The only notable difference to current HTML is that URLs are converted to the ICN architecture naming scheme (NDN in our case). The case study also assumes that the main HTML is distributed as a single block of data. That is, the HTML data is not separated in different parts as proposed by Wang, Krishnamurthy and Wetherall (2014). This assumption is in line with the behavior of current Web systems. An exception to it might be the use of asynchronous Javascript requests (AJAX) to load additional HTML content after the initial page load. However, data loaded from asynchronous requests can also be divided in multiple parts depending on its complexity. In our case study we focus on the HTML content obtained in the initial load of the page and do not consider the use of asynchronous requests.

The above assumptions define the *baseline scenario* used in our evaluation, without considering the use of relations. This scenario is compared against one that employs relations to describe the structure of HTML data. The proposal is explored next.

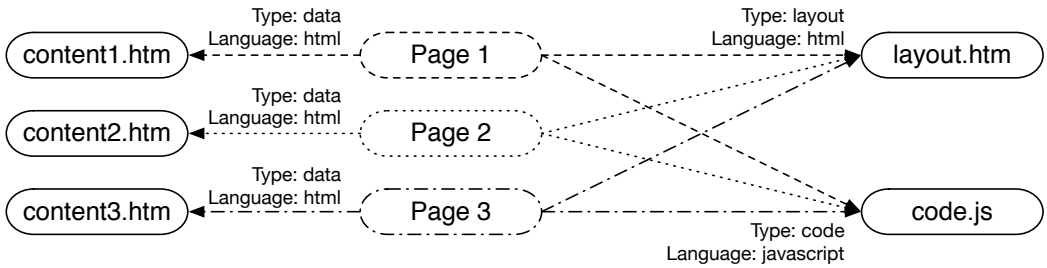
5.2 A Model for Web Pages based on Relations

We propose to use relations to publish Web pages in ICN based on composition with data of objects already available in the network. More specifically, instead of using one object to transmit the complete HTML data of a page (our baseline scenario), HTML is divided into multiple objects to enable reuse of common parts among multiple pages. Such a division is based on the study from Wang, Krishnamurthy and Wetherall (2014) about the use of micro-caching in Web applications. The study shows that HTML data

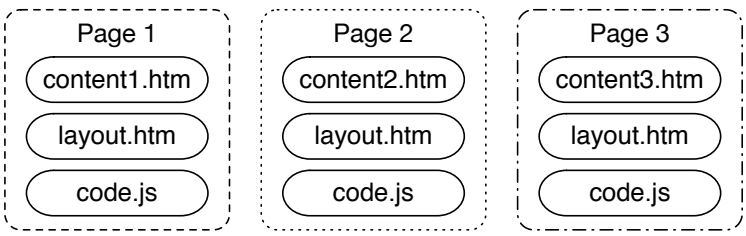
from one page can be divided in three main elements: *layout*, *data*, and *code*. The layout part contains the HTML portions that define the general layout of the Web page, which might be reused by different contents. The data part contains the actual information of interest to the user (e.g. the actual blog post). Finally, the code part contains Javascript code used to manipulate the remaining page elements. According to the authors findings, each part presents distinct behavior regarding request rates and update intervals. The reutilization of the code element is used in the current Internet for Javascript libraries. In summary, popular CDNs distribute current versions of popular libraries that can be linked by any publisher on the Internet. The advantage is that these libraries are then downloaded through a highly available CDN instead of the publisher site, which can be a small infrastructure with few resources. The idea from our case study is to employ this strategy to more elements of HTML code and employ an ICN for content distribution.

Figure 5.1(a) illustrates a Web page structured using relations. For the sake of comparison, Figure 5.1(b) presents the same page described according to our baseline scenario. With relations, the structure described in a single HTML file is divided into three parts. The publisher can create new contents by creating a new manifest that links a new data object with existing layout and code. In turn, clients obtain the relation structure description prior to obtaining the actual Web page data. After obtaining the relation structure, the client can identify and request the objects required to reconstruct the page.

Figure 5.1: Web page structure example.



(a) Relations



(b) Baseline

Source: Authors (2016).

The main advantage of the proposed method to compose Web pages is the increased caching performance due to the reuse of data. This leads to additional performance improvements such as smaller download latency and network traffic. Additional advantages can be achieved with the use of relations to publish Web content in ICN. For example, relation attributes can be used to list the parameters used in each HTML part. A Web page may include a parameter that customizes the content of its data part, but that does not influence layout and code. In this case, the parameter should be applied only to the data part request and not to the remaining objects. The relation to the data part can have an attribute describing the availability of such a parameter. Relations can also be used to simplify Web page customization. For example, relations can include two page layout entries, one destined to desktops and the other, to mobile clients. The client can request only the layout of choice without additional programming on the publisher side.

In our case study we investigate the performance gains obtained when an HTML page is divided into data, layout and code and the latter two are used to compose new contents. Each of the three HTML parts present different behavior regarding its request rate when multiple Web pages are created using composition. In the next section, we discuss the methodology to be used to evaluate the advantages of using relations to compose Web pages distributed via ICN.

5.3 Evaluation Methodology

To guide our analysis we defined three main research questions to be answered by a series of experiments:

Q1: How does the use of relations affect the distribution of client requests to objects? Our goal is to identify the impact of relations in the popularity of objects and how it may potentially affect the network mechanisms.

Q2: What is the effect of relations on the usage of network resources? Our goal is to analyze the variation of network bandwidth and cache space required when the relations are used.

Q3: Does the overhead generated by relations negatively impact application performance in a scenario with contents of small size? Our goal is to study whether the use of relations with small sized objects adds a significant overhead or improves the client performance.

To conduct our experiments, we used our extension of the ndnSIM simulator with NDNrel and added support for content publication with object composition. We use the simulator to evaluate scenarios where the publisher creates new HTML content through the composition of already available objects. Next, we describe the scenario and parameters employed to evaluate the case study.

5.3.1 Simulation Scenario

Our simulation employs two main scenarios: *NDNrel*, that uses our relations mechanism to distribute HTML content with object composition, and *default NDN*, that uses an unmodified version of NDN to distribute the content. Based on results from Wang, Krishnamurthy and Wetherall (2014), we divide the HTML content into three main elements: *layout* (CSS and HTML from page layout), *code* (Javascript and HTML from scripting), and *data* (XML and HTML from actual content). We assume that a complete HTML content requires all three parts, which may be distributed as individual objects together with their describing relations (NDNrel scenario) or encapsulated in a single object variant (default NDN scenario). The results presented in Section 5.4 focus on the performance difference between NDNrel and default NDN.

5.3.2 Content and Workload

We analyzed the trace of a proxy server during two months to capture the Web content size distribution. Our analysis observed a variation of HTML objects between 1 kB and 3.26 MB, with an average size of 10 kB. These values are used to describe the content objects (composed of all three components: layout, code, and data) in our simulation. Regarding the size of individual parts in the NDNrel scenario, we employ the observations from Wang, Krishnamurthy and Wetherall (2014), which defines a distribution between layout, code, and data approximately to 30-30-40%. The transmission of the objects to clients is done, without loss of generality, using chunks with a maximum size of 1 KB.

The content catalog begins the experiment with 10,000 Web pages and expands throughout the simulation. A publisher creates new *variants* of the starting (original) contents every 2 minutes, an update interval proportional to the observations presented by

Wang, Krishnamurthy and Wetherall (2014). In the NDNrel scenario, each new variant generates an HTML content and a manifest that identifies the objects necessary to reconstruct it. The HTML content created reuses a previously published layout and code objects while it adds a new object as the data part. In the default NDN scenario, the publisher creates a new object containing all three components for each new variant.

Requests for contents are generated continually at each network node with intervals defined by a Poisson process. Content selection is made according to a Zipf popularity distribution with its α parameter set to 0.7 and 0.9, which encompass the popularities observed in known studies about HTML content (PENTIKOUSIS et al., 2015b).

5.3.3 Network infrastructure

The experiments are executed using topologies based on real traces obtained from the Internet Topology Zoo (KNIGHT et al., 2011). We experimented with multiple topologies and found that different configurations yielded results with congruous behavior. Thereby, the results presented later are based on one representative topology, namely the British Telecom Latin America, which has 45 nodes and 50 links connecting them. In the simulation, NDN packets are routed according to the shortest path to the publisher, creating a spanning tree rooted in the content provider. Consequently, from the 50 topology links, only 44 are used for content distribution.

Regarding the routers cache, previous studies argue that the available space will be small compared to the content catalog size to maintain line rate lookup speeds (ZHANG; LI; LIN, 2013). Thus, our evaluation uses cache with storage equivalent to 1% of the initial content catalog size of the default NDN case. We employ the default on-path cache from the NDN architecture with LCE (leave a copy everywhere) and LRU (least recently used) policies for cache placement and eviction, respectively. Finally, the content publisher is positioned on a randomly selected node, which varies in each experiment run.

5.3.4 Execution

The execution starts with the initial contents published and empty caches. A *warm-up* period, ranging from the beginning of execution to the moment that the caches are filled (as in Rossini and Rossi (2013)), is employed to stabilize network conditions

before observation. After the warm-up period, the simulation executes for 60 minutes to measure and collect the evaluated metrics.

The simulation campaigns are comprised of multiple executions of both scenarios (NDNrel and default NDN). The results presented in the next section are based on the central tendencies of the collected metrics. We summarize the parameters of the simulation in Table 5.1.

Table 5.1: Simulation parameters.

Parameter	**Value**
Initial content catalog size	10,000
New content publication	New content variants every 2 minutes
Content composition	3 parts: layout (30% of content size), code (30%), data (40%)
Content popularity	Zipf with $\alpha = \{0.7, 0.9\}$
Content size	Trace: min 1 KB, max 3.26 MB, average 10 KB
Chunk size	1 KB
Topology	45 nodes, 44 active links
Cache size	1% of catalog (default NDN)
Cache policies	LCE, LRU
Simulation time	60 min

Source: Authors (2016).

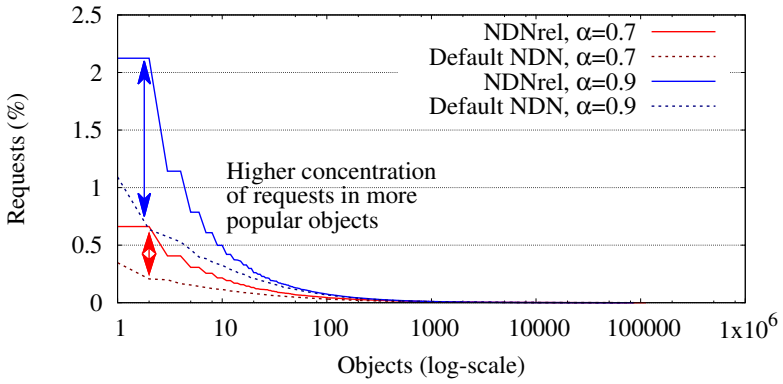
5.4 Results

We present in this section the results obtained from the evaluation scenario previously described. The relation mechanism analysis is divided into the following three parts. First, we discuss its impact on the object catalog, notably its popularity distribution, and caching. Then, we show the benefits of using relations to the end users (clients and publisher). Finally, we investigate the effects on the network, shedding light on the reasons why relations improve the performance of end users. Before the discussion of each result, we describe the evaluated metric and its goal in our analysis.

5.4.1 Object Catalog

The first step in the evaluation is the analysis of how the relation mechanism affects the object catalog and the in-network caching. We investigate the object popularity distribution in both scenarios through the number of times they were requested, shown in Figure 5.2. Its horizontal and vertical axes represent respectively the published objects (ordered by their popularity) and the percentage of requests issued for a given object. Recall from Section 5.3 that, in the default NDN scenario, an object contains a complete content with all three parts, while in the NDNrel scenario, each object carries an individual content part. We expect that the reuse of objects enabled by the relations will increase the popularity of the objects shared by multiple variants. This behavior is beneficial to the performance of the end users and the network in NDN because it leverages the in-network caching for a better content dissemination, as it will be shown throughout the evaluation.

Figure 5.2: Requests to objects.



Source: Authors (2016).

The object catalog begins with 10 thousand (default NDN scenario) or 30 thousand (NDNrel scenario) content objects and evolves through time with the creation of new variants, ending with up to 300,000 (default NDN scenario) or 320,000 (NDNrel scenario) objects. There are three types of objects in the catalog regarding their popularity: popular, unpopular, and ignored. The popular objects are those that receive at least 0.01% of total requests. The unpopular ones are the requested objects that are not popular. Lastly, the ignored objects receive no requests throughout the experiment.

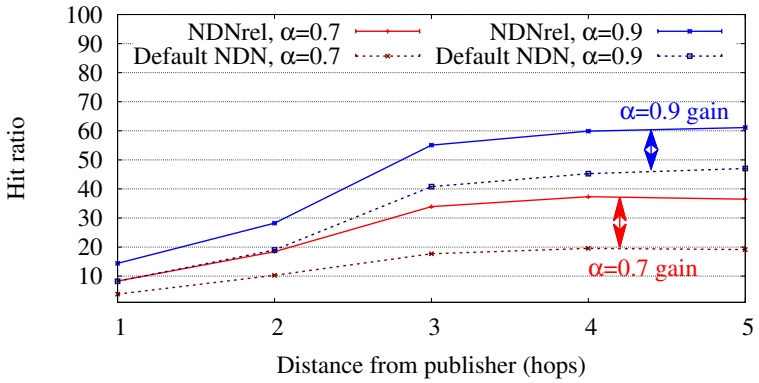
Figure 5.2 shows that the distributions of popular, unpopular, and ignored objects in the default NDN scenario are 0.6-30.2-69.2% ($\alpha = 0.7$) and 1.5-20.3-78.2% ($\alpha = 0.9$). The high percentage of ignored objects is a combination of the popularity distribution and the lifetime of contents in the simulation (i.e. objects published towards the end of

the simulation have a smaller chance of being requested). Regarding the popular ones, although they comprise a small percentage of the catalog, they are responsible for 14.5% ($\alpha = 0.7$) and 35.3% ($\alpha = 0.9$) of total requests of the clients.

The relation mechanism changes the requests issued because it enables objects to be shared among different contents. On the one hand, the results show that the distributions of popular, unpopular, and ignored objects in the NDNrel scenario remain similar to the default NDN one with values of 0.8-34.4-64.8% ($\alpha = 0.7$) and 1.2-25.5-73.3% ($\alpha = 0.9$). On the other hand, the popular objects concentrate a higher number of requests: 25.6% of total requests when $\alpha = 0.7$ and 45.1% when $\alpha = 0.9$. The shared objects accumulate more requests because each of them is a component required by multiple content variants. Therefore, every client interested in one of the variants needs to request the shared object to retrieve the complete content.

Figure 5.3 shows the impact of the relation mechanism on the in-network cache hit ratio. The routers are categorized in tiers according to their distance (in hops) to the publisher. The horizontal axis presents the router tiers, and the vertical axis describes the average hit ratio of the routers at the given tier. We expect that the use of relations will increase the cache hit ratio due to the alteration of the content popularity distribution, which makes the popular contents even more popular. In particular, the routers farther from the publisher (i.e. in the leaves of the spanning tree) will benefit more because they receive fewer requests and with less variability than those in the core.

Figure 5.3: Cache hit ratio.



Source: Authors (2016).

In the default NDN scenario, routers have an average cache hit ratio of 17.5% ($\alpha = 0.7$) and 40.3% ($\alpha = 0.9$). The difference in the results is a consequence of the object popularity distribution employed to issue content requests. We confirm that when more requests are concentrated on the popular objects ($\alpha = 0.9$), there is a higher

probability of a cached content being requested because of the caching policies used in NDN. In the case of requests being more evenly distributed ($\alpha = 0.7$), the benefit from caching is lower due to the high variance of requests, resulting in a smaller probability that a given content will be cached.

Analyzing deeper into the router tiers, we observe that the cache hit ratio improves as the routers are farther from the publisher. The results vary between 4% and 19% ($\alpha = 0.7$) and from 8% to 48% ($\alpha = 0.9$). The leaf routers (farther from the publisher) serve only a portion of the users, resulting in a small variance of requests for different contents and a higher probability of having them cached. Core routers (closer to the publisher), on their turn, receive the aggregation of multiple leaf routers and users, which causes a higher variance of the requested contents and a lower probability of having them in the cache.

The relations mechanism improves the usage of the in-network caching because of the higher concentration of requests in the popular contents. In average, the routers enhance their cache hit ratio to 33.5% ($\alpha = 0.7$) and 54.1% ($\alpha = 0.9$). These results are intuitive considering the higher concentration of requests to the most popular contents caused by the relations. Specifically, the relation mechanism reuses some published objects to compose various contents, increasing the popularity of the reused objects and concentrating even more the requests for them. The results of the router tiers follow the same trend described in the analysis of the NDN default scenario but with better results. The routers cache hit ratio varies between 8% and 38% ($\alpha = 0.7$) and from 15% to 61% ($\alpha = 0.9$).

In summary, *the use of relations increases the request rate of the popular objects because of the shared objects used to represent contents, which increases the routers cache hit ratio.* They allow the reuse of content parts (that would be duplicated in each variant), removing the data redundancy and concentrating the requests from all different content variants in a single object. Having very popular objects is beneficial for the content dissemination performance because it improves the efficiency of the NDN caching mechanism.

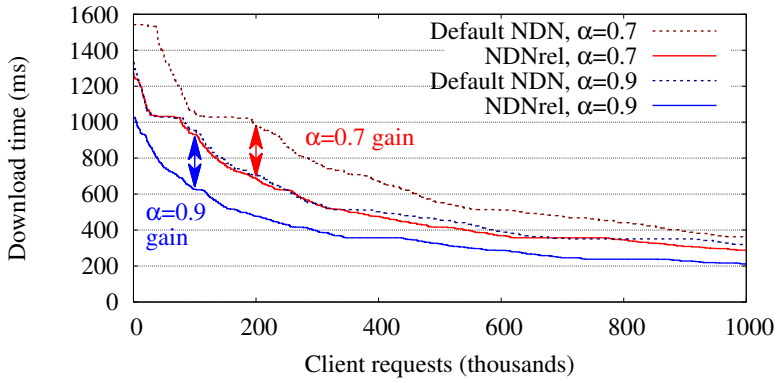
5.4.2 End Users

This section analyzes the benefits caused by the relation mechanism to the end users: clients and publisher. Particularly, we focus on evaluating the clients download

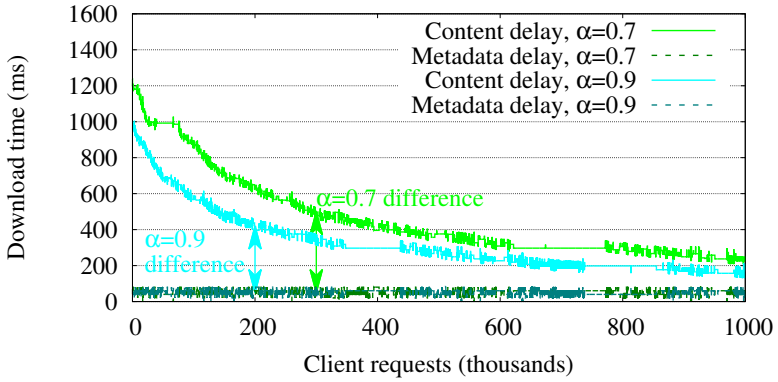
time and the publisher load.

Figure 5.4(a) illustrates the download time of clients. The horizontal axis depicts the individual requests from clients while its vertical axis, the download time in milliseconds. This metric represents the period to retrieve the entire content, which begins with the request for either the relation (NDNrel scenario) or the first chunk (default NDN scenario) of an object, and concludes with the retrieval of its last chunk. To further evaluate the effect of relations, Figure 5.4(b) presents the download time of manifests compared to that of content data for each client request. We expect that the relation mechanism will reduce the download time of clients due to a more efficient usage of the in-network caching to disseminate contents, as a result of the change in the popularity distribution of the catalog.

Figure 5.4: Client download time



(a) Download time



(b) Download time in the relations scenario

Source: Authors (2016).

The results show that the download time of clients varies from 300 ms to 1.6 s in the default NDN scenario, with averages of 686 ms ($\alpha = 0.7$) and 531 ms ($\alpha = 0.9$). The observed variation occurs due to the characteristics of each object, especially its

popularity and size. Popular contents are more likely to be found in the router caches than unpopular ones due to their higher request rate and NDN caching policies. Because contents cached on-path are closer to clients than the publisher, retrieving data from them reduces the download time. The impact of the size of an object is intuitive because larger ones require more time to be transmitted than smaller contents.

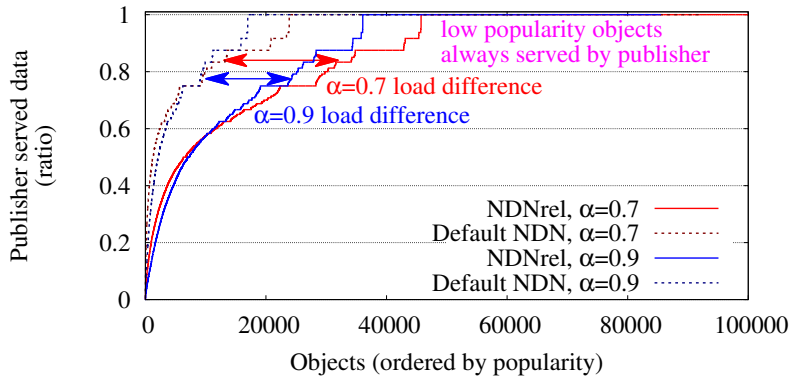
The use of relations reduces the range of clients download times to between 200 ms and 1.3 s. The benefit of relations is more visible in the average download time, which is reduced by more than 25%, achieving 513 ms ($\alpha = 0.7$) and 374 ms ($\alpha = 0.9$). The improvement is a direct result of relations, which increase even more the popularity of popular contents and leverages the NDN caching. Specifically, the routers push the popular objects closer to the clients and satisfy a higher number of requests (discussed later in Section 5.4.3). Regarding the manifest overhead, the average time to retrieve it is only 43 ms (or approximately 10% of the download time), as shown in Figure 5.4(b). Based on our results so far, the benefits obtained with relations justify this additional overhead in the download time.

The results of the publisher load, presented in Figure 5.5(b), demonstrate further the benefits of the relation mechanism. The horizontal axes represent each object ordered by the popularity while the vertical axes describe the two metrics measured: the percentage of requests satisfied (Figure 5.5(a)), and the volume of data (in Kilobytes) transmitted (Figure 5.5(b)) by the publisher. We expect that the use of relations will reduce the publisher load as a consequence of the higher usage of in-network caches to provide the requested contents, as previously discussed.

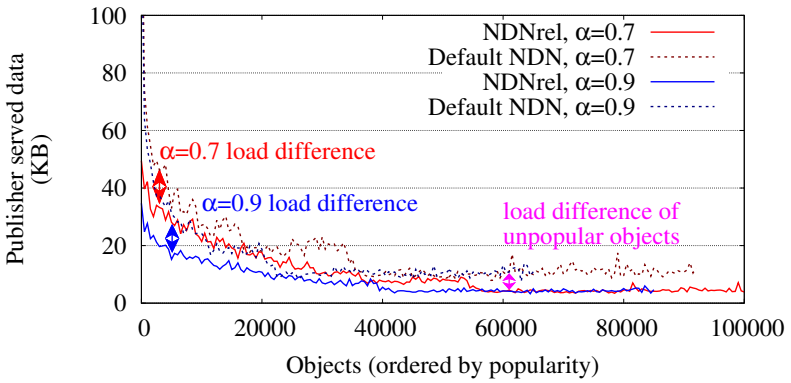
The results of this analysis evidence two groups of objects regarding the providing source: those served by the publisher and in-network caches and those provided exclusively by the publisher. Content provided by an in-network cache reduces the publisher load because it does not need to process the request or send the content data. The results also show that the load reduction is more significant in the popular objects than in the unpopular ones because of their higher request rate.

In the default NDN scenario (for both values of α), approximately 25% of objects are provided partially by the in-network caches, and only 1% have more than half of their requests satisfied by them. The objects that do not benefit from the in-network caching, comprising 75% of the served objects, are provided exclusively by the publisher because of their unpopularity. To complement the analysis, we take a look at the volume of data transmitted by the producer, shown in Figure 5.5(b). The in-network caching helps

Figure 5.5: Publisher load



(a) Request ratio per object



(b) Data per object

Source: Authors (2016).

alleviate the publisher load, keeping the transmitted data between 100 KB and 20 KB when providing the popular objects. Considering the average size of the objects (10 KB), the publisher sends only up to 10 copies of the popular ones to serve a high number of client requests. In comparison, the unpopular ones generate a traffic of 10 KB, but they receive a significantly smaller number of requests.

Analyzing the NDNrel scenario, we conclude that the changes in objects popularity increase the in-network caching and, consequently, reduce the publisher load considerably. In Figure 5.5(a), it is possible to see that caches partially serve almost half of the contents, and around 10% of the objects have less than 50% of the requests provided by the publisher. The impact is also significant in the data traffic (Figure 5.5(b)), which is reduced to between 40 KB and 10 KB for the most popular contents and only 4 KB for the unpopular ones. Although the objects are smaller in this scenario because of the relations (around 4 KB), they also receive more requests. Therefore, the similar proportion of up to 10 copies sent to provide the popular contents means a more efficient usage of in-network

caching to serve the data. Overall, the traffic generated by the publisher is reduced by more than 32%: from 1.6 GB to 1 GB ($\alpha = 0.7$), and from 1.1 GB to 0.7 GB ($\alpha = 0.9$).

In summary, *the use of relations reduces the client download time and the publisher load because of the better usage of the in-network caching to disseminate content.* The benefits are a consequence of the relations, which amplify the advantages of NDN to distribute popular contents in the network.

5.4.3 Network

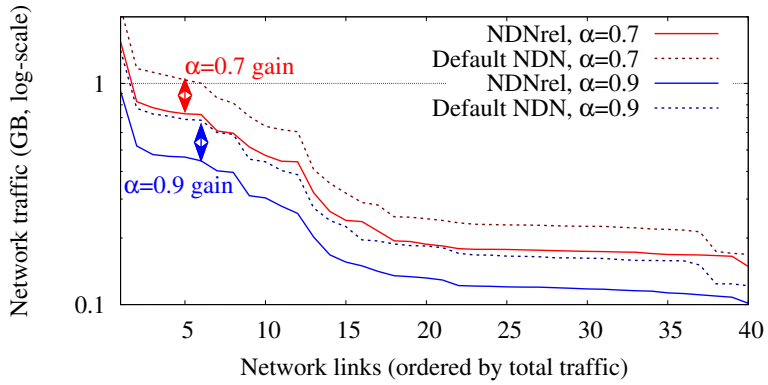
The last part of the evaluation investigates the effects of the relation mechanism on the network, shedding light on the underlying reasons for the better performance using relations. During the analysis, we discuss the overall network traffic and the content placement in the network.

The total network traffic measured in each link during the content distribution is shown in Figure 5.6(a). The horizontal axis presents the topology links ordered by traffic while the vertical axis, the network traffic measured in Gigabytes. In turn, Figure 5.6(b) individually presents the traffic generated by manifests and actual content data. We expect that the relation mechanism will reduce the network traffic because of the better efficiency of in-network caching. This result is visible through the higher cache hit ratio and the placement of contents closer to the clients, discussed later in this section.

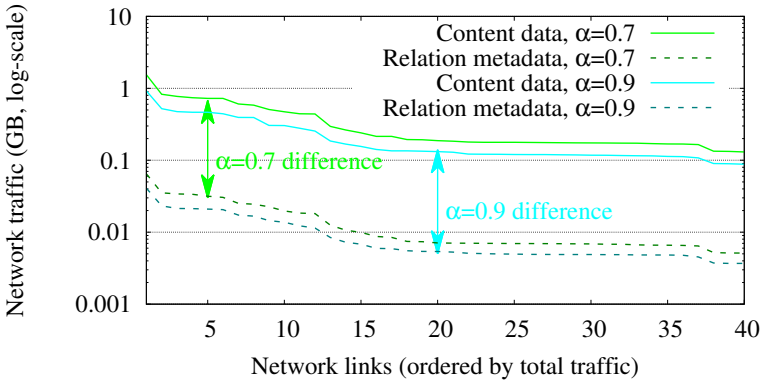
The results show a variation of traffic in the links according to their distance to the content publisher. In the default NDN scenario, the traffic on the links ranges from 2.21 GB to 0.08 GB. Intuitively, because the transmission creates a spanning tree with the shortest path, the links transmit not only the traffic of their local clients but also the aggregation of the lower levels of the tree. Therefore, the closer the link is to the publisher, the higher is its traffic volume. Overall, the content distribution in the default NDN scenario generated a total network traffic of 19.5 GB ($\alpha = 0.7$) and 13.3 GB ($\alpha = 0.9$).

The use of relations reduces the overall network traffic by 26.4% ($\alpha = 0.7$) and 30.7% ($\alpha = 0.9$), down to 14.3 GB and 9.2 GB, respectively. Link-wise, the reduction is proportional to its traffic, being more significant (regarding total volume) in the links closer to the publisher than those farther. The reduced traffic measured in the links spans between 1.55 GB and 0.06 GB and is a consequence of the better usage of in-network caching. Lastly, as can be seen in Figure 5.6(b), the retrieval of the relations corresponds to only 4% of the total traffic, which is insignificant given the overall reduction.

Figure 5.6: Network traffic



(a) Total traffic



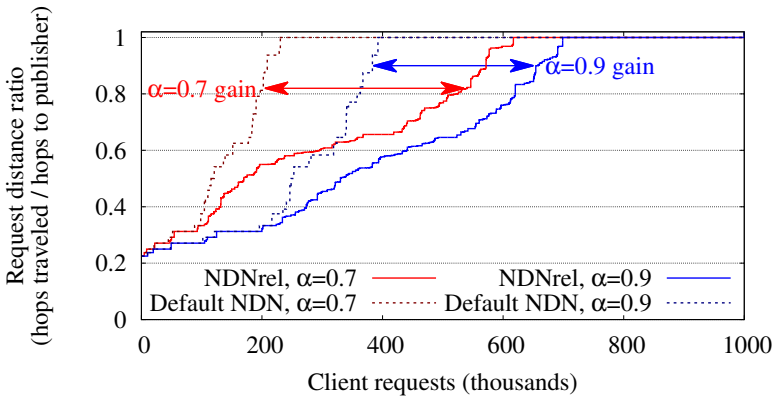
(b) Traffic relations in the relations scenario

Source: Authors (2016).

Figure 5.7 shows the distance that data served to requests traveled in the network. The horizontal axis presents each content request from clients while the vertical axis represents the average distance ratio of all interests sent. The distance ratio is calculated dividing the number of hops from the client to the closest content copy by the distance between the client and the publisher. This metric gives the percentage of the worst-case scenario path that a request needs to travel before finding the data. In the experiments, the median distance to the publisher is of 3 hops, with a maximum of 5 hops. We expect that relations reduce the distance from the requester to the closer copy because of the high cache hit ratio, especially on the leaf routers.

The results confirm previous observations that the in-network caches provide part of the requests on behalf of the publisher. The average distance ratios of the requests in the default NDN scenario are 0.88 ($\alpha = 0.7$) and 0.77 ($\alpha = 0.9$). Because of the NDN caching mechanism, the contents are pushed closer to clients (especially the popular ones), reducing the distance that requests travel until the data is found. Overall, 23% of

Figure 5.7: Distance to contents



Source: Authors (2016).

requests do not reach the publisher when $\alpha = 0.7$ and 39.2% when $\alpha = 0.9$.

With the relations scheme, the distance ratios reduce to 0.75 ($\alpha = 0.7$) and 0.66 ($\alpha = 0.9$), as a consequence of the higher percentage of requests that do not reach the publisher. Up to 61.7% ($\alpha = 0.7$) and 69.9% ($\alpha = 0.9$) of requests find the data before reaching the publisher. The distance ratio reduction is another consequence of the higher efficiency of the caching mechanism and helps to explain the improvement of the client performance when using relations.

In summary, *relations reduce the total network traffic because of a more efficient in-network caching that causes objects to be retrieved closer by the clients.* The major part of the gains come from the popular objects, which are shared among different content variants, and leverage the NDN caching to disseminate the data.

6 FINAL CONSIDERATIONS

In this chapter we present some final considerations about this thesis and possible directions of future work. We also present the publications and other achievements attained with this work.

6.1 Conclusions

In this thesis we investigate how the method used to publish contents in an information-centric network can influence its capacity to distribute data. More specifically, we focus on the use of multiple objects to distribute the different parts that compose a single content. This strategy enables different features that improve the performance of content distribution mechanisms, such as the reuse of objects to lower data redundancy in the network. However, the use of such a strategy requires a mean for publishers to indicate which objects clients should obtain to reconstruct the original content.

Based on our observations, we formulate the following hypothesis: *If a model to establish a complex set of relations among data objects is available in an ICN, publishers can employ new methods to describe contents as sets of data objects. Such methods would result in better application performance and usage of network resources, because popular objects present a higher request rate and redundancy on published data is lowered.* Following this hypothesis, we elaborate the design of a model and a mechanism to represent a content as multiple objects linked by a set of relations. Then, we evaluate our proposal with simulated experiments based on two relevant case studies.

The work on this thesis achieved three main contributions:

1. **Proposal of a model to represent a content as multiple objects with relations among them.**
   The model enables a publisher to identify the content parts a client should obtain in order to reconstruct the original content. It is application agnostic and makes no assumption about what type of data is contained in objects. In turn, the semantic of each relation is described by publishers using key-value attributes. The simplicity of the model enables a publisher to easily describe different types of content and possibly integrate it with other tools to structure information.

2. **Analysis of the design aspects related to the implementation of the relations**

model in current ICN architectures.

We explore the following main design aspects regarding the implementation of the model in the NDN architecture: (*i*) how to store relations; (*ii*) how to distribute relations to clients; (*iii*) how to manage relations; (*iv*) authenticity of relation information; and (*v*) the format used to represent relations. We overview available implementation options and qualitatively explore their possible impact to communication performance due to relation overhead.

3. **Analysis of a case study based on multimedia content distribution to measure the impact of relations on application performance and network resource usage.**

   Multimedia content was chosen as first case study because it is a relevant application with a constantly growing traffic on the current Internet. We evaluate the distribution of a video content with multiple audio channels, using relations to separate video and audios in different objects. We compare the results from a scenario with relations against a default CCN baseline, in which all parts of the content are transmitted in a single object. The results obtained in this case study are described in a paper presented at the 2015 IFIP/IEEE IM Symposium. They show promising improvements in application performance and network resources usage when contents are modeled using relations. Compared to default CCN operations, content download times are improved 34%, publisher load in 56% and bandwidth usage in 43%.

4. **Analysis of a case study based on Web pages to measure the impact of relations in a scenario in which the overhead of the model might become a relevant issue.**

   This case study highlights the advantage of using relations to distribute versioned content in an ICN architecture. Recent studies from Wang, Krishnamurthy and Wetherall (2014) suggest that the use of micro-caching can reduce the network resources required to distribute Web content with frequent updates. We apply this insight to ICN using the concept of relations to enable the division of HTML content into micro-cacheable parts. Results from simulation experiments demonstrated that, even in a scenario with small data objects and potential overhead from the relations metadata, NDNrel can reduce download times in an average of 28% and the publisher load in 34%. These improvements are a direct result of higher hit-ratio of

in-network caches due to changes in the request distribution of data objects. These effects occur because redundant data spread among multiple objects is eliminated of the network due to the use of relations. The results obtained in this case study are described in a paper submitted to the Elsevier Computer Networks Journal and is currently undergoing revision.

6.2 Future Work

The study conducted in this thesis demonstrates the potential of the relations model to improve application and network performance in ICN environments. Based on our results, we intend to take it further by studying different issues observed in our work. Two particular issues are envisioned as direct lines of investigation:

1. **Collaborate with the efforts to develop the CCNx manifest specification.** The research community is currently exploring possible usage scenarios and alternatives to its data structures. The insights from our study can be used to guide the discussions about design decisions for the specification. In particular, the description format used for manifests can be designed to further the flexibility offered to applications when modeling contents.

2. **Explore a third case study, focused on the use of relations to publish information from IT management systems.** This case study was proposed in the context of the work on this thesis. We identified specific issues in the design of this case study that will demand additional efforts to be correctly modeled and analyzed. Consequently, we opted to leave the analysis of this case study as future work due to the limited timeframe available to complete the work on this thesis.

Both lines of work will expand the relevance of our work in the ICN research community and further our insights into the advantages of relations in different usage scenarios.

6.3 Achievements

The development of this thesis led to the publication of the following peer-reviewed paper, which describes the relations model and the case study about multimedia content

(the paper is attached to Appendix B of this document):

- CCNrel: leveraging relations among objects to improve the performance of CCN.
  Rodolfo S. Antunes, Matheus B. Lehmann, Rodrigo B. Mansilha, Christian Esteve Rothenberg, Luciano P. Gaspary, Marinho P. Barcellos. In: *14th IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*, Ottawa, Canada, 2015.

This work also led to the submission of the following journal paper, currently undergoing revision, that describes further improvements to the relations model and new results from the case study about Web pages (the paper is attached to Appendix C of this document):

- NDNrel: a mechanism based on relations among objects to improve the performance of NDN.
  Rodolfo S. Antunes, Matheus B. Lehmann, Rodrigo B. Mansilha, Luciano P. Gaspary, Marinho P. Barcellos. Currently undergoing revision from: *Elsevier Computer Networks (COMNET)*.

The work presented in this thesis proposal evolved from the development of different studies related to content distribution. These are mainly focused on the analysis of P2P systems performance and led to the coauthoring of the following publications:

- Exploring your Neighborhood: Comparing Connection Strategies in Swarming Networks through Evolving Graphs.
  Matheus B. Lehmann, **Rodolfo S. Antunes**, Marinho P. Barcellos. *13th IEEE International Conference on Peer-to-Peer Computing (P2P 2013)*, Trento, Italy, 2013.

- Em Busca do Vizinho Perfeito: Um Modelo para Estratégias de Gerência de Vizinhança em Sistemas Descentralizados de Distribuição de Conteúdo.
  Matheus B. Lehmann, **Rodolfo S. Antunes**, Marinho P. Barcellos. *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2013)*, Brasília, Brazil, 2013.

- Disconnecting to Connect: Understanding Optimistic Disconnection in BitTorrent.
  Matheus B. Lehmann, Lucas F. Muller, **Rodolfo S. Antunes**, Marinho P. Barcellos. *12th IEEE International Conference on Peer-to-Peer Computing (P2P 2012)*, Tarragona, Spain, 2012.

- Desvendando o Impacto da Desconexão Otimista no BitTorrent.
  Matheus B. Lehmann, Lucas F. Muller, **Rodolfo S. Antunes**, Marinho P. Barcellos. *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2012)*, Ouro Preto, Brazil, 2012.

- Characterizing Dissemination of Illegal Copies of Content through Monitoring of BitTorrent Networks.
  Adler H. Schmidt, **Rodolfo S. Antunes**, Marinho P. Barcellos, Luciano P. Gaspary. *13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*, Maui, USA, 2012.

- Beyond Network Simulators: Fostering Novel Distributed Applications and Protocols through Extendible Design.
  Marinho P. Barcellos, **Rodolfo S. Antunes**, Hisham H. Muhammad, Ruthiano S. Munaretti. *Journal of Network and Computer Applications*, Elsevier, v.35, pg.328-339, 2012.

- Challenging the Feasibility of Authentication Mechanisms for P2P Live Streaming.
  Rafael V. Coelho, Jonata T. Pastro, **Rodolfo S. Antunes**, Marinho P. Barcellos, Ingrid Jansch-Porto, Luciano P. Gaspary. *6th IFIP/ACM Latin America Networking Conference (LANC 2011)*, Quito, Ecuador, 2011.

- A Conservative Strategy to Protect P2P File Sharing Systems from Pollution Attacks.
  Marinho P. Barcellos, Luciano P. Gaspary, Weverton L. C. Cordeiro, **Rodolfo S. Antunes**. *Concurrency and Computation*, Wiley & Sons, v.23, pg.117-141, 2011.

The studies related to this thesis also intersected with the activities of a work group of the Rede Nacional de Ensino e Pesquisa (RNP) from November 2012 to October 2013. The work group, named GT-ICN[1] had the goal of developing a system for educational multimedia content distribution using ICN over the RNP Ipê network infrastructure[2]. The activities of this work group offered an invaluable opportunity to develop our knowledge about CCN and multimedia content distribution, which were later employed in the case study described in this proposal. The results from the work group were presented at a demo session in the 14th Workshop of RNP (WRNP 2013), a satellite event of the 31th Brazilian Symposium on Computer Networks.

[1] More information about the GT-ICN activities are available at <labcom.inf.ufrgs.br/~gticn>.
[2] <www.rnp.br/en/services/connectivity/ipe-network>

The work presented in this proposal also resulted in the undergraduate work entitled “An Object Relations Mechanism to Save Resources in the CCN Network Architecture: design, implementation and evaluation”, developed by the student Bruna Rizzardo Fiorentin as requirement for obtaining her Bs.C. in Computer Engineering in 2012. Finally, the work on this thesis also resulted in student travel grants awarded to participation in selected events, including LANC 2011, SIGCOMM 2012 and IM 2015.

REFERENCES

4WARD PROJECT. *The FP7 4WARD Project*. 2008. Available from Internet: <www.4ward-project.eu>. Accessed in: aug. 2014.

AFANASYEV, A.; MOISEENKO, I.; ZHANG, L. **ndnSIM: NDN simulator for NS-3**. 2012. Available from Internet: <named-data.net/wp-content/uploads/TRndnsim.pdf>. Accessed in: aug. 2014.

AHLGREN, B. et al. A survey of information-centric networking. **IEEE Communications Magazine**, Piscataway, v. 50, n. 7, p. 26–36, jul. 2012.

ANDROUTSELLIS-THEOTOKIS, S.; SPINELLIS, D. A survey of peer-to-peer content distribution technologies. **ACM Computing Surveys**, New York, v. 36, n. 4, p. 335–371, dec. 2004.

APACHE SOFTWARE FOUNDATION. **Apache Subversion**. 2014. Available from Internet: <subversion.apache.org>. Accessed in: sep. 2014.

BARI, F. et al. A survey of naming and routing in information centric networks. **IEEE Communications Magazine**, Piscataway, v. 50, n. 12, p. 44–53, dec. 2012.

CHA, M. et al. I tube, you tube, everybody tubes. In: SIGCOMM CONFERENCE ON INTERNET MEASUREMENT, 7., 2007, San Diego. **Proceedings...** New York: ACM Press, 2007. p. 1–14.

CHAI, W. K. et al. Cache “less for more” in information-centric networks (extended version). **Elsevier Computer Communications**, San Francisco, v. 36, n. 7, p. 758–770, abr. 2013.

CHEN, J. et al. Coexist: a hybrid approach for content oriented publish/subscribe systems. In: SIGCOMM INFORMATION-CENTRIC NETWORKING WORKSHOP, 2., 2012, Helsinki. **Proceedings...** New York: ACM Press, 2012. p. 31–36.

CHOI, J.; REAZ, A. S.; MUKHERJEE, B. A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes. **IEEE Communications Surveys Tutorials**, Piscataway, v. 14, n. 1, p. 156–169, feb. 2012.

CISCO SYSTEMS. **Cisco Visual Networking Index: Forecast and Methodology, 2014–2019**. 2016. Available from Internet: <www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf>. Accessed in: jan. 2016.

CLARK, D. D. The design philosophy of the darpa internet protocols. **ACM SIGCOMM Computer Communication Review**, New York, v. 25, n. 1, p. 102–111, jan. 1995.

DANNEWITZ, C.; D’AMBROSIO, M.; VERCELLONE, V. Hierarchical DHT-based name resolution for information-centric networks. **Elsevier Computer Communications**, San Francisco, v. 36, n. 7, p. 736–749, abr. 2013.

FARRELL, S. et al. **Naming Things with Hashes**. 2013. Available from Internet: <tools.ietf.org/html/rfc6920>. Accessed in: jul. 2014.

FOTIOU, N.; TROSSEN, D.; POLYZOS, G. Illustrating a publish-subscribe internet architecture. **Springer Telecommunication Systems**, Heidelberg, v. 51, n. 4, p. 233–245, dec. 2012.

FRICKER, C. et al. Impact of traffic mix on caching performance in a content-centric network. In: ANNUAL INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS WORKSHOPS, 31., 2012, Orlando. **Proceedings...** Washington: IEEE, 2012. p. 310–315.

GHODSI, A. et al. Information-centric networking: seeing the forest for the trees. In: WORKSHOP ON HOT TOPICS IN NETWORKS, 10., 2011, Cambridge. **Proceedings...** New York: ACM Press, 2011. p. 1:1–1:6.

ISO/IEC. **Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats**. 2014. Available from Internet: <www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=65274>. Accessed in: aug. 2015.

ISO/IEC. **Information technology – Coding of audio-visual objects – Part 12: ISO base media file format**. 2015. Available from Internet: <www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=68960>. Accessed in: aug. 2015.

JACOBSON, V. et al. VoCCN: Voice-over content-centric networks. In: WORKSHOP ON RE-ARCHITECTING THE INTERNET, 2., 2009, Rome. **Proceedings...** New York: ACM Press, 2009. p. 1–6.

JACOBSON, V. et al. Networking named content. In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES, 5., 2009, Rome. **Proceedings...** New York: ACM Press, 2009. p. 1–12.

KAUFFMANN, B.; PELTIER, J.-f.; TRUONG, P. **Final NetInf Architecture**. 2013. Available from Internet: <www.sail-project.eu/wp-content/uploads/2013/01/SAIL-DB3-v1.1-final-public.pdf>. Accessed in: sep. 2014.

KNIGHT, S. et al. The internet topology zoo. **IEEE Journal on Selected Areas in Communications**, Piscataway, v. 29, n. 9, p. 1765–1775, oct. 2011.

KOPONEN, T. et al. A data-oriented (and beyond) network architecture. In: SIGCOMM CONFERENCE, 20., 2007, Kyoto. **Proceedings...** New York: ACM Press, 2007. p. 181–192.

KULINSKI, D.; BURKE, J. **NDNVideo: Random-Access Live and Pre-Recorded Streaming Using NDN**. 2012. Available from Internet: <www.named-data.net/techreport/TR007-streaming.pdf>. Accessed in: aug. 2014.

KUROSE, J. Information-centric networking: The evolution from circuits to packets to content. **Elsevier Computer Networks**, San Francisco, v. 66, n. 1, p. 112–120, jun. 2014.

KUTSCHER, D. et al. **ICN Research Challenges**. 2016. Available from Internet: <tools.ietf.org/html/draft-irtf-icnrg-challenges-04>. Accessed in: feb. 2016.

LEDERER, S. et al. **Adaptive Video Streaming over ICN**. 2016. Available from Internet: <tools.ietf.org/html/draft-irtf-icnrg-videostreaming-07>. Accessed in: feb. 2016.

LEE, M. et al. Content discovery for information-centric networking. **Elsevier Computer Networks**, San Francisco, v. 83, n. 1, p. 1–14, jun. 2015.

MOISEENKO, I. **Fetching content in Named Data Networking with embedded manifests**. 2014. Available from Internet: <named-data.net/wp-content/uploads/2014/09/ndn-tr-25-manifest-embedding.pdf>. Accessed in: aug. 2014.

MOISEENKO, I.; STAPP, M.; ORAN, D. Communication patterns for web interaction in named data networking. In: INTERNATIONAL CONFERENCE ON INFORMATION-CENTRIC NETWORKING, 1., 2014, Paris. **Proceedings...** New York: ACM Press, 2014. p. 87–96.

MOSKO, M. **CCNx Label Forwarding**. 2014. Available from Internet: <www.ccnx.org/pubs/ccnx-mosko-labelforwarding-01.txt>. Accessed in: aug. 2015.

MOSKO, M. **CCNx Publisher Serial Versioning**. 2015. Available from Internet: <https://tools.ietf.org/html/draft-mosko-icnrg-ccnxserialversion-00>. Accessed in: aug. 2015.

MOSKO, M. et al. **CCNx Manifest Specification**. 2015. Available from Internet: <www.ccnx.org/pubs/draft-wood-icnrg-ccnxmanifests-00.html>. Accessed in: aug. 2015.

NAMED-DATA NETWORKING PROJECT. **NFD – Named Data Networkign Forwarding Daemon**. 2014. Available from Internet: <named-data.net/doc/NFD>. Accessed in: aug. 2015.

PALLIS, G.; VAKALI, A. Insight and perspectives for content delivery networks. **Communications of the ACM**, New York, v. 49, n. 1, p. 101–106, jan. 2006.

PALO ALTO RESEARCH CENTER. **CCNx**. 2015. Available from Internet: <www.ccnx.org>. Accessed in: aug. 2015.

PENTIKOUSIS, K. et al. **Information-Centric Networking: Baseline Scenarios**. 2015. Available from Internet: <https://tools.ietf.org/html/rfc7476>. Accessed in: aug. 2015.

PENTIKOUSIS, K. et al. **Information-centric Networking: Evaluation Methodology**. 2015. Available from Internet: <tools.ietf.org/html/draft-irtf-icnrg-evaluation-methodology-03>. Accessed in: nov. 2015.

PERINO, D.; VARVELLO, M. A reality check for content centric networking. In: SIGCOMM INFORMATION-CENTRIC NETWORKING WORKSHOP, 1., 2011, Toronto. **Proceedings...** New York: ACM Press, 2011. p. 44–49.

PERINO, D.; VARVELLO, M.; PUTTASWAMY, K. P. N. ICN-RE: Redundancy elimination for information-centric networking. In: SIGCOMM INFORMATION-CENTRIC NETWORKING WORKSHOP, 2., 2012, Helsinki. **Proceedings...** New York: ACM Press, 2012. p. 91–96.

PSIRP PROJECT. **PSIRP**. 2011. Available from Internet: <www.psirp.org>. Accessed in: aug. 2014.

PURSUIT PROJECT. **PURSUIT**. 2013. Available from Internet: <www.fp7-pursuit.eu>. Accessed in: aug. 2014.

RAJAHALME, J. et al. On name-based inter-domain routing. **Elsevier Computer Networks**, San Francisco, v. 55, n. 4, p. 975–986, mar. 2011.

ROSSINI, G. et al. Multi-terabyte and multi-gbps information centric routers. In: ANNUAL INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 33., 2014, Toronto. **Proceedings...** Washington: IEEE, 2014. p. 181–189.

ROSSINI, G.; ROSSI, D. Evaluating CCN multi-path interest forwarding strategies. **Elsevier Computer Communications**, San Francisco, v. 36, n. 7, p. 771–778, abr. 2013.

SAIL PROJECT. **The Network of Information: Architecture and Applications**. 2011. Available from Internet: <www.sail-project.eu/wp-content/uploads/2011/08/SAIL_DB1_v1_0_final-Public.pdf>. Accessed in: aug. 2014.

SAIL PROJECT. **Scalable and Adaptive Internet Solutions**. 2013. Available from Internet: <www.sail-project.eu>. Accessed in: aug. 2014.

SAIL PROJECT. **Network of Information**. 2014. Available from Internet: <www.netinf.org>. Accessed in: sep. 2014.

SOLLINS, K. R. Pervasive persistent identification for information centric networking. In: SIGCOMM INFORMATION-CENTRIC NETWORKING WORKSHOP, 2., 2012, Helsinki. **Proceedings...** New York: ACM Press, 2012. p. 1–6.

TROSSEN, D.; PARISIS, G. Designing and realizing an information-centric internet. **IEEE Communications Magazine**, Piscataway, v. 50, n. 7, p. 60–67, jul. 2012.

VARVELLO, M.; PERINO, D.; ESTEBAN, J. Caesar: A content router for high speed forwarding. In: SIGCOMM INFORMATION-CENTRIC NETWORKING WORKSHOP, 2., 2012, Helsinki. **Proceedings...** New York: ACM Press, 2012. p. 73–78.

WÄHLISCH, M.; SCHMIDT, T. C.; VAHLENKAMP, M. Lessons from the past: Why data-driven states harm future information-centric networking. In: NETWORKING CONFERENCE, 12., 2013, New York. **Proceedings...** Laxenburg: IFIP, 2013. p. 60–67.

WANG, X. S.; KRISHNAMURTHY, A.; WETHERALL, D. How much can we micro-cache web pages? In: INTERNET MEASUREMENT CONFERENCE, 12., 2014, Vancouver. **Proceedings...** New York: ACM Press, 2014. p. 249–256.

XYLOMENOS, G. et al. Caching and mobility support in a publish-subscribe internet architecture. **IEEE Communications Magazine**, Piscataway, v. 50, n. 7, p. 52–58, jul. 2012.

XYLOMENOS, G. et al. A survey of information-centric networking research. **IEEE Communications Surveys & Tutorials**, Piscataway, v. 16, n. 2, p. 1024–1049, may 2014.

ZHANG, G.; LI, Y.; LIN, T. Caching in information centric networking: a survey. **Elsevier Computer Networks**, San Francisco, v. 57, n. 16, p. 3128–3141, nov. 2013.

ZHANG, L. et al. **Named Data Networking Project**. 2010. Available from Internet: <named-data.net/wp-content/uploads/TR001ndn-proj.pdf>. Accessed in: aug. 2014.

APPENDIX A — RESUMO EXPANDIDO

Neste capítulo apresenta-se um resumo expandido do trabalho desenvolvido. Serão abordados (*i*) o contexto do trabalho, (*ii*) o problema investigado e a hipótese formulada, e (*iii*) os principais resultados obtidos.

A.1 Contexto

Os protocolos atualmente utilizados na Internet foram projetados para permitir o acesso de clientes remotos a recursos computacionais caros disponibilizados por um pequeno grupo de instituições (CLARK, 1995). Desde então, a Internet e seus cenários de uso evoluíram além das expectativas de seus criadores. Atualmente, ela é uma rede mundial utilizada principalmente para a distribuição de uma ampla variedade de conteúdos digitais. Por exemplo, aplicações de distribuição de vídeo sozinhas geraram 66% do tráfego de rede global nos últimos anos (CISCO, 2016).

A pilha de protocolos original da Internet requer funcionalidades adicionais para lidar com as demandas e a escala dos sistemas de distribuição de conteúdo atuais. Tal fato é evidenciado pelo advento de mecanismos em nível de aplicação voltados para a distribuição de conteúdo em larga escala, tais como sistemas Par-a-Par (P2P, do inglês *Peer-to-Peer*) (ANDROUTSELLIS-THEOTOKIS; SPINELLIS, 2004) e redes de distribuição de conteúdo (CDNs, do inglês *content distribution networks*). Em particular, CDN é o mecanismo de distribuição de conteúdo em escala global mais amplamente utilizado atualmente: 36% do tráfego de rede global tem como origem alguma CDN (CISCO, 2016). Porém, tanto sistemas P2P quanto CDNs também apresentam desvantagens. Por exemplo, sistemas P2P são conhecidos por gerar grandes volumes de tráfego entre domínios devido a falta de conhecimento sobre a rede física utilizada pela aplicação. Por sua vez, infraestruturas CDN rapidamente se tornam caras e complexas de gerenciar quando utilizadas para distribuir um extensivo catálogo de conteúdos populares.

Os fatos apresentados acima combinados com os esforços para desenvolver arquiteturas de rede para a Internet do Futuro levaram à concepção das *Redes Orientadas a Conteúdo* (ICN, do inglês *Information-Centric Networks*) (AHLGREN et al., 2012). Originalmente, o paradigma ICN propõe a mudança da atual camada de rede *orientada a hosts* para uma *orientada a dados*, consequentemente mudando elementos fundamentais da comunicação. *Nomes* únicos identificam cada um dos objetos disponíveis na rede e

são utilizados como principal informação de roteamento por mecanismos de rede. Clientes enviam uma requisição para um nome específico à rede, a qual será responsável por rotear a requisição para um *publicador* e, após, os dados de volta para o cliente. ICN também propõe o uso extensivo de *cache* para melhorar o desempenho da distribuição de conteúdo.

O foco de ICN está em serviços baseados na distribuição de conteúdo, mas uma arquitetura de rede voltada para a Internet precisa suportar todos os tipos de aplicações que se comunicam em uma escala global. Tais incluem aquelas que utilizam um modelo centrado em conexão, que se baseia na conexão entre dois *hosts*. Trabalhos iniciais sobre ICN, tal como o de Jacobson et al. (2009a), mostram que é possível portar aplicações centradas em conexão para o paradigma ICN. Porém, tal operação pode resultar em perdas de desempenho em função do maior número de mensagens necessárias para a implementação de protocolos orientados a conexão em uma arquitetura ICN. Estudos recentes sugerem que um cenário mais plausível seja a coexistência de ambas infraestruturas baseadas em IP e ICN (CHEN et al., 2012).

Apesar do debate sobre a aplicabilidade de ICN, vários estudos mostram que infraestruturas ICN têm o potencial de aprimorar os atuais sistemas de distribuição de conteúdo (XYLOMENOS et al., 2014). Estes estudos se focam no desenvolvimento de aspectos arquiteturais de ICN para melhorar a performance da rede, por exemplo estratégias de roteamento e cache. Entretanto, Poucos estudos focam no impacto causado pelas características dos conteúdos distribuídos utilizando infraestruturas ICN. Mais especificamente, o conceito de objetos de dados unicamente identificados podem ser utilizados de modos distintos por diferentes tipos de conteúdos, resultando em desempenhos distintos da rede e aplicações.

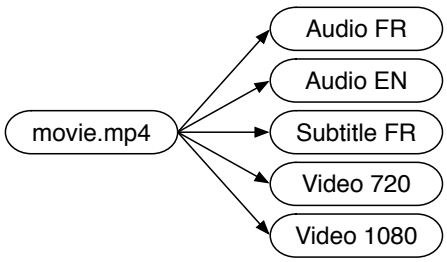
Nesta tese, o principal objetivo é explorar o uso de múltiplos objetos para a distribuição de conteúdos. Mais especificamente, nós permitimos que um publicador descreva um conteúdo através do estabelecimento de *relações* entre um conjunto de objetos individuais. Em resumo, uma relação é um *link* entre dois objetos indicando que os dados de um complementam de alguma forma os dados do outro. Seguindo esse conceito, um conteúdo se torna uma coleção de múltiplos objetos e as respectivas relações que caracterizam as interações entre seus dados. Nós empregamos estas definições para formular um modelo que publicadores podem utilizar para distribuir conteúdos através de conjuntos de objetos relacionados. Nosso estudo analisa os aspectos de implementação desses conceitos em uma arquitetura ICN e avalia seu impacto no desempenho da rede e aplicações.

A.2 Problema e Hipótese

Um *objeto* é a unidade básica de distribuição de dados e é unicamente identificado de acordo com um esquema de nomeação definido pela arquitetura ICN (JACOBSON et al., 2009b). O *nome* único de um objeto é a principal informação utilizada por clientes para requisitar dados da rede. Arquiteturas ICN podem ainda dividir um objetos em múltiplos *chunks*. Tal divisão é agnóstica à estrutura do conteúdo e usualmente considera aspectos relativos à infraestrutura de rede subjacente. Ela é utilizada para facilitar a transmissão de dados e a implementação de mecanismos de controle de fluxo. Nosso estudo não tem foco nos *chunks*; estes são considerados um aspecto de implementação do nosso mecanismo quando requerido pela arquitetura. Ao invés, nosso trabalho considera um objeto como a unidade básica de dados que é unicamente identifica da na rede.

Um método simples de empregar o conceito de objeto para publicar conteúdos é armazenar todos os dados em um único objeto. Em outras palavras, um objeto encapsula todos os componentes necessários que o cliente precisa obter para utilizar o conteúdo. Entretanto, objetos conteúdos atualmente disponíveis na Internet são melhor descritos como coleções de elementos distintos que podem ser separados e distribuídos como objetos de dados individuais. Um exemplo é um conteúdo multimídia como o apresentado na Figura A.1. Como ilustrado, este tipo de conteúdo apresenta componentes distintos, tais como canais de vídeo em diferentes qualidades, canais de áudio em diferentes idiomas, e legendas.

Figura A.1: Conteúdo de vídeo separado em múltiplos objetos



Fonte: Autores (2016).

Algumas aplicações atuais empregam múltiplos objetos para distribuir um conteúdo. HTML é um dos exemplos mais notáveis. Entretanto, tais aplicações empregam modelos de descrição de conteúdos projetados para atender a requisitos específicos. Em nosso trabalho, estamos interessados em um modelo genérico que permita descrever diferentes tipos de conteúdos através de múltiplos objetos, de modo que qualquer aplicação

em um ambiente ICN possa utilizá-lo. O amplo uso de relações para distribuir conteúdos apresenta duas vantagens principais para a arquitetura de rede. Primeiro, ela permite que clientes selecionem apenas as partes necessárias para reproduzir o conteúdo de acordo com suas preferências pessoais, tais como um idioma de áudio em particular em um filme. Como resultado, haverá uma redução no tráfego necessário para distribuir o conteúdo. A segunda vantagem é que partes já disponíveis na rede podem ser reutilizadas para a publicação de novos conteúdos, como por exemplo uma folha de estilos empregada em múltiplas páginas Web. Como resultado, a redundância dos dados publicados é diminuída.

Um conteúdo distribuído através de múltiplos objetos requer um meio para identificar as partes necessárias e seus respectivos identificadores. Tal *descrição* de um conteúdo deve estar disponível através de metadados que possam ser acessados e analisados por qualquer cliente que deseje recuperá-lo. Em nosso trabalho, tais metadados descrevem as várias *relações* estabelecidas entre os objetos individuais que constituem o conteúdo. No exemplo da Figura A.1, cada objeto conteria uma das partes individuais do conteúdo, isto é, um canal de áudio ou vídeo, ou uma legenda. Por sua vez, cada relação estabelece como os dados de um objeto agem sobre os dados de outro, resultando assim no conteúdo original. Por exemplo, uma relação entre um objeto contendo um canal de áudio e outro com uma legenda indica que o último contém a transcrição do audio do primeiro. O tipo de dados contidos em cada objeto e a semântica de suas respectivas relações depende diretamente do tipo de conteúdo modelado com estes conceitos.

Arquiteturas ICN atuais apresentam dois métodos principais que permitem a publicação da estrutura de relações de um conteúdo: *nomes hierárquicos* e *objetos de links*. Nomes hierárquicos são um tipo de identificador de objeto empregado por algumas arquiteturas ICN (JACOBSON et al., 2009b). Eles são similares a URLs de sistemas HTTP atuais. Cada elemento de um nome hierárquico tem relações implícitas com componentes pais e filhos, os quais permitem a descrição de um conteúdo baseados em múltiplos objetos. Por exemplo, o nome “</ufrgs/ppgc/video.mp4/video_h264/720p>” permite ao cliente inferir que o conteúdo “video.mp4” tem um canal de vídeo codificado no formato h.264 com uma qualidade de 720p. Apesar de nomes hierárquicos permitirem um método direto para implementar relações, duas limitações devem ser consideradas. Primeiro, todos objetos que pertencem a um conteúdo devem ser publicados com identificadores contidos na mesma estrutura hierárquica. Consequentemente, não é possível reutilizar objetos de diferentes conteúdos sem adicionar complexidade ao esquema de nomeação. Segundo,

mudanças na estrutura do conteúdo devem ser refletidas nos nomes das respectivas partes. Por consequência, nomes de objetos mudariam depois de sua publicação, criando instabilidades nos mecanismos de roteamento da rede.

Objetos de *links* são uma funcionalidade implementada na arquitetura NetInf (SAIL PROJECT, 2014). Em suma, um objeto de *link* contém uma lista de nomes para outros objetos que podem estar relacionados de algum modo. Uma estrutura hierárquica de objetos de *links* com metadados adicionais pode ser utilizada para descrever um conteúdo baseado em múltiplos objetos. Entretanto, um conteúdo com uma estrutura complexa resultará em um grande número de objetos de *links* porque estes podem manter apenas a informação sobre um nível de hierarquia. Consequentemente, clientes devem obter um grande número de objetos de *links* antes de requisitar o conteúdo propriamente dito.

Em relação ao objetivo de evitar a redundância de dados, Perino, Varvello and Puttaswamy (2012) propõem o uso de algoritmos de identificação de redundância para encontrar *chunks* idênticos em nível de bit. Tais *chunks* redundantes são isolados e publicados uma única vez para evitar a duplicação de dados e a redução do desempenho das caches. Resultados deste estudo mostram o potencial de ganho de desempenho ao se isolar dados duplicados em nível de bit e publicá-los de forma compartilhada. Entretanto, a vantagem deste método é limitada pelo uso de algoritmos automatizados para identificação de redundância que explora conteúdos de um único publicador. Tal método não considera a possiblidade de vários publicadores distribuírem conteúdos compartilhados entre eles. Além disso, o algoritmo proposto utiliza apenas comparações em nível de bit para identificar a redundância. Ele não considera casos onde a redundância vai além desse tipo de comparação, tal como em um vídeo com o mesmo conteúdo mas com configurações de codificação diferentes.

Com base nas observações apresentadas até o momento, concluímos que:

1. Arquiteturas ICN não possuem um modelo geral que permite a representação de um conteúdo através de uma estrutura complexa com múltiplos objetos de dados;

2. Não existem estudos sobre o comportamento da rede e aplicações quando diferentes tipos de conteúdo são distribuídos utilizando múltiplos objetos.

A partir destas conclusões, formulamos a seguinte hipótese:

Se um modelo para estabelecer um conjunto complexo de relações entre objetos estiver disponível em uma ICN, publicadores podem aplicar novos métodos para descrever conteúdos através de conjuntos de objetos de dados. Tais métodos resultariam em melhor performance aplicação e uso dos recursos da rede, porque objetos populares teriam uma maior taxa de requisições e a redundância dos dados publicados seria menor.

Um *modelo de relações* pode ser utilizado para implementar soluções para problemas conhecidos da área de ICN, tais como o versionamento de conteúdos ou a coleta de dados de monitoramento de rede, como descrito em Kutscher et al. (2016). Então, nesta tese, nosso objetivo é apresentar um modelo para estabelecer relações entre objetos para ICN e explorar a aplicabilidade deste modelo em cenários relevantes para Redes Orientadas a Conteúdo.

A.3 Objetivos e Contribuições

O estudo proposto neste trabalho possui três objetivos principais: (*i*) apresentar um modelo que permite a publicadores em uma ICN descrever um conteúdo através de um conjunto de objetos de dados ligados por relações arbitrárias; (*ii*) analisar os aspectos de projeto da implementação do modelo em uma arquitetura ICN relevante; e (*iii*) avaliar o impacto do uso de relações nas aplicações e na infraestrutura ICN. Nós atingimos estes objetivos, respectivamente, através de três contribuições principais de nosso trabalho, descritas a seguir.

A primeira contribuição é a proposta de um modelo para estabelecer relações entre objetos. Projetamos este modelo para *ser agnóstico às aplicações*, *suportar diferentes estruturas de relações*, e *ser retrocompatível com especificações ICN atuais*. O modelo é agnóstico às aplicações e permite que diferentes sistemas compartilhem um conteúdo sem a necessidade de converter relações entre diferentes formatos. Por exemplo, uma aplicação não precisa suportar várias especificações de contêineres multimídia (tais como MKV ou MP4) para que seja possível identificar o canal de áudio que pertence a um filme. Aplicações precisam apenas ter acesso à descrição da semântica das relações para interpretar os dados de cada um dos objetos que compõem o conteúdo. Para permitir tal descrição, o modelo proposto permite o uso de atributos, que são pares chave-valor que

podem ser adicionados à estrutura de relações. Por sua vez, o modelo é projetado para suportar diferentes estruturas de relações para evitar restringir as aplicações no momento de projetar a distribuição de dados entre diferentes objetos. Os exemplos apresentados em nosso trabalho se focam em estruturas hierárquicas porque estas são encontradas em aplicações comuns e relevantes, tais como páginas Web e conteúdo multimídia. Porém, outras estruturas podem ser projetadas dependendo das características do conteúdo, por exemplo, um grafo que inclui ciclos. Finalmente, o modelo é projetado com foco na retrocompatibilidade com arquiteturas ICN atuais, a fim de evitar possíveis impactos em suas especificações. Como uma contribuição adicional em relação ao modelo, apresentamos uma análise matemática dos possíveis ganhos relacionados ao espaço necessário para manter um catálogo de conteúdo utilizando relações. Tal análise demonstra os efeitos do uso de relações na eliminação da redundância de dados da rede.

A segunda contribuição do nosso trabalho é a análise dos principais aspectos de projeto relacionados à implementação do modelo na arquitetura ICN *Named-Data Networking* (NDN) (ZHANG et al., 2010). Analisamos os principais requisitos e como atendê-los com as funcionalidades presentes no NDN. Nossa análise é focada no uso de manifestos, os quais são meta-objetos que descrevem as relações entre os objetos que possuem os dados dos conteúdos. Discutimos os *trade-offs* relacionados ao armazenamento, organização, gerenciamento, autenticidade e representação das relações, levando em consideração o desempenho da rede e das aplicações.

Finalmente, a terceira contribuição desta tese é avaliar o impacto do uso do modelo de relações através de dois cenários relevantes de distribuição de conteúdos, sendo estes: *conteúdos multimídia* e *páginas Web*. Cada um destes cenários é avaliado em um estudo de caso individual. O estudo de caso multimídia avalia o comportamento de relações com objetos que possuem um grande volume de dados. Múltiplos objetos permite que clientes obtenham apenas as partes necessárias do conteúdo para reproduzi-lo de acordo com preferências específicas de qualidade e região. Por sua vez, o estudo de caso de páginas Web explora os ganhos obtidos em um cenário onde o uso de relações gera uma sobrecarga não negligível. Neste estudo de caso, relações permitem a publicação de páginas Web através de composição, a qual permite o reuso de objetos já existentes para evitar redundância dos dados. Como objetos HTML possuem um tamanho pequeno em média, é possível que a sobrecarga resultante dos metadados das relações possa impactar negativamente o desempenho das aplicações e da rede. Consequentemente, este cenário avalia em mais detalhes os ganhos obtidos em comparação à sobrecarga adicionada em função do uso de

relações. Os principais resultados obtidos nestes estudos de caso são descritos a seguir.

A.4 Principais Resultados

Ambos estudos de caso sobre conteúdo multimídia e páginas Web foram avaliados através de simulações utilizando uma extensão do ambiente NDNSim (AFANASYEV; MOISEENKO; ZHANG, 2012). Em ambos os casos foram avaliados dois cenários principais. O primeiro, denominado *NDN padrão*, realiza a distribuição dos conteúdos utilizando apenas mecanismos básicos do NDN, sem considerar relações entre objetos. O segundo cenário, denominado *NDNrel*, emprega o modelo de relações proposto para dividir os conteúdo em múltiplos objetos para distribuição aos clientes. Os diferentes parâmetros utilizados na simulação consideram valores encontrados na literatura relevante sobre as aplicações utilizadas nos estudos de caso, além de traços coletados sobre o uso destas.

No estudo de caso sobre conteúdo multimídia, observa-se que o uso de relações reduz o tempo de download dos clientes em média 32%, a carga do publicador em média 51%, e o tráfego total da rede em média 38%. Tais ganhos de desempenho ocorrem em função do efeito do uso de relações sobre os mecanismos de distribuição de conteúdo utilizados pela arquitetura NDN. Mais especificamente, observa-se que as requisições dos clientes são atendidas por roteadores mais próximos dos clientes, pois o número de saltos percorridos por estas é reduzido em média 30% com o uso de relações. Tal redução ocorre pois as caches dos roteadores apresentam uma maior taxa de acertos, servindo em média 22% mais requisições em comparação com o caso padrão. Nossos resultados mostram que tais ganhos ocorrem pois o uso de relações altera o modo como as requisições dos clientes são distribuídas entre os objetos publicados na rede. Isto é, requisições que são espalhadas entre dados replicados em diferentes objetos no caso padrão passam a ser concentradas em um único objeto compartilhado por diferentes variantes de um conteúdo quando utiliza-se o NDNrel. Como resultado, tais objetos apresentam uma maior taxa de acerto nas caches e uma maior presença na rede. Adicionalmente, o espaço total necessário para manter o catálogo de conteúdos nos publicadores e nas caches é menor.

Por sua vez, no estudo de caso sobre páginas Web observa-se que o uso de relações reduz o tempo de download dos clientes em média 25%, a carga do publicador em média 32%, e o tráfego de rede em média 28%. Adicionalmente, observa-se que a sobrecarga gerada pelo uso de relações no tempo de download equivale a, em média, 10% do tempo de download dos clientes. Ou seja, uma sobrecarga média de 10% utilizando relações resulta

em uma redução de 25% no tempo de download dos clientes em comparação com o caso padrão. Do mesmo modo, observa-se que a sobrecarga gerada pelas relações é equivalente a 4% do tráfego total da rede. Ou seja, uma sobrecarga de 4% utilizando relações resulta em uma redução de 28% no tráfego total em comparação com o caso padrão. Os demais resultados da simulação mostram novamente que tais ganhos ocorrem pois o uso de relações altera o modo como as requisições dos clientes são distribuídas entre os objetos publicados na rede. Mais especificamente, o uso de composição para distribuir as páginas Web amplia a taxa de requisição dos objetos compartilhados entre as diferentes variantes do conteúdo. Como resultado, os objetos mais populares na rede concentram aproximadamente 30% mais requisições do que no caso padrão. Como resultado, as caches na rede apresentam uma maior taxa de acertos para tais objetos, servindo aproximadamente 59% mais requisições do que no caso padrão. Consequentemente, as requisições dos clientes serão atendidas por roteadores mais próximos, reduzindo o número de saltos percorridos em torno de 15%.

Tais resultados demonstram o potencial de ganhos que podem ser obtidos com o uso de relações. Tal potencial pode ser explorado através de diferentes caminhos. Dentre estes, destaca-se a colaboração com esforços de pesquisa de diferentes projetos de ICN para geração de uma especificação de manifestos que seja compatível com o modelo de relações proposto, além de outros usos. Além disso, é interessante a realização de estudos com diferentes aplicações, principalmente aquelas que apresentem conteúdos que mostrem o potencial do modelo proposto para representar estruturas complexas de objetos.

APPENDIX B — PAPER PUBLISHED AT IFIP/IEEE IM 2015

- **Title**: *CCNrel: leveraging relations among objects to improve the performance of CCN*
- **Conference**: IFIP/IEEE Integrated Network Management Symposium (IM 2015)
- **Type**: Main track (full-paper)
- **Qualis**: B1
- **Date**: May 11-15, 2015
- **Held at**: Ottawa, CA
- **Digital Object Identier (DOI)**: <http://dx.doi.org/10.1109/INM.2015.7140293>

In this paper, we present CCNrel as a backward-compatible mechanism for CCN that enables publishers to distribute contents as related objects. Differently from existing relation mechanisms, which focus on one type of content and are application-specific, CCNrel is generic and enables the use of relations in both current and novel application domains. First, we discuss CCNrel fundamental concepts and main design aspects. Next, we use CCNrel as foundation for a case study of data redundancy elimination in multimedia content distribution. Through extensive simulation work we evaluate the potential benefits of leveraging relations measured by the clients experience and overall network efficiency. Results of the presented use case show that, on average and when compared to default CCN operations, content download times are improved in 34%, publishers load in 56%, and the network bandwidth usage in 43%.

CCNrel: leveraging relations among objects to improve the performance of CCN

Rodolfo S. Antunes*, Matheus B. Lehmann*, Rodrigo B. Mansilha*,
Christian Esteve Rothenberg†, Luciano P. Gaspary*, Marinho P. Barcellos*
* Institute of Informatics, Federal University of Rio Grande do Sul, RS, Brazil
† Faculty of Electrical and Computer Engineering, University of Campinas, SP, Brazil
{rsantunes,mblehmann,rbmansilha}@inf.ufrgs.br
chesteve@dca.fee.unicamp.br, {paschoal,marinho}@inf.ufrgs.br

***Abstract*—Content-Centric Networking (CCN) is a promising architectural approach that focuses on the efficient distribution of uniquely named data objects. A piece of content is represented by a single object in the network and is divided into multiple chunks which can be uniquely named and cached by network nodes. However, in its current form, the potential of CCN is not fully exploited due to the lack of common means to express and take advantage from possible relations that may exist among different objects. Our work explores the simple yet effective idea of supporting and exploiting such relations in CCN. In this paper, we present CCNrel as a backward-compatible mechanism for CCN that enables publishers to distribute contents as related objects. Differently from existing relation mechanisms, which focus on one type of content and are application-specific, CCNrel is generic and enables the use of relations in both current and novel application domains. First, we discuss CCNrel fundamental concepts and main design aspects. Next, we use CCNrel as foundation for a case study of data redundancy elimination in multimedia content distribution. Through extensive simulation work we evaluate the potential benefits of leveraging relations measured by the clients experience and overall network efficiency. Results of the presented use case show that, on average and when compared to default CCN operations, content download times are improved in 34%, publishers load in 56%, and the network bandwidth usage in 43%.**

I. Introduction

The Content-Centric Networking (CCN) model [1], [2] is based on a network architecture where a data object is a self-contained entity that uniquely binds a name to a set of bits. Each object in the network represents a complete content (e.g. a document or multimedia file). This simple representation allows chunks of content to be individually named and cached by the network nodes. However, as we argue in our paper, because the current CCN lacks means to relate content objects at arbitrary granularities, it misses opportunities for further improvement gains in terms of efficiency and performance, from client, server and network perspectives.

Content can be modeled as a set of multiple objects provided *relations* are employed to define links among the different objects. Many types of content can employ relations. Some of them are more obvious, such as multimedia, while others are less intuitive but can follow the same principle, such as log files. Taking as an example a multimedia content comprising two objects: a video (common to every version), and an audio, selected between multiple options varying according to a property (e.g. language). With relations, each audio channel can be published as an independent object, separately from the video channel. Relations can then be used to link the audio channels to the video, enabling applications to identify the available audio options. A log file, on its turn, is a sequentially versioned content and may also benefit from relations. Updates in the log may be stored in differential objects related to each other according to their version. A client can obtain a given log versions by following the objects relations.

In this paper, we propose CCNrel, a backward-compatible extension to the CCN architecture that enables publishers to distribute contents as related objects. Our goal is to explore the concept of relations among objects and how it can be used to model contents in CCN. The concept of relations behind CCNrel is flexible because it does not impose any restriction on the way publishers can model contents (Sec. II). To demonstrate the potential of CCNrel, we evaluate it in a case study (Sec. III) around multimedia content distribution. As expected, the evaluation results (Sec. IV) show that the achieved data redundancy elimination lead to promising increases in user and network performance. While some state of the art solutions employ a similar concept at the application level, their main drawback is the binding to specific types of contents and applications, whereas CCNrel enables relations to be used in arbitrary novel ways beyond those already explored. When compared to related work in Information-Centric Networking (ICN) research (Sec. V), CCNrel stands out as the first proposal tailored to CCN and presenting a backward-compatible approach that does not require modification of CCN routers.

CCN is a novel proposal for computer networks that brings an myriad of new issues related to the operation and management of networks. Such issues should be thoroughly investigated in the development stage of CCN, allowing the incorporation of required methods and mechanisms for proper management. In the above context, this paper presents two important contributions. First, we propose and explore the design aspects of a mechanism to enable the use of object relations to model contents in CCN. Second, we present quantitative results from a case study that employs CCNrel for redundancy elimination, a technique that can potentially improve the network network performance and the quality experienced by the end clients.

II. CCNrel: Relations Mechanism for CCN

In this section, we first discuss the fundamental concept of relations employed in CCNrel, our backward-compatible

978-3-901882-76-0 @2015 IFIP

relations mechanism proposal for CCN. Next, we exemplify the use of relations to model contents. Finally, we explore the main aspects considered in the design of the relation mechanism for CCN.

A. Fundamental Concepts

In our proposal, an *object* represents an individual piece of information on the network. A *relation* from an object a to another, b, is a link indicating that the data of a can be complemented by the data of b. Each relation has one or more *attributes*, which are key-value pairs with additional information that describe the relation.

The semantic of a relation is opaque to the network and is known by the applications that employ the mechanism. We make this design decision to keep the network core simple. Semantic inferences over relations and similar tasks are left to the application level, allowing the network to focus on the storage and transmission.

B. Modeling Contents with Relations

The concept of relations enables new ways to model a content into objects published in the network. We focus on three particular modeling examples: (*i*) *decomposition:* publish a content as multiple, unique objects; (*ii*) *composition:* share previously published objects to create a new content; and (*iii*) *versioning:* modify an object data without creating an entirely new object copy. To demonstrate these models we present examples that demonstrate the advantages of relations, namely: a multimedia content, user generated playlists, and a cumulative log.

Multimedia content. This example demonstrates how a content can be decomposed with the use of relations. Decomposition allows redundant parts from a content to be published as a single object, eliminating data redundancy in the network. Figure 1 depicts a multimedia content with multiple audio, subtitle and video quality options. Relations allow each option to be published as an individual object, which is related to the main content. The attributes of each relation can be used to indicate the type and other characteristics of options. A client can access the relations of the "Movie" object to identify the available audio, video and subtitle options and then download only the objects of interest.

Playlist. This example shows how a client can create a new content based on already published objects with the use of relations. The goal in this case is to avoid the republication of data already existing in the network, therefore reducing redundancy. Figure 2 depicts an example in which clients create and publish playlists with their favorite songs. This is done with relations by linking the already published objects containing the audio data with another that represents the playlist. The attributes, in this case, can be used to indicate the track ordering. Clients interested in the playlist just have to access the related objects to obtain the actual songs. In the example, the "Song D" is added to all three playlists, but only one copy of the object is necessary because relations are used to build the playlists.

Log file. The third example shows how a versioned content can be modeled with the use of relations. The goal is to enable a content to be updated without the need to republish a new complete object version. Figure 3 depicts a sequentially versioned log, updated with new entries according to a division criteria (e.g. all entries from a specific hour). Each version has two relations: one to the previous version and the other, to the object containing the new entries (the delta from the previous version). This division successively applies to prior versions of the log and allows the user to obtain the entries up to a specific version or only a set of entries from a given delta.

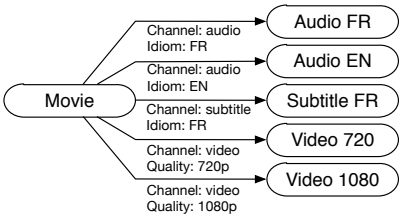


Figure 1. Multimedia modeling example

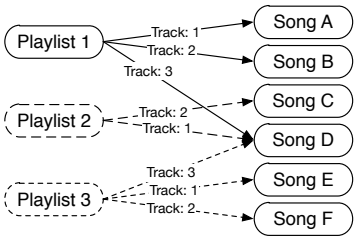


Figure 2. Playlist modeling example

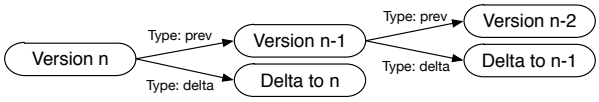


Figure 3. Log file modeling example

To enable the widespread usage of relations to model contents in CCN, the above concepts should be made available as an API enabling publishing and retrieval of relations. The API, in turn, access the functionalities of a backward-compatible mechanism implemented on top of the CCN client library. Next, we consider the key aspects of a relations mechanism for CCN.

C. Mechanism Design Aspects

The design of a relations mechanism for CCN must consider four main aspects: (*i*) where relations should be stored; (*ii*) how a client can retrieve relations; (*iii*) how a publisher can manage relations; and (*iv*) what format to use when describing relations. Each of these aspects is explored next.

Relations storage. We first consider how to make relations publicly available in the network. A trivial method would be to include relations in the object data. Such a method, however, would suffer from the following limitations: (*i*) clients would have to get the object data before they could process relations and request other objects, reducing the effectiveness of relations; and (*ii*) it would hinder the possibility of updating

relations because it would require a new object to be published (as discussed later in this section).

To avoid these drawbacks, relations can be stored in a *manifest*, separately from object data. Two different methods can be used to publish the manifest: (*i*) in an individual object; or (*ii*) as a metadata from the related objects. The first method has a simple implementation and only requires the client to know the name of the manifest object. The second method, in turn, requires a mechanism to publish object metadata. Such a mechanism is provided by CCN: it allows publishers to distribute additional information of a given content (i.e. the thumbnail of a video). Published metadata can be accessed using the special name component *%C1.META* and a metadata identifier. To employ this mechanism, the manifest can be published as a metadata identified by the name *<object name>/%C1.META/relations*. Because the first method to store the manifest has no additional requirements from the architecture and CCN implements the mechanism needed by the second method, both can be used by CCNrel.

Relations retrieval. The relations of a content can be stored in *one* or *multiple* manifests. In the first method, one manifest stores the complete relation structure of a content (e.g., with multiple levels and indirect links). In this case, related objects store a pointer to the manifest, which can be a metadata or an individual object. The use of one manifest allows a client to retrieve the complete structure with at most two requests. In the second method, each object maintains only its direct relations in a manifest stored as metadata. Since there are multiple manifests, a client must recursively obtain the distributed relations from the related objects to recover the complete content structure.

Ideally, both methods of relation storage should be supported. The use of one or multiple manifests depends on the complexity of the content relations structure. Multiple manifests are advantageous when partial retrieval of the content structure is desirable (e.g. a specific page from a complex Web site). In this case, the use of a central manifest requires the client to parse a potentially large structure only to find a specific group of relations. However, in cases where a complex relation structure must be recovered multiple manifests introduce a higher overhead for content retrieval because relations have to be obtained with recursive requests. In these cases, the use of one manifest introduces smaller network overhead.

Management operations. The mechanism must allow relations to be created, edited, and removed. Creating relations requires only the publication of one or more manifests for objects –depending on the storage method employed to maintain relations structure. Edition and removal, in turn, require manifests to be updated after publication. These operations, thus, require the use of a versioning mechanism for their implementation. To circumvent this issue, we use the default versioning mechanism of the CCN architecture. Albeit simple compared to the issue of object versioning in CCN [3], this mechanism suits our requirements for maintaining manifests.

With respect to the scope of the operations, it depends on the method used to publish relations. When relations are published as an object metadata, the mechanism limits the use of the above operations to the original publisher of the object. This behavior is compatible to the one imposed by the CCN metadata mechanism: only the creator of the original object can publish a metadata about it. If relations are published as an individual object, any user can create and maintain a manifest, possibly referencing objects from various publishers. The use of individual objects, thus, increases the mechanism flexibility. We do not consider the case of one manifest maintained by multiple publishers, because such would require the implementation of concurrent operations. This would increase the complexity of the mechanism but nonetheless will be investigated in future work.

Common description format. To support the desired concept of relations, the employed manifest should enable: (*i*) the representation of a hierarchical structure for the description of complex contents with multiple levels of relations; and (*ii*) allow attributes to be included in the description of relations. Any description format that satisfies these requirements can be used to store relations. More specifically, we propose the use of a well known generic format, such as XML or JSON.

As mentioned in Section II-A, applications are free do define a structure for relations and respective attributes within a content. However, if different applications access a common type of content they would benefit from a standardized relations structure. This can be achieved with the use of schema languages to define the relation structure of specific contents. The above mentioned formats also present languages for schema definition, namely XSD [4] and JSON-Schema [5], respectively. Content schemas themselves can also be published as objects in the network (the specifics of their distribution lies beyond the scope of this paper).

In the next section, we explore a scenario in which contents with redundant components are decomposed and distributed as related objects. Our goal is to demonstrate the potential of content modeling enabled by object relations.

III. Evaluation Methodology

This section describes the case study used in our evaluations. Distribution of multimedia content over CCN is used as basic scenario because it is a relevant application that presents a high network resource requirement. We employ relations as foundation for content decomposition, with the goal of eliminating the data redundancy in published objects. To guide our analysis we defined two main research questions to be answered by a rich series of experiments:

Q1: How much network users performance is improved with relation-based decomposition? The goal is to evaluate the improvement in clients experienced quality and publishers request load.

Q2: How much more efficiently network resources are used when relation-based decomposition is employed? The goal is to understand the benefits on the network, specially in the overall traffic and cache utilization.

We implemented a relations mechanism based on the design decision described in Section II-C over the latest version of CCNx (currently 0.8.2). However, to achieve a higher scale in our experiments, we decided to use a simulation environment to conduct our analysis. In the remainder of this section, we describe the characteristics of the scenario employed in our

experiments. The observed metrics and result analysis will be discussed in Section IV.

Simulation environment. We extended the well-known ndnSIM simulator [6] to include support for CCNrel and decomposition. We use the simulator to evaluate scenarios where a multimedia content is modeled as a set of decomposed objects, which represent a video and different audio channels for it. We name this simulation scenario **CCNrel**. We also employ an unmodified version of the simulator as a baseline scenario, in which each content variation (choice of audio and video channel) is published as a unique object. This baseline scenario is named **default CCN**. The results presented in Section IV focus on the performance difference between CCNrel and default CCN. Additionally, we explore the impact of different content popularity distributions by using two different parameter values, which will be discussed later. So, there are four different scenarios to be evaluated in the experiments.

Content and workload. The requested multimedia content follows the characteristics from common VoD systems. We model each content as an HD video file with a content length of 25 min. The content stream rate is 5 Mbps, out of which 192 Kbps are related to audio. The content catalog is composed by a set of 10.000 movies published by a single producer. Each movie, in turn, has a fixed number of versions, equivalent to the available options of audio channel, which is set to the number of nodes present in the topology. Without loss of generality, objects distributed to clients are divided in 50 KB chunks.

Requests for contents are generated continually at each network node with intervals defined by a Poisson process. Content selection is made according to a Zipf popularity distribution with its α parameter set to 0.7 and 1.2, which encompass the popularity curve known to model contents in a VoD system ($\alpha = 1.0$) [7]. The chosen values reflect oscillations that occur on the popularity of VoD content due to factors such as viewing hours. Also, the chosen values are similar to those employed in current literature [8]. The content version, in turn, is selected by users according to their location in the network, which is the same of the network node they are connected to. Each node, in turn, is assumed to have a locality distinct to each other in the network. We employ such a behavior to simulate the effects of locality in object requests (for example, audio files tend to be chosen according to country).

Network infrastructure. We use topologies based on real traces obtained from the Internet Topology Zoo [9]. We experimented with multiple topologies and found that different configurations yielded results with congruous behavior. Thereby, the results presented later are based on one representative topology, namely the British Telecom Latin America. This topology presents a total of 45 nodes and 50 links among them. In the simulation, CCN packets are routed according to the shortest path to the publisher, creating a spanning tree rooted in the content provider. Consequently, from the 50 topology links, 44 are actually used for content distribution.

Regarding routers cache, previous studies argue that the available space will be small with respect to content catalog in order to maintain line rate lookup speeds [10]. Thus, our evaluation uses cache with space equivalent to 1% of the content catalog size of the default CCN case. Finally, regarding the content publisher, we position it on a randomly selected node, which varies in each experiment runs.

Table I
SIMULATION PARAMETERS

Parameter	Value
Content catalog size	10,000
Content popularity	Zipf with $\alpha = \{0.7, 1.2\}$
Content length	25 min
Content versions	44
Chunk size	50 KB
Total content bitrate	5 Mbps
Audio content bitrate	192 Kbps
Video content bitrate	4,808 Kbps
Topology	45 nodes, 44 active links
Cache size	1% of catalog (default CCN)
Simulation time	60 min

Execution. The execution starts with all contents published and empty caches. A *warm up* time is employed to stabilize network conditions prior to observation. It is comprised by the period since the beginning of execution until the moment caches are completely filled, as in [11]. After the warm up period, we let the network execute for 60 minutes. The results presented in Section IV consider values after warm up.

Our simulation campaigns are comprised of multiple executions of both simulators (CCNrel and default CCN). The results presented in the next section are based on central tendencies from multiple executions. We summarize the parameters of our simulation in Table I.

IV. EVALUATION RESULTS

We now focus on a thorough evaluation of the proposed case study. Prior to each result, we describe the computed metrics and any particular simulation configuration where appropriate. As expected, the obtained results will provide detailed evidence on how relation-based decomposition (*i*) improves the performance of network clients, and (*ii*) contributes to the better utilization of network resources.

A. User Performance

Our analysis begins with a comparison of user performance between CCNrel and default CCN. In the evaluation, besides the client requesting contents, we also consider the publisher as a network user. Thus, we focus firstly in two metrics: the *client download time* and *data volume served by the publisher* (or publisher load). The first metric reflects the QoE perceived by clients. The second metric, in turn, is an indicative of the resources required by a publisher to distribute content. With the proposed mechanism we expect both clients download time and publisher load to be reduced. Finally, to complement the results from the aforementioned metrics, we focus on the *request hop distance*. This last metric indicates how far requests are forwarded before fulfilled and, consequently, the efficiency of in-network caching.

Client download time. Download times observed in our experiments are presented in a CDF, depicted in Figure 4. The figure has two pairs of curves, presenting the results for

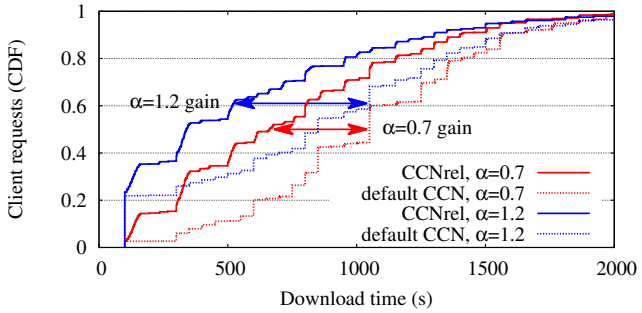


Figure 4. CDF of clients download time

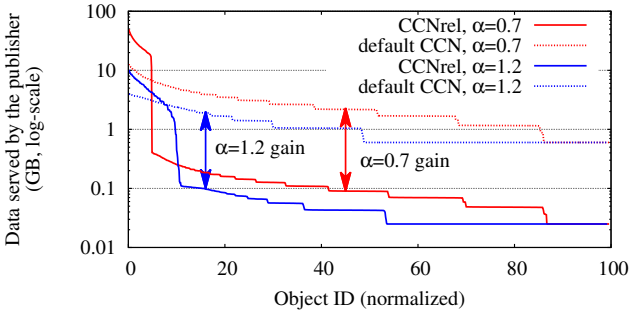


Figure 5. Publisher load

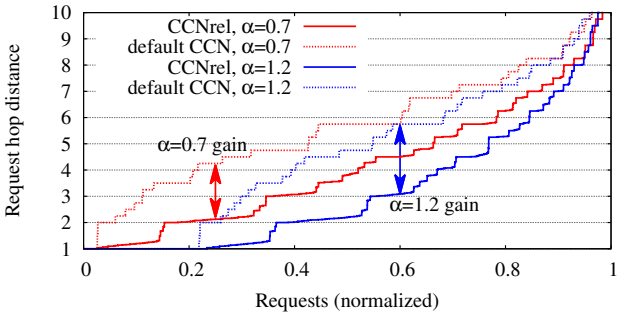


Figure 6. Request hop distance

scenarios with either $\alpha = 0.7$ or $\alpha = 1.2$. Each pair, in turn, compares the performance of CCNrel and default CCN.

Figure 4 shows an improvement of average client download times when CCNrel is used in both content popularity scenarios. When popularity is configured to $\alpha = 0.7$, CCNrel reduces download times 29.2% on average. This occurs because the proposed mechanism eliminates redundancy of objects through decomposition. As result, requests previously scattered among duplicated objects become concentrated in less chunks, increasing their distribution performance. With $\alpha = 1.2$ the achieved reduction of download times reaches 34.3%. This is explained by the increased concentration of requests to already popular objects, which amplifies the benefits of the mechanism (as demonstrated later in our analysis). In the curves from scenarios with $\alpha = 1.2$ the first 20% of requests present very similar download times, indicating a negligible impact. We verified that these requests are directed to objects with high popularity, which were stored in caches at 1-hop distance from clients. Consequently, these requests are fulfilled with very small latency, independent of CCNrel usage.

Publisher load. The publisher load, in turn, is depicted in Figure 5. Its horizontal axis presents the objects ordered by their popularity while the vertical axis (which is in logarithmic scale), the volume of data transferred by the publisher for each object (the lower, the better). Similarly to the previous graph, Figure 5 depicts two pairs of curves to compare the performance of CCNrel and default CCN under different popularity configurations.

Results show that, for both values of alpha, there is a group of objects (10% of the catalog) that generate a higher data volume with CCNrel. This occurs because requests previously scattered among different copies of duplicated data are now concentrated in one object due to redundancy elimination. Because of the higher request ratio, objects with higher popularity generate more traffic volume in the entire network, including the publisher. However, the remaining objects present a reduction on the data volume served by the publisher. This happens because these objects become smaller with redundancy elimination (as explained later in our analysis). Looking at the overall data volume served by the publisher, we observe a difference of 45.7% when $\alpha = 0.7$ and 55.7% when $\alpha = 1.2$.

Request hop distance. The gains perceived by users are a consequence of the positive effects of CCNrel over the network infrastructure. This is evidenced by the *hop distance of requests*, which we explore next. We consider as request hop distance the number of hops traversed by a request prior to reaching a content copy. Its value depends on the topology properties and the shortest path between clients and the publisher. In simulations, the distance between clients and the publisher had a minimum of 2 hops, a maximum of 10, and an average of 5.3. Shorter hop distances indicate that requests are fulfilled by copies from caches closer to clients (the lower the curve, the better). We expect the use of CCNrel to result in a reduction on the overall hop distance. Figure 6 depicts the values of this metric observed in the experiments. Its horizontal axis presents the normalized number of requests ordered by their respective hop distance.

The request hop distance corroborates the results observed in user performance. That is, we observe a general reduction on the hop distance when CCNrel is used. This result demonstrates that in-network caching is fulfilling more client requests in cache units closer to requesters. Consequently, the server load and the download time of clients are reduced, improving overall distribution performance. Moreover, the value of α also affects the hop distance of requests. In average, when $\alpha = 0.7$, the hop distance reduction offered by CCNrel is of 26.3% and, when $\alpha = 1.2$, the hop distance reduction is of 34.3%. This indicates that the gain provided by CCNrel depends on the value of α. In the curves from scenarios with $\alpha = 1.2$ the first 20% of requests are fulfilled within 1 hop. As previously explained, these requests belong to very popular objects that tend to constantly remain in cache.

So far, we found out that *relation-based decomposition improves user performance because it reduces both client download times and requests served by the publisher.* This is a consequence of smaller request hop distance, which is a result of better of network resource utilization due to less data redundancy. Next, we explore how CCNrel influences network resources, such as bandwidth and cache space.

B. Network Resources Utilization

Next we explore the impact of CCNrel on the behavior of network resources efficiency. Thus, we first explore the metrics of *network traffic* and *cache hit ratio*, which indicate if the available network bandwidth and cache space, respectively, are efficiently used. To further explain the behavior of network resources efficiency, we explore the *number of caching operations*, which indicates how frequent content is substituted in caches. Finally, we focus on the *object request distribution*, which indicates how the object catalog popularity is influenced by CCNrel and the consequent impact on the behavior of in-network caching.

Network traffic. A consequence of the smaller request path size is the reduction of the *network traffic* generated by content distribution. Requests are satisfied by caches closer to the clients, thereby reducing the number of links through which data is transmitted and, consequently, the overall network traffic. We illustrate in Figure 7 the traffic observed in our experiments. The horizontal axis presents topology links used for content transmission, while the vertical axis, the network traffic in logarithmic scale.

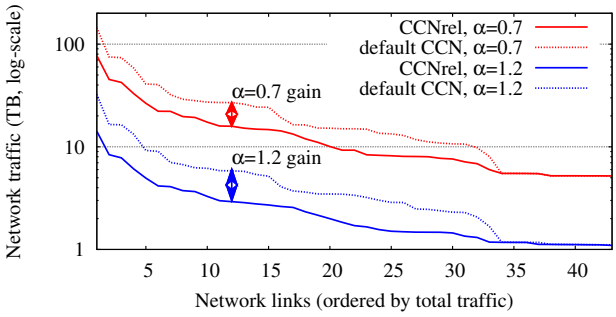


Figure 7. Total network traffic: (average gain of 34% and 42%, shown in log scale)

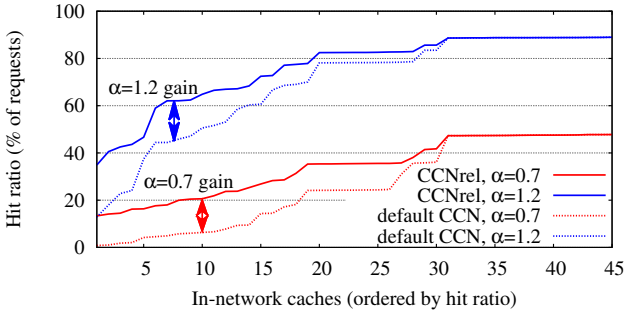


Figure 8. Cache hit ratio

Considering the overall network traffic, CCNrel offers a reduction of 33.6% on transmitted data when $\alpha = 0.7$, while with $\alpha = 1.2$ the reduction is of 42.4%. There are two distinct behaviors depending on the distance of the link to the content publisher. Links closer to the publisher transmit requests (and data) for multiple content versions because they are hubs from multiple network regions. Since CCNrel enables objects to be shared among different versions, cache is used more efficiently than in CCN, resulting in objects stored closer to the edge of the network. Therefore, the traffic reduction on these links is higher. The other behavior concerns links connected to leaf routers. They only transmit data from a single version to clients, causing most (or all) of the popular contents to be in cache, in both CCNrel and default CCN. Consequently, these links present small traffic difference when CCNrel is used.

Cache hit ratio. The reduction of the network traffic and requests path occurs because in-network caches present a higher *hit ratio*. Figure 8 depicts the hit ratios observed in our experiments. Its horizontal axis presents the caches from each router ordered by observed hit ratios.

Results show that CCNrel offers an average improvement of 33.6% in the cache hit ratio when $\alpha = 0.7$ and 9.8% when $\alpha = 1.2$. CCNrel presents a smaller hit ratio gain with $\alpha = 1.2$ because caches in the default CCN have a considerable efficiency gain with such a content distribution. As result, the average CCNrel gain is reduced in comparison to $\alpha = 0.7$. Regarding the behavior of CCNrel, similar to network traffic, routers have their performance based on topological position. Routers closer to the publisher (or core routers) may serve several different versions of the content. With CCNrel, a smaller number of chunks must be cached to distribute all content versions, allowing more objects to be stored in each router. By covering a higher percentage of the content catalog than CCN, core routers in the CCNrel scenario have higher cache hit ratio. Edge routers (leafs of the spanning tree) need to serve only one version. Consequently, CCNrel will have negligible impact to the cache hit ratio in these routers.

Caching operations. Caches are more efficient with CCNrel because their content changes less frequently. This effect can be evaluated by the *number of operations performed by caches*, as measured in [12]. This metric presents the number of storage and eviction operations performed in each network cache. When less content substitution occur there is a higher chance that a content will remain available to fulfill new requests. We compute the number of cache operations from our experiments and depict it in Figure 9. The horizontal axis presents the topology routers in order of cache operations.

Results show that CCNrel reduces cache operations in 32.1% when $\alpha = 0.7$ and 43.9% when $\alpha = 1.2$. This phenomenon is an effect of the reduction in the object catalog data redundancy. As result, the cache space available for other contents to be stored is increased and the the substitution of contents due to cache eviction policies is reduced. Consequently, when a content is stored it will remain longer in caches because there is a smaller chance that it will be substituted. Values of cache operations follow the same behavior from results of traffic and cache hit ratio:

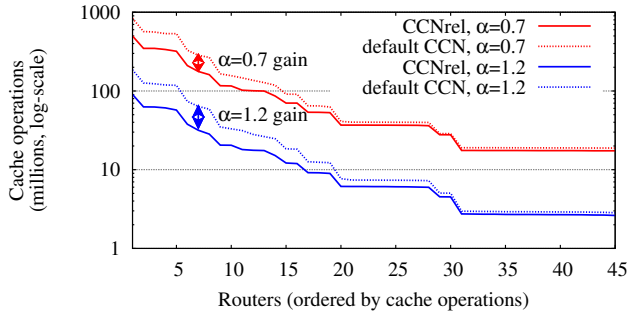


Figure 9. Caching operations: (average gain of 32% and 44%, shown in log scale)

the gain in the number of operations is higher in routers closer to the publisher. The higher performance difference with $\alpha = 1.2$ occurs because requests become even more concentrated, further reducing the cache operations due to less distinct objects forwarded by routers.

Distribution of requests. CCNrel reduces cache operations because it alters the *distribution of requests* for different objects. Recall from the previous section that clients select (*i*) a content to request based on a Zipf distribution and (*ii*) an audio version based on the locality of the network node they are connected to. These two factors and the use of CCNrel will directly influence the distribution of requests to the available object catalog. We expect CCNrel to concentrate requests even more into a smaller group of objects in comparison to default CCN. The request distributions observed in our experiments are depicted in Figure 10. All graphs present on the horizontal axis the objects ordered by their popularity. Also, the vertical axis of all graphs are in logarithmic scale to ease the visualization of low popularity contents. We present results in two graph versions for the sake of clarity (horizontal axis are either in linear or logarithmic scale).

Results show that CCNrel reduces the request rate of the least popular objects in approximately 96% for both values of α. Also, the data request volume of the most popular object with CCNrel is 13.6 times higher than CCN with $\alpha = 0.7$ and 14.6 times higher with $\alpha = 1.2$. This occurs because the long tail of the distribution is mostly composed of objects carrying audio information. These objects are requested by clients interested in a specific content version and are composed by a smaller number of chunks. Video objects, in turn, will have their popularity increased, being concentrated on the beginning of the curve. This occurs because the requests from redundant data will be concentrated on a common (smaller) set of chunks that are the result from content CCNrel.

The behaviors observed in the request distribution and cache efficiency occur due to the number of published objects and their sizes. Recall that in our evaluation scenario 10,000 contents are available. In the default CCN each content version results in a complete object with audio and video. Therefore, considering 44 content versions, there are 440,000 objects available to clients. Taking into account content length and chunk size, the above object catalog results in approximately 412 billion chunks. In the CCNrel case, however, contents

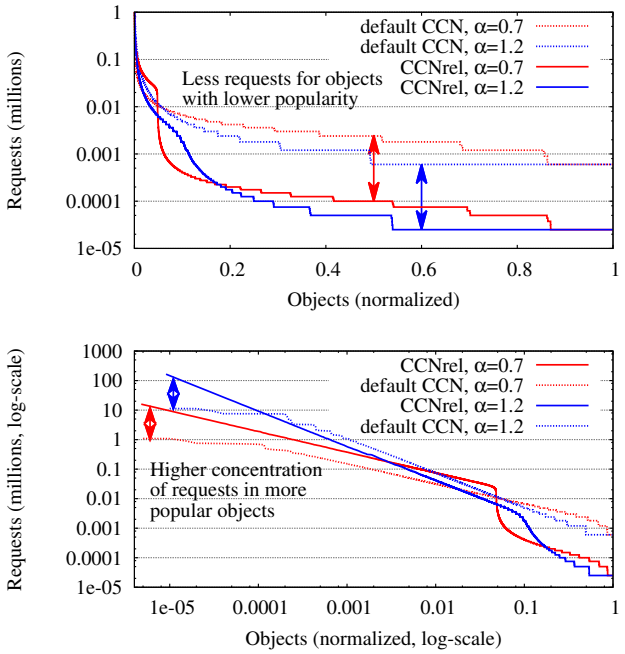


Figure 10. Object request distribution

generate a total of 450,000 objects, a 2.3% increase compared to default. However, the number of chunks is decreased to 16.5 billion, a 96% decrease in the catalog space. This reduction occurs because, with CCNrel, only 10,000 objects represent the video channel. The remaining 440,000 are only audio data, which is 96% smaller than video.

Summarizing, we observed that *relation-based decomposition improves the utilization of network resources, reducing network traffic and increasing the hit ratio of caches*. This occurs because *relation-based decomposition reduces the catalog storage size and modifies the distribution of requests among objects in a way that improves cache usage and stability*.

V. Related Work

In this section, we first discuss related work that employs the concept of content and semantic relations at the application layer. Next, we focus on relation mechanisms proposed in the context for CCN and other ICN architectures.

The concept of relations is explored in different systems for content distribution. The most popular example are Web pages, which employ relations as a fundamental concept for the creation of complex documents with the use of multiple objects. In the context of multimedia, relations are employed to enable the creation of complex contents based on multiple audio and video channels. The DASH standard [13] employs a manifest file to specify a multimedia content with various components available in different HTTP URLs. The client can select a subset of these components for playback depending on its quality requirements. Also related to multimedia, SMIL [14] is a markup language used for the composition of rich multimedia presentations based on audiovisual elements stored in different objects. The above examples employ a specific

format to describe how contents are formed with data acquired from different source objects. However, it is important to note that these formats are created targeting specific application domains and they are not applicable to other types of contents.

The literature of ICN includes three pieces of work (ICN-RE [15], NetInf [16], PSIRP/Pursuit Blackadder [17]) related to our CCNrel proposal.

ICN-RE [15] employs a concept similar to relations for implicit redundancy elimination of object data. In a nutshell, ICN-RE identifies, isolates and publishes byte-identical portions of different contents as a single object. The remaining parts are published individually. The mechanism uses a meta-object that lists the names of all objects that should be downloaded to rebuild the original content. ICN-RE uses the concept of relations in the meta-object to enable redundancy elimination of objects data. However, the format of this meta-object is strictly designed for the mechanism of implicit redundancy elimination. Further, ICN-RE requires modifications in ICN routers.

NetInf [16] introduces the concept of information objects (IOs), which are collections of metadata and pointers to actual data objects. The metadata contained in IOs allows clients to perform semantic queries about published contents. The PURSUIT project proposes a publish/subscribe architecture including an information-centric middleware [17] for the Blackadder prototype based on semantic technologies and metadata. The proposed middleware enables, among other things, establishing relations through common semantic attributes. The fundamental difference of these works on ICN is the focus on ICN architectures other than CCN. Thus, their findings cannot be directly extended to CCN due to specificities in naming (e.g., flat IDs under nested scopes) and routing mechanisms (e.g. separated control and data planes [18]). Further, those studies do not provide a detailed evaluation on the potential network performance gains when leveraging content and semantic relations among data objects.

In summary, our work is novel in exploring the design aspects of introducing backwards-compatible relations mechanisms for CCN to allow publishers modeling their contents in innovative ways. We argue that allowing publishers to use their knowledge about contents to define relations among objects can bring benefits beyond those achieved by mechanisms such as implicit decomposition. While current proposals for relations in content distribution are specific to their application domains (e.g. multimedia, P2P), CCNrel is designed to be generic and allow current and future ICN applications to natively benefit from their intrinsic semantic relations.

VI. Final Remarks

While CCN is a promising architectural proposal that enables efficient distribution of uniquely named data objects, its full network performance gains are currently hindered by a lack of means to model related contents in CCN. This paper aims to close this gap by means of CCNrel, a backward-compatible mechanism for CCN that enables publishers to distribute contents as related objects. The proposed mechanism enables features such as content decomposition, object re-use, and efficient content updates.

To demonstrate the potential of CCNrel we employ extensive simulations on a case study of relation-based content decomposition in multimedia content distribution. Results have shown that CCNrel improves user performance, reducing client download times in an average of 34% and publisher request load in 56%. CCNrel reduces the request hop distance and, consequently, improves the network traffic in an average of 43%. The improvements are a direct result of an improved utilization of caches, which have their hit ratio increased 34% on average. All these improvements can be rooted back to the change in object request distribution, since duplicated objects (e.g., as presented in the video content use case) can be concentrated in a single, non-redundant copy.

We plan to further study the impact of relations with the analysis of additional case studies where both obvious and hidden relations can be leveraged. In addition, we plan to conduct experiments for different realistic network topologies using the Mini-CCNx emulator first, and then running the CCNrel prototype in a global-scale testbed. Finally, we envision the use of relations to be applied in network-level components such as caching and routing.

References

[1] L. Zhang et al., "Named data networking project," NDN Project, Tech. Rep. NDN-0001, 2010. [Online]. Available: http://named-data.net/wp-content/uploads/TR001ndn-proj.pdf

[2] V. Jacobson et al., "Networking Named Content," *Communications of the ACM*, vol. 55, no. 1, pp. 117–124, January 2012.

[3] G. Xylomenos et al., "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–26, Jul. 2013.

[4] D. C. Fallside and P. Walmsley, "XML schema part 0: Primer second edition," 2004.

[5] F. Galiegue, K. Zyp, and G. Court, "JSON schema: core definitions and terminology," 2013.

[6] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN Project, Tech. Rep. NDN-0005, 2012.

[7] J. Choi, A. S. Reaz, and B. Mukherjee, "A Survey of User Behavior in VoD Service and Bandwidth-Saving Multicast Streaming Schemes," *IEEE Comm. Surveys Tutorials*, vol. 14, no. 1, pp. 156–169, Feb. 2012.

[8] G. Rossini, M. Garetto, D. Rossi, and E. Leonardi, "Multi-terabyte and multi-gbps information centric routers," in *INFOCOM 2014: 33rd IEEE International Conference on Computer Communications*, 2014.

[9] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Comm.*, vol. 29, no. 9, pp. 1765–1775, October 2011.

[10] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, 2013.

[11] G. Rossini and D. Rossi, "Evaluating CCN multi-path interest forwarding strategies," *Computer Comm.*, vol. 36, no. 7, pp. 771 – 778, 2013.

[12] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "Less for More" in Information-centric Networks (ext. ver.)," *Computer Communications*, vol. 36, no. 7, pp. 758–770, Apr. 2013.

[13] ISO/IEC, "Information technology – dynamic adaptive streaming over HTTP (DASH) – part 1: Media presentation description and segment formats," 2014.

[14] D. Bulterman et al., "Synchronized multimedia integration language (SMIL 3.0)," 2008.

[15] D. Perino, M. Varvello, and K. Puttaswamy, "ICN-RE: Redundancy Elimination for Information-Centric Networking," in *ICN '12: SIGCOMM Information-Centric Networking Workshop*, 2012, pp. 91–96.

[16] C. Dannewitz et al., "Network of Information (NetInf) – An information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, Apr. 2013.

[17] B. Tagger, D. Trossen, A. Kostopoulos, S. Porter, and G. Parisis, "Realising an application environment for information-centric networking," *Computer Networks*, vol. 57, no. 16, pp. 3249–3266, Nov. 2013.

[18] P. Jokela, A. Zahemszky, C. Rothenberg, S. Arianfar, P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," in *SIGCOMM '09: ACM SIGCOMM Conf. on Data Communication*, pp. 195–206.

APPENDIX C — PAPER SUBMITTED TO ELSEVIER COMNET

- **Title**: *NDNrel: a mechanism based on relations among objects to improve the performance of NDN*
- **Journal**: Elsevier Computer Networks (COMNET)
- **Qualis**: B1

In this paper, we extend our previous work on the relations model and the NDNrel backward-compatible mechanism that allows publishers to distribute contents as related objects. We present a new version of our mechanism that reduces the cost of distributing relation information to clients by leveraging advances in the usage of metadata and manifests on ICN architectures. Also, the new version considers additional aspects such as the authentication of relations and data that comprise the content. We revisit our evaluation of the mechanism and present an extensive analysis of the overhead caused by the usage of relations. We explore a new case study, focused on the distribution of Web content, which allow us to carefully analyze the trade-off between latency and additional transmission cost generated by relation information. Our findings demonstrate that, even with the additional overhead incurred by relations, the mechanism can reduce on average 28% client download time, and 34%, global network traffic.

Elsevier Editorial System(tm) for Computer Networks

Manuscript Draft

Manuscript Number:

Title: NDNrel: a mechanism based on relations among objects to improve the performance of NDN

Article Type: Regular Paper

Keywords: information-centric networks; content-centric networks

Corresponding Author: Prof. Marinho Pilla Barcellos, Ph.D

Corresponding Author's Institution: UFRGS - Federal University of Rio Grande do Sul

First Author: Rodolfo S Antunes, PhD student

Order of Authors: Rodolfo S Antunes, PhD student; Matheus B Lehmann, MSc in CS; PhD Student; Rodrigo B Mansilha, MSc in CS; PhD student; Luciano P Gaspary, PhD; Marinho Pilla Barcellos, Ph.D

Abstract: Named-Data Networking (NDN) is a promising architectural approach that focuses on the efficient distribution of data objects. Each of these objects represents an individual piece of content that is uniquely named and can be cached by network nodes. Recent work on ICN explores new uses of the data object concept to enable advanced applications based on content distribution. Such work includes the use of metadata and manifests to describe contents as sets of multiple correlated objects. In this paper, we extend our previous work on a model and a backward-compatible mechanism, named NDNrel, that allows publishers to distribute contents as related objects. We present a new version of our mechanism that reduces the cost of distributing relation information to clients by leveraging advances in the usage of metadata and manifests on ICN architectures. Also, the new version considers additional aspects such as the authentication of relations and data that comprise the content. We revisit our evaluation of the mechanism and present an extensive analysis of the overhead caused by the usage of relations. We explore a new case study, focused on the distribution of Web content, which allow us to carefully analyze the trade-off between latency and additional transmission cost generated by relation information. Our findings demonstrate that, even with the additional overhead incurred by relations, the mechanism can reduce on average 28% client download time, and 34%, global network traffic.

28 February 2016

Dear Computer Networks Editor-in-Chief,

Please find attached a paper we are submitting to Computer Networks Journal. The paper represents a substantially enhanced version of a paper published in the 14th IFIP/IEEE Symposium on Integrated Network and Service Management (May 2015), and brings enough of **novel content** in comparison with our previous work, with three points to be highlighted.

First, the paper being submitted introduces an **enhanced version** of a backward-compatible mechanism that extends the NDN architecture to enable the distribution of contents as related objects. In this new version, we incorporate recent advances in the use of metadata and manifests on ICN architectures (discussed in IRTF ICN working groups) and review additional design aspects, such as the authentication of relation information and the data that comprises a content.

Second, to make this submission more robust, we added a **new set of results** obtained from a case study with Web content. Unlike the previous case study, this one is intended to “stress” the relations mechanism with small objects, allowing us demonstrate the potential benefits of relations to the network and applications performance when content data is smaller and induce larger overheads.

Third and last, there were **general improvements** throughout the paper, such as the inclusion of further detail about the relations model proposed for our mechanism and a rewrite of the text to provide better explanations, with simple and updated examples.

The overall result is a paper with a substantial delta in contribution over our previous publication.

Yours faithfully,

Prof. Marinho Barcellos, on behalf of the authors.

NDNrel: a mechanism based on relations among objects to improve the performance of NDN

Rodolfo S. Antunes, Matheus B. Lehmann, Rodrigo B. Mansilha, Luciano P. Gaspary, Marinho P. Barcellos

Institute of Informatics – Federal Unifersity of Rio Grande do Sul (INF/UFRGS)
Av. Bento Gonçalves, 9500, Setor 4 – Porto Alegre, RS

Abstract

Named-Data Networking (NDN) is a promising architectural approach that focuses on the efficient distribution of data objects. Each of these objects represents an individual piece of content that is uniquely named and can be cached by network nodes. Recent work on ICN explores new uses of the data object concept to enable advanced applications based on content distribution. Such work includes the use of metadata and manifests to describe contents as sets of multiple correlated objects. In this paper, we extend our previous work on a model and a backward-compatible mechanism, named NDNrel, that allows publishers to distribute contents as related objects. We present a new version of our mechanism that reduces the cost of distributing relation information to clients by leveraging advances in the usage of metadata and manifests on ICN architectures. Also, the new version considers additional aspects such as the authentication of relations and data that comprise the content. We revisit our evaluation of the mechanism and present an extensive analysis of the overhead caused by the usage of relations. We explore a new case study, focused on the distribution of Web content, which allow us to carefully analyze the trade-off between latency and additional transmission cost generated by relation information. Our findings demonstrate that, even with the additional overhead incurred by relations, the mechanism can reduce on average 28% client download time, and 34%, global network traffic.

1. Introduction

Information-Centric Networking (ICN) proposes a communication architecture with mechanisms tailored to improve the performance of content distribution [1, 2, 3]. The performance of these mechanisms is directly impacted by the structure of the contents. More specifically, how the concept of uniquely identified data objects is applied to different types of information.

Some dominant types of content, such as video [4] and Web pages [5], can be regarded as sets of individual data elements. To trivially distribute such contents through an ICN, a publisher could naively store all associated elements in a single object. However, as we will show later, this method would negatively impact the performance of content distribution, because of oversized objects and excessive data redundancy. More advanced distribution mechanisms can help publishers distribute more efficiently sets of individual data elements.

Manifests and metadata enable the design of new distribution mechanisms for ICN applications, and this is currently a highly debated theme [6] in the research community. In our study, we explore how they can make the distribution of multiple-object contents more efficient. More specifically, we allow publishers to describe a content as a set of individual objects, by establishing *relations* among them. In a nutshell, a relation is a link between two objects indicating that the data from one complements in some way the data from the other. Following this concept, a content becomes a collection of multiple objects and the respective relations that characterize the interactions among their data. We employ these definitions to formulate a model that publishers can use to distribute contents as sets of related objects.

This paper expands our study, first introduced in [7], on a backward-compatible extension to the NDN architecture that enables the distribution of contents as related objects. Compared to our previous work, this paper includes two main contributions. First, it describes the new version of our relations mechanism for the NDN architecture, henceforth called NDNrel. This new version reduces the cost of distributing relation information to clients. We achieve such reduction by incorporating recent advances in the use of metadata and manifests on ICN architectures. Also, we review additional design aspects, such as the authentication of relation information and the data that comprises a content. Second, we revisit the evaluation of the proposed mechanism. We add a second case study, based on the Web. We chose a scenario to stress the relations mechanism, to determine potential gains and measure traffic overhead when objects are smaller and in-

Email addresses: rsantunes@inf.ufrgs.br (Rodolfo S. Antunes), mblehmann@inf.ufrgs.br (Matheus B. Lehmann), rbmansilha@inf.ufrgs.br (Rodrigo B. Mansilha), paschoal@inf.ufrgs.br (Luciano P. Gaspary), marinho@inf.ufrgs.br (Marinho P. Barcellos)

duce larger overheads.

The concept of relations behind NDNrel is flexible because it does not impose any restriction on the way publishers can model contents (Sec. 2). In turn, the mechanism implementation (Sec. 3) benefits from recent advances in ICN research, such as manifest objects, to improve the performance of applications that employ relations. To demonstrate the potential of NDNrel, we evaluate it in two case studies, focused on popular Internet applications. The first case study focuses on multimedia content distribution (Sec. 4). In turn, the second case study focuses on the distribution of HTML content (Sec. 5). As expected, results from both evaluations show that the achieved data redundancy elimination leads to promising increases in user and network performance, even in an unfavorable scenario with significant overhead from relations.

NDNrel enables relations to be used in arbitrary novel ways beyond those already explored. When compared to related work in Information-Centric Networking (ICN) research (Sec. 6), NDNrel stands out as a proposal tailored to NDN and presenting a backward-compatible approach that does not require modification to routers.

2. A Model for Relations among Objects in ICN

In this section, we first discuss the fundamental concept of relations employed in NDNrel, our backward-compatible relations mechanism proposal for NDN. Next, we exemplify the use of relations when modeling contents.

2.1. Fundamental Concepts

The proposed NDNrel model employs three fundamental elements: object, relation, and content. An *object* represents an individual piece of information that is uniquely identified in the network. The model also assumes that objects are the basic data element that clients can request to the network. This representation is equivalent to the one adopted by current ICN architectures [8] and guarantees compatibility with applications that opt not to use the proposed model. In turn, the characteristics of the data contained in objects will vary according to the application that publishes contents and how it employs the relations model.

A *relation* from an object a to another, b, is a link indicating that the data of a can be complemented by the data of b in some way. Each relation has a semantic that is directly related to the data contained in the linked objects. The semantic depends on the way a specific type of content is modeled using relations. Nevertheless, the NDNrel model enables the semantic of a relation to be explicitly presented to applications using it. To that end, each relation can be associated with one or more *attributes*, which are key-value pairs with additional information to describe it. Applications can use attributes to further analyze a relation and take different actions depending on the content. For example, an application may decide not to retrieve a related object if it judges that the data is not relevant.

Finally, a *content* represents some information that a publisher wishes to make available to clients in the network. Currently, ICN architectures adopt a direct relation when mapping contents to objects. That is, an object o encapsulates all the information required by an individual content C. With NDNrel, the content C can now be represented as a tuple $\langle O, R \rangle$. O is the set of objects that carry the content data. In turn, R is the set of relations that describe how to use the data from objects in O to reconstruct the content C.

ICN architectures do not make assumptions about the semantic of contents stored in objects. In other words, routing and caching mechanisms only see an object as an arbitrary block of data. We employ the same assumptions regarding relations. That is, a priori, the semantic of relations is opaque to the ICN infrastructure and is known by the applications that publish and request the content. We maintain this design decision to keep the network core simple. Semantic inferences over relations and similar tasks are left to the application level, allowing the network to focus on storage and transmission. The above concepts can be used in a myriad of ways to model contents published in an ICN. The next section demonstrates the flexibility of NDNrel by exemplifying different features that can be implemented in ICN with it.

2.2. Modeling Contents with Relations

Recent discussions from the ICN research community suggest that ICN architectures require additional mechanisms for applications to implement advanced features such as: content personalization and internationalization, aggregation of related content data, and data control for versioned contents. To demonstrate the flexibility of the proposed relations model, we focus on three particular usages for relations that enable the implementation of the features mentioned above. These are: (*i*) *decomposition:* publish a content as multiple unique objects; (*ii*) *composition:* share previously published objects to create a new content; and (*iii*) *versioning:* modify an object data without creating an entirely new object copy. We present examples that demonstrate these modelings and the advantages of using NDNrel, namely: a multimedia content, a Web page, and a cumulative log.

2.2.1. Decomposition

Decomposition enables a publisher to divide contents into multiple parts, which are individually made available in the network. More specifically, the process of decomposition assumes that a publisher wishes to distribute a content contained in a mass of data that may be separated into distinct parts. Each part is published as an individual object and relations are used to describe how the data in objects should be used to reconstruct the content. The use of decomposition allows redundant parts

that may exist in the original data to be eliminated, reducing the required storage space and network traffic needed for distribution. Further, decomposition allows clients to obtain only specific content parts, also reducing network traffic and download times.

Figure 1 illustrates an example of a content decomposed using relations. In the example, the multimedia content has a set of options regarding video quality (720p or 1080p), audio language (French or English), and subtitles (French). Relations allow the data of each option to be published as an individual object related to the main content "Movie". The attributes of each relation indicate how the data of each object relates to the main content and the option it represents. A client can access the relations of the "Movie" object to identify the available audio, video and subtitle options and then download only the objects of interest. Consequently, the client only consumes the network resources necessary to obtain the requested content options instead of the entire content data.

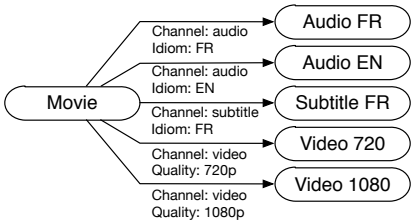


Figure 1: Multimedia modeling example

2.2.2. Composition

Composition enables the creation of new contents based on the information of already published objects. As such, the basic assumption for the process of composition is that a set of objects is available before the creation of contents. The publisher employs relations to create a new content based on the data contained in a subset of objects already available in the network. The publisher may also include newly created objects with additional data of the composed content. The main advantage of composition lies in the lower data redundancy achieved with the reuse of previously published data. Such reuse can also result in higher caching performance depending on the popularity of objects used in composed contents.

Figure 2 illustrates the reuse of objects in the context of Web pages. The example depicts an object that contains the desktop version of a page (`bodydesk.html`). In turn, this object is related to other components required to the page, such as stylesheets, images, and other HTML data. The publisher wishes to add a new version of the page specifically for mobile devices. Instead of creating a new object with all page components, the publisher simply creates objects for the specific elements of the mobile version and reuse the objects already available from the desktop version. In the example, the publisher adds the `bodymobi.html` object, which contains the specific information of the mobile version while linking previously existing objects, such as images and additional HTML data. The publisher also adds a stylesheet specific to the mobile version (`mobile.css`) and ignores the one from the desktop version. Both page versions can be accessed through the web page relation structure, from which the client can identify and obtain the objects specific to the desired version. As previously mentioned, the objects common to both versions are also benefited from improved caching potential since they will receive requests that would be directed to two different copies of the same data.

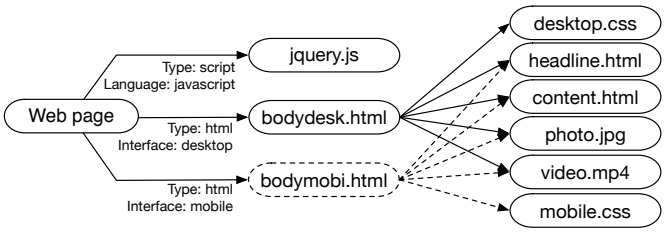


Figure 2: Web page modeling example

2.2.3. Versioning

Versioning enables the data of a previously published content to be updated without breaking the ICN principle of unique naming. The process of versioning assumes that a content is already published using relations (e.g. using decomposition) and must be modified (e.g. adding a new audio channel to a multimedia content). Thus, the process of versioning involves the publication of an updated version of a content relation set, which describes the state of the content after the update. It is important to note that the process of versioning is different from that of composition because the latter assumes the creation of an entirely new piece of content while the first assumes an already existing content is modified. Further, the process of versioning assumes that the new version may exclude objects from a relation structure while composition only assumes the possibility of inclusion. The naming of a content with multiple versions is a known problem in ICN [9]. Relevant solutions to this issue are introduced by ICN architectures and are discussed in the context of this work in Section 3.

Figure 3 illustrates the use of relations to enable the versioning of a log file distributed in an ICN. The log is updated with new entries according to a division criteria (e.g. all entries from a particular hour). Each version n has two relations. The first relation points to the previous version $(n-1)$. In turn, the second relation points to the object containing the new entries, that is, the delta from the previous version (Δn). Redundancy is avoided with this simple model because only the changes from previous versions are added with each update. Furthermore, previous versions of the content are still accessible to applications. It is interesting to note that this simple model of versioned content can benefit from the use of techniques

employed in modern content versioning systems. For example, the *delta-skip* algorithm employed in Subversion [10], which reduces the chain of delta objects that must be processed for the reconstruction of a content with a large number of published versions.

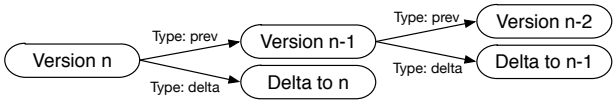


Figure 3: Log file modeling example

The above concepts should be made available as an API for publishing and retrieval of relations to enable the widespread usage of NDNrel to model contents in NDN. The API, in turn, accesses the functionalities of a backward-compatible mechanism implemented on top of the NDN client library. In the next section we consider the key aspects of a relations mechanism for NDN.

3. Design Aspects

The model presented in the previous section can be used to publish contents if it is available as a mechanism in ICN architectures. In this section, we detail the design of NDNrel, a relations mechanism tailored for NDN. Our goal is to implement the relations model using the NDN architecture concepts and features. We identified five main design aspects of the mechanism:

- **Storage:** What type of object is used to store information about relations.
- **Organization:** How can a publisher organize the relation structure to describe the content.
- **Management:** How clients can create, modify and remove relation information from the network.
- **Authenticity:** What methods are used to verify that relation information and the linked objects are authentic regarding the content.
- **Representation:** What format is used to represent the relation structure and data objects of a content.

Each of these aspects is explored in depth next. We begin with the options available to store the relation information.

3.1. Relations Storage

The first aspect to consider is what type of object the mechanism uses to store relation information. The characteristics of the object used to store relations should not limit the flexibility of the mechanism. We consider two main options: *internally* or *externally* to objects with content data. In the first case, relation information is kept in the same objects that store the content data. In the second, information is stored externally to the content object. Next, we detail the storage options, discussing their advantages and limitations.

Internally. The first option is the most trivial and adds relations into the objects with the content data. Figure 4 illustrates it with an example of five objects with embedded relations. These objects can have up to three components: header, relation structure (optional), and content data. The object header must identify the chunks that contain relation information, enabling the clients to retrieve them and parse the relations before requesting the actual content data. The relation structure (when available) describes the content organization whereas the data contains the actual content.

This storage option has a limitation when updating the relation structure. Since ICN principles forbid updates on existing objects, the entire data part would be replicated to edit the relations of an object. Although a versioning control could alleviate the update overhead, the content data would still be replicated in multiple objects differentiated only by their relation structure.

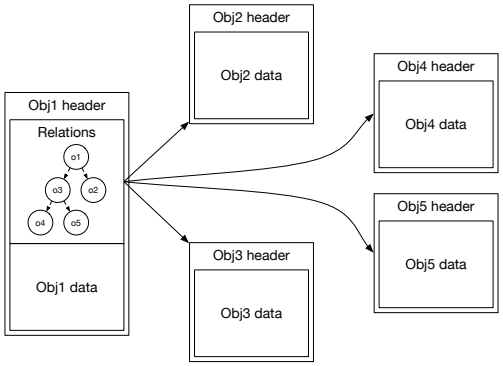


Figure 4: Relations stored within the data object

Externally. The second option is to store the relation information separately from content data. It reduces the overhead to manage relations because there is no need to replicate the content object when when an update of the relation structure is done. Two alternatives exist to implement this option: metadata or manifest objects. We discuss each one next.

Metadata is a piece of information that provides additional information about a published object and can be retrieved independently of the content data. Figure 5 exemplifies the use of metadata to store the relations of the virtual part "Obj1". A virtual part does not contain content data (e.g. the Movie object presented in Section 2.2.1) and results in an object with a header and associated metadata, but no payload. NDN offers a metadata feature that enables publishers to add metadata to objects. These can be accessed using the reserved name component %C1.META and a metadata identifier.

Manifests, in turn, are separate and self-contained objects that maintain the relation structure of a given content. Figure 6 demonstrates the relation structure of the content stored as a manifest. This approach is simpler and

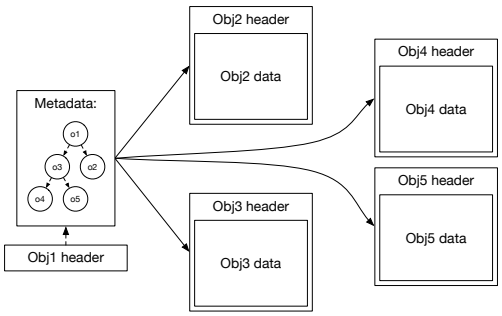


Figure 5: Relations stored using metadata

more flexible than metadata because it does not need to be associated with an existing object.

This storage option could be implemented using the CCNx manifest specification [6], which is a result of an ongoing work of exploring manifests to improve content distribution. The specification defines a simple structure that describes a collection of named objects. However, CCNx manifests fail to satisfy requirements of the relations model (presented in Section 2.1), as we will explain later in Section 3.5.

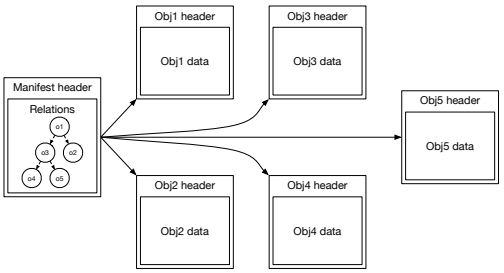


Figure 6: Relations stored using manifests

NDNrel design choice. We choose to store relations using manifests because it is the most flexible option. Storing relations directly into data objects would limit the mechanism ability to manage the relations due to the update overhead. In turn, the use of metadata requires every information to be associated with a published object. This restriction reduces the flexibility of the mechanism, particularly when using "virtual" content elements. Next, we address the retrieval methods available to recover relations stored as manifests.

3.2. Relations Organization

The second aspect regards how the publisher organizes the relation structure of content: *flat* or *hierarchical*. The publisher should consider the trade-off between size and complexity of the relation structure to organize the contents. Next, both approaches are discussed.

Flat approach. The flat approach centralizes the complete relation information of content in a single manifest. This characteristic allows the clients to find the entire content structure in one location and retrieve it with a single request, as shown in Figure 7. On the negative side, this approach does not provide modularity for sub-parts of the content, which prevents their reusability to define other contents. In general, the flat approach is better suited for contents that have simpler descriptions or that need low retrieval delay.

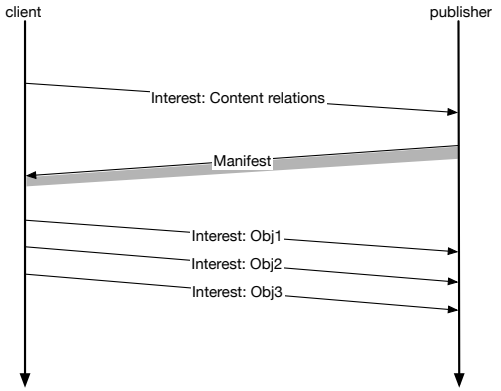


Figure 7: Relation retrieval using flat organization

Hierarchical approach. The hierarchical approach allows the publisher to divide the relation information into multiple manifests. Sub-parts of content are structured as modules, and they can be reused to compose other contents, simplifying the description of complex contents. Figure 8 illustrates the retrieval process, in which clients can access specific relations. However, they need to issue multiple requests to obtain the entire content structure since the relations are spread in different manifests. This approach is recommended for contents that are complex or popular because they benefit more from the modularity and reusability of manifests.

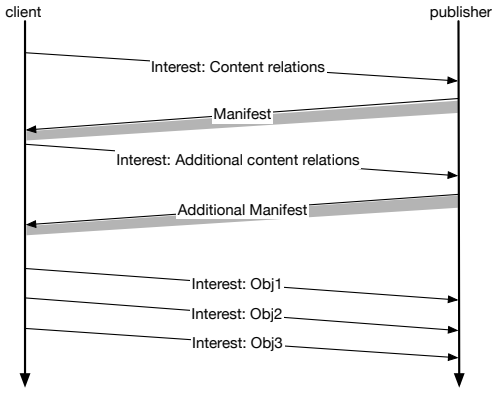


Figure 8: Relation retrieval using hierarchical organization

NDNrel design choice. The use of the approaches described above depends directly on the type of content to be described with relations. Thus, we design NDNrel to support

both approaches, increase the flexibility of the mechanism. Publishers should use their knowledge of the content to organize the relations in the best way. Next, we explore the alternatives to implement operations to manage relations of contents.

3.3. Relations Management

The management aspect refers to *create*, *modify*, and *remove* relations of contents. Relation mechanisms have different capabilities according to operations they support. In NDN, the creation of manifests is natively supported, whereas the other two operations are not and may require additional mechanisms. We discuss the details of each operation next.

Creating a relation is the simplest operation, requiring only the publication of a set of manifests that describes the content structure. A publisher can use any object available in the network (including those from third-party publishers) in new contents because it does not affect the data object. It is supported natively by NDN and does not require any extension.

To modify a published manifest, the publisher needs to create a new one with the updated information and mark the previous manifest as obsolete. Both actions are required to preserve the unique identification of objects in ICN. NDN supports a simple version control mechanism by adding a name segment containing an incremental counter to the object [11].

The third and last operation can be achieved using two methods. The first one is relying on the modification operation. In the extreme case that all relations are removed, an empty manifest would be published. The second method is to delete the manifest object from the network, which is still an open challenge in ICN [3]. One of the biggest concerns is how to eliminate copies of the object located in clients (other from the publisher) and caches.

NDNrel design choice. To improve the capabilities of NDNrel, we include all three operations. The relation modification is enabled through the versioning segment of the object name whereas its removal is done using the modification operation. Next, we consider issues related to the authenticity of relations.

3.4. Relations Authenticity

The fourth aspect is the authenticity of the relation structure created by the publisher. The mechanism needs to guarantee that the *relation structure* (manifest) and the *linked objects* retrieved by the clients are those specified by the publisher.

Clients can authenticate a relation structure by verifying the authenticity of the manifest objects that describe it. NDN uses a standard public key signature mechanism to support object authenticity, in which every object is signed with the private key of its publisher. The signature information (algorithm, public key, and key locator) are sent with the manifest object to the clients, enabling them to verify its authenticity.

The objects listed in a manifest benefit from the same NDN mechanism but require an extra step to guarantee that their data is what should be used in the content. The publisher can add the signature information of related objects in the manifests (as relation attributes) [12]. This information allows clients to verify the retrieved objects, including those from third-party publishers.

NDNrel design choice. The NDN mechanisms are sufficient to guarantee the authenticity in NDNrel. The manifest (and relation structure) are authenticated without any modification whereas the linked objects have an extra step. We add the signature information of those objects as attributes in the manifests, enabling clients to verify their authenticity. Next, we elaborate some considerations about the description format employed in NDNrel manifests.

3.5. Relations Representation

The fifth and last aspect of the mechanism design is the format to represent the relations. It should support the model features (presented in Section 2.1): (*i*) a hierarchical structure that represents complex contents with multiple levels of relations; and (*ii*) the attributes included in the description of relations. We consider two possible formats: the *CCNx manifest specification* [6] and the *structured description formats* (e.g. JSON or XML).

CCNx manifest specification. The CCNx manifest enables the description of an object collection formatted as a list of names and their corresponding signature value [6]. The specification fails to satisfy the requirements of our proposed model for two reasons. First, it does not allow publishers to add metadata to relations (except its signature value). In other words, publishers can not give any semantics to the relations between the objects as required by our relations model, limiting the description richness of the mechanism. Second, the CCNx manifest uses a simple list of objects to define the content, forcing publishers to use multiple manifests when describing hierarchical (and complex) contents. Not only the specification prevents the use of a single manifest (flat approach) for hierarchical contents, it also restricts the organizational flexibility of relations by requiring a manifest for each level. These limitations reduce the richness and flexibility capabilities to describe contents, narrowing the type of content that the mechanism can represent, and could add significant overhead due to poorly structured content.

Structured description formats. JSON and XML are current standards applied for various uses, especially in Web applications. Both formats satisfy the model requirements by enabling the description of a hierarchical structure of elements and the association of attributes to these elements.

The combination of these properties grants flexibility for the mechanism to represent different types of contents.

NDNrel design choice. Due to the limitations identified in the CCNx manifest specification, the implementation of NDNrel encodes the manifests with a structured description format. Specifically, JSON was chosen over XML because it is less verbose, reducing the mechanism overhead. We may change this decision in the future if the CCNx manifest specification evolves to a design that supports the proposed relations model.

In the following Sections, we quantitatively evaluate the concepts and design decisions of NDNrel to model and distribute content using relations. This evaluation is based on two case studies considering relevant applications on the Internet: multimedia and HTML content. We begin with a summary of the results from the first case study, based on multimedia content.

4. Summary: Multimedia Content Case Study

This section summarizes the main results of our evaluation of a case study based on multimedia content distribution. Our previous work on NDNrel, described in [7], presents the complete analysis of the results obtained in this case study. Due to the nature of the multimedia content, our evaluation did not consider the overhead caused by the use of relations, because the size of content data was predominant on download times and network traffic (Section 5 will present an additional case study to address that aspect).

4.1. Methodology

Our evaluation of the multimedia case study has two main goals: (*i*) identify how much the network users performance is improved with relation-based decomposition; and (*ii*) verify how much more efficiently network resources are used when relation-based decomposition is employed. To achieve these goals, we develop a scenario in which each content comprises multiple audio channels that clients can select to customize their experience.

Our evaluation has two main scenarios: a baseline, named *default NDN*, and another that employs the relations mechanism, named *NDNrel*. In the default NDN scenario, each object contains one of the possible combinations of video and audio channels for a content. That is, each content leads to as many objects as the combination of each available audio channel with the video channel. A client has to discover and request the exact object that contains the data that matches the selected options for viewing the content. In contrast, on the NDNrel scenario, we employ our relations mechanism to decompose each audio and video channel in a separate object. Relations are used to describe the available audio and video channels. Clients use them to identify the available options and requests only the objects containing the exact data that matches the desired customization.

To evaluate the described scenarios, we extended the well-known ndnSIM simulator [13] to include support for NDNrel and content decomposition. We also employ an unmodified version of the simulator to evaluate the default NDN scenario. The methodology used in our experiments is described in detail in [7]. Table 1 summarizes its main parameters.

Table 1: Multimedia case study simulation parameters

Parameter	Value
Content catalog size	10,000
Content popularity	Zipf with $\alpha = \{0.7, 1.2\}$
Content length	25 min
Content versions	44
Chunk size	50 KB
Total content bitrate	5 Mbps
Audio content bitrate	192 Kbps
Video content bitrate	4,808 Kbps
Topology	45 nodes, 44 active links
Cache size	1% of catalog (default CCN)
Simulation time	60 min

4.2. Results

We now describe a summary of the main results from our case study about multimedia content distribution. Our focus is on the main metrics that indicate the impact of relations on application and network performance, which are: *client download time*, *publisher load*, and *global network traffic*. Additionally, we present a summary of the main insights obtained from our analysis. The complete result discussion is available in [7].

We begin with the difference between NDNrel and default NDN regarding client download time, which reflects the application QoE perceived by clients. The download times observed in our experiments are presented in a CDF, depicted in Figure 9. The figure has two pairs of curves, representing the results for scenarios with either $\alpha = 0.7$ or $\alpha = 1.2$. Each pair, in turn, compares the performance of NDNrel and default NDN. With the mechanism, we expect a reduction on the download time.

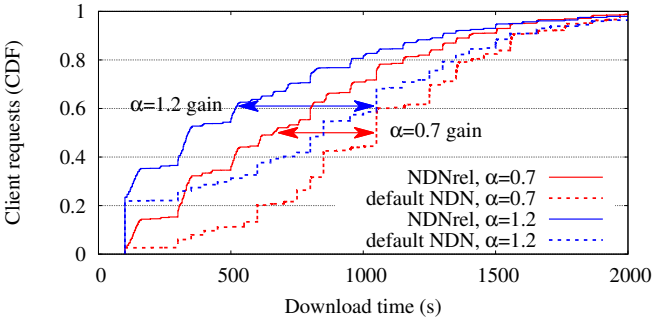


Figure 9: CDF of clients download time

Figure 9 shows an improvement of average client download times when NDNrel is used in both content popularity scenarios. When popularity is lower ($\alpha = 0.7$), NDNrel

reduces download times 29.2% on average. This occurs because the proposed mechanism eliminates redundancy of objects through decomposition. As result, requests previously scattered among duplicated objects become concentrated in less chunks, increasing their distribution performance. With higher popularity ($\alpha = 1.2$) the achieved reduction of download times reaches 34.3%. This is explained by the increased concentration of requests to already popular objects, which amplifies the benefits of the mechanism (as demonstrated later in our analysis). In the curves from scenarios with $\alpha = 1.2$ the first 20% of requests present very similar download times, indicating a negligible impact. We verified that these requests are directed to objects with high popularity, which were stored in caches at 1-hop distance from clients. Consequently, these requests are fulfilled with very small latency, independent of NDNrel usage.

Next, we explore the *data volume served by the publisher* (or publisher load), which indicates the resources a publisher needs to distribute the content. The publisher load is depicted in Figure 10. Its horizontal axis presents the objects ordered by their popularity while the vertical axis (which is in logarithmic scale), the volume of data transferred by the publisher for each object (the lower, the better). Similarly to the previous graph, Figure 10 depicts two pairs of curves to compare the performance of NDNrel and default NDN under different popularity configurations. We expect the publisher load to be reduced when relations are used.

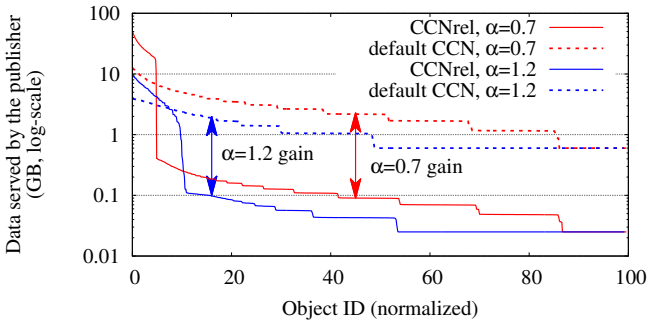


Figure 10: Publisher load

Results show that, for both values of alpha, there is a group of objects (10% of the catalog) that generate a higher data volume with NDNrel. This occurs because requests previously scattered among different copies of duplicated data are now concentrated in one object due to redundancy elimination. Because of the higher request ratio, objects with higher popularity generate more traffic volume in the entire network, including the publisher. However, the remaining objects present a reduction on the data volume served by the publisher. This happens because these objects become smaller with redundancy elimination (as explained later in our analysis). Looking at the overall data volume served by the publisher, we observe a difference of 45.7% when $\alpha = 0.7$ and 55.7% when $\alpha = 1.2$. Albeit expected to be very common, such scenarios represent favorable settings for redundancy elimination.

Finally, we explore the network traffic observed in our evaluation scenarios. We consider the traffic resulting from the transmission of content copies in response to client requests. Figure 11 illustrates the traffic observed in our experiments. The horizontal axis presents topology links used for content transmission, while the vertical axis, the network traffic in logarithmic scale. We expect that the use of relations will reduce the traffic resulting from content transmission.

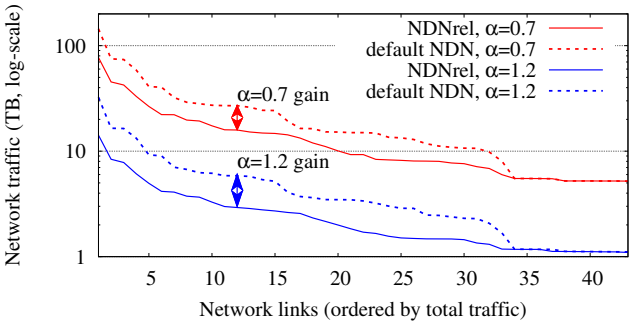


Figure 11: Total network traffic: (average gain of 34% and 42%, shown in log scale)

Considering the overall network traffic, NDNrel offers a reduction of 33.6% on transmitted data when $\alpha = 0.7$, while with $\alpha = 1.2$ the reduction is of 42.4%. There are two distinct behaviors depending on the distance of the link to the content publisher. Links closer to the publisher transmit requests (and data) for multiple content versions because they are hubs from multiple network regions. Since NDNrel enables objects to be shared among different versions, cache is used more efficiently than in NDN, resulting in objects stored closer to the edge of the network. Therefore, the traffic reduction on these links is higher. The other behavior concerns links connected to leaf routers. They only transmit data from a single version to clients, causing most (or all) of the popular contents to be in cache, in both NDNrel and default NDN. Consequently, these links present small traffic difference when NDNrel is used.

The presented results demonstrate that the relations mechanism can reduce the download time, publisher load and network traffic when used for content distribution. Further analysis of our simulation shows that these gains originate from three reasons: *smaller distance between clients and content copies*, *higher cache hit ratio* and *higher concentration of requests to popular objects*.

Our results indicate that the use of relations reduces both the latency to obtain contents and the global network traffic because requests are fulfilled by content copies available in routers closer to clients. In average, when compared to the default NDN scenario, NDNrel reduces the number of hops travelled by queries in an average of 30.3%. The distance to content copies and also the pub-

lisher load are reduced because caches present a higher hit ratio. On average, when compared to the default NDN scenario, the hit-ratio of caches is 21.7% higher with the use of relations.

The higher hit-ratio of caches happens because the use of relations changes the distribution of requests and the sizes of the objects published in the network. These changes occur because relations eliminate the redundancy of the video channel among content variants. Recall that in our evaluation scenario 10,000 contents are available. In the default NDN, each content version results in a complete object with audio and video. Therefore, considering 44 content versions, there are 440,000 objects available to clients. This catalog size results in approximately 412 billion chunks considering content length and chunk size. In the NDNrel case, however, the video channel is a single object that is shared by all variants of a content. As a result, the complete catalog generates a total of 450,000 objects, a 2.3% increase compared to default NDN. However, the number of chunks is reduced to 16.5 billion, a 96% decrease in the catalog space. This reduction occurs because, with NDNrel, only 10,000 objects represent the video channel. The remaining 440,000 are only audio data, which is 96% smaller than video. Because all variants of a content share the objects containing the video channel, they concentrate a higher number of requests. In contrast, in the default scenario, the video data is replicated for each variant, thus spreading requests for the same data and reducing the hit ratio of caches.

This case study demonstrated the potential advantages of using relations to distribute contents. However, it did not present a detailed analysis of the overhead caused by the relations mechanism. To fill this gap, we developed a new case study, which is explored in the next section.

5. Case Study: Web Pages

In this section we look at the effects of the relations model in the distribution of HTML content. In contrast to the multimedia case study, HTML content is comprised of small sized objects, which may result in non-negligible overhead when modeled with relations. In particular, applications may experience higher retrieval latency and the network, higher traffic. However, we show that the change in the objects catalog and its popularity distribution combined with ICN's native mechanisms not only negate such overhead but also improve the application and network performance. The first part of the section describes the methodology employed to evaluate the case study. The second part presents the obtained results.

5.1. Evaluation Methodology

To guide our analysis we defined three main research questions to be answered by a series of experiments:

Q1: How does the use of relations affect the distribution of client requests to objects? Our goal is to identify the impact of relations in the popularity of objects and how it may potentially affect the network mechanisms.

Q2: What is the effect of relations on the usage of network resources? Our goal is to analyze the variation of network bandwidth and cache space required when the relations are used.

Q3: Does the overhead generated by relations negatively impacts application performance in a scenario with contents of small size? Our goal is to study whether the use of relations with small sized objects adds a significant overhead or improves the client performance.

To conduct our experiments, we used our extension of the ndnSIM simulator with NDNrel and added support for content publication with object composition. We use the simulator to evaluate scenarios where the publisher creates new HTML content through the composition of already available objects. Next, we describe the scenario and parameters employed to evaluate the case study.

5.1.1. Simulation Scenario

Our simulation employs two main scenarios: *NDNrel*, that uses our relations mechanism to distribute HTML content with object composition, and *default NDN*, that uses an unmodified version of NDN to distribute the content. Based on results from [5], we divide the HTML content into three main elements: *layout* (CSS and HTML from page layout), *code* (Javascript and HTML from scripting), and *data* (XML and HTML from actual content). We assume that a complete HTML content requires all three parts, which may be distributed as individual objects together with their describing relations (NDNrel scenario) or encapsulated in a single object variant (default NDN scenario). The results presented in Section 5.2 focus on the performance difference between NDNrel and default NDN.

5.1.2. Content and Workload

We analyzed the trace of a proxy server during two months to capture the Web content size distribution. Our analysis observed a variation of HTML objects between 1 kB and 3.26 MB, with an average size of 10 kB. These values are used to describe the content objects (composed of all three components: layout, code, and data) in our simulation. Regarding the size of individual parts in the NDNrel scenario, we employ the observations from [5], which defines a distribution between layout, code, and data approximately to 30-30-40%. The transmission of the objects to clients is done, without loss of generality, using chunks with a maximum size of 1 kB.

The content catalog begins the experiment with 10,000 Web pages and expands throughout the simulation. A publisher creates new *variants* of the starting (original) contents every 2 minutes, an update interval proportional

to the observations presented in [5]. In the NDNrel scenario, each new variant generates an HTML content and a manifest that identifies the objects necessary to reconstruct it. The HTML content created reuses a previously published layout and code objects while it adds a new object as the data part. In the default NDN scenario, the publisher creates a new object containing all three components for each new variant.

Requests for contents are generated continually at each network node with intervals defined by a Poisson process. Content selection is made according to a Zipf popularity distribution with its α parameter set to 0.7 and 0.9, which encompass the popularities observed in known studies about HTML content [14].

5.1.3. Network infrastructure

The experiments are executed using topologies based on real traces obtained from the Internet Topology Zoo [15]. We experimented with multiple topologies and found that different configurations yielded results with congruous behavior. Thereby, the results presented later are based on one representative topology, namely the British Telecom Latin America, which has 45 nodes and 50 links connecting them. In the simulation, NDN packets are routed according to the shortest path to the publisher, creating a spanning tree rooted in the content provider. Consequently, from the 50 topology links, only 44 are used for content distribution.

Regarding the routers cache, previous studies argue that the available space will be small compared to the content catalog size to maintain line rate lookup speeds [1]. Thus, our evaluation uses cache with storage equivalent to 1% of the initial content catalog size of the default NDN case. We employ the default on-path cache from the NDN architecture with LCE (leave a copy everywhere) and LRU (least recently used) policies for cache placement and eviction, respectively. Finally, the content publisher is positioned on a randomly selected node, which varies in each experiment run.

5.1.4. Execution

The execution starts with the initial contents published and empty caches. A *warm-up* period, ranging from the beginning of execution to the moment that the caches are filled (as in [16]), is employed to stabilize network conditions before observation. After the warm-up period, the simulation executes for 60 minutes to measure and collect the evaluated metrics.

The simulation campaigns are comprised of multiple executions of both scenarios (NDNrel and default NDN). The results presented in the next section are based on the central tendencies of the collected metrics. We summarize the parameters of the simulation in Table 2.

Table 2: Web pages case study simulation parameters

Parameter	Value
Initial content catalog size	10,000
New content publication	New content variants every 2 minutes
Content composition	3 parts: layout (30% of content size), code (30%), data (40%)
Content popularity	Zipf with $\alpha = \{0.7, 0.9\}$
Content size	Trace: min 1 kB, max 3.26 MB, average 10 kB
Chunk size	1 kB
Topology	45 nodes, 44 active links
Cache size	1% of catalog (default NDN)
Cache policies	LCE, LRU
Simulation time	60 min

5.2. Results

We present in this section the results obtained from the evaluation scenario previously described. The relation mechanism analysis is divided into the following three parts. First, we discuss its impact on the object catalog, notably its popularity distribution, and caching. Then, we show the benefits of using relations to the end users (clients and publisher). Finally, we investigate the effects on the network, shedding light on the reasons why relations improve the performance of end users. Before the discussion of each result, we describe the evaluated metric and its goal in our analysis.

5.2.1. Object Catalog

The first step in the evaluation is the analysis of how the relation mechanism affects the object catalog and the in-network caching. We investigate the object popularity distribution in both scenarios through the number of times they were requested, shown in Figure 12. Its horizontal and vertical axes represent respectively the published objects (ordered by their popularity) and the percentage of requests issued for a given object. Recall from Section 5.1 that, in the default NDN scenario, an object contains a complete content with all three parts, while in the NDNrel scenario, each object carries an individual content part. We expect that the reuse of objects enabled by the relations will increase the popularity of the objects shared by multiple variants. This behavior is beneficial to the performance of the end users and the network in NDN because it leverages the in-network caching for a better content dissemination, as it will be shown throughout the evaluation.

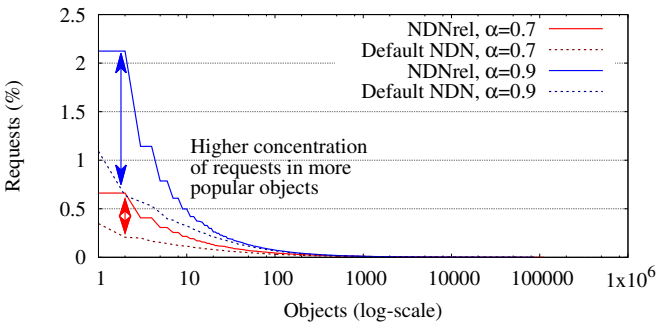


Figure 12: Requests to objects

The object catalog begins with 10 thousand (default NDN scenario) or 30 thousand (NDNrel scenario) objects and evolves through time with the creation of new variants, ending with up to 300,000 (default NDN scenario) or 320,000 (NDNrel scenario) objects. There are three types of objects in the catalog regarding their popularity: popular, unpopular, and ignored. The popular objects are those that receive at least 0.01% of total requests. The unpopular ones are the requested objects that are not popular. Lastly, the ignored objects receive no requests throughout the experiment.

Figure 12 shows that the distributions of popular, unpopular, and ignored objects in the default NDN scenario are 0.6-30.2-69.2% ($\alpha = 0.7$) and 1.5-20.3-78.2% ($\alpha = 0.9$). The high percentage of ignored objects is a combination of the popularity distribution and the lifetime of contents in the simulation (i.e. objects published towards the end of the simulation have a smaller chance of being requested). Regarding the popular ones, although they comprise a small percentage of the catalog, they are responsible for 14.5% ($\alpha = 0.7$) and 35.3% ($\alpha = 0.9$) of total requests of the clients.

The relation mechanism changes the requests issued because it enables objects to be shared among different contents. One the one hand, the results show that the distributions of popular, unpopular, and ignored objects in the NDNrel scenario remain similar to the default NDN one with values of 0.8-34.4-64.8% ($\alpha = 0.7$) and 1.2-25.5-73.3% ($\alpha = 0.9$). On the other hand, the popular objects concentrate a higher number of requests: 25.6% of total requests when $\alpha = 0.7$ and 45.1% when $\alpha = 0.9$. The shared objects accumulate more requests because each of is a component required by multiple content variants. Therefore, every client interested in one of the variants needs to request the shared object to retrieve the complete content.

Figure 13 show the impact of the relation mechanism on the in-network cache hit ratio. The routers are categorized in tiers according to their distance (in hops) to the publisher. The horizontal axis presents the router tiers, and the vertical axis describes the average hit ratio of the routers at the given tier. We expect that the use of relations will increase the cache hit ratio due to the alteration of the content popularity distribution, which makes the popular contents even more popular. In particular, the routers farther from the publisher (i.e. in the leaves of the spanning tree) will benefit more because they receive fewer requests and with less variability than those in the core.

In the default NDN scenario, routers have an average cache hit ratio of 17.5% ($\alpha = 0.7$) and 40.3% ($\alpha = 0.9$). The difference in the results is a consequence of the object popularity distribution employed to issue content requests. We confirm that when more requests are concentrated on the popular objects ($\alpha = 0.9$), there is a higher probability of a cached content being requested because of the caching policies used in NDN. In the case of requests being more evenly distributed ($\alpha = 0.7$), the benefit from caching is lower due to the high variance of requests, resulting in a smaller probability that a given content will be cached.

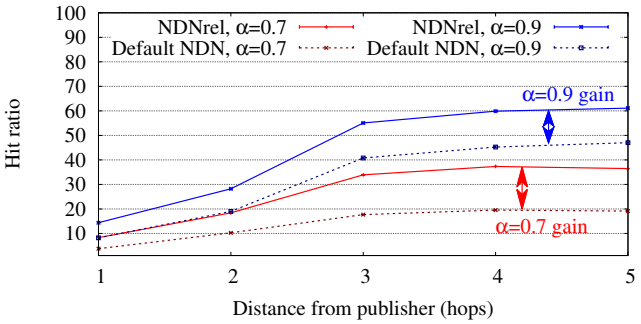


Figure 13: Cache hit ratio

Analyzing deeper into the router tiers, we observe that the cache hit ratio improves as the routers are farther from the publisher. The results vary between 4% and 19% ($\alpha = 0.7$) and from 8% to 48% ($\alpha = 0.9$). The leaf routers (farther from the publisher) serve only a portion of the users, resulting in a small variance of requests for different contents and a higher probability of having them cached. Core routers (closer to the publisher), on their turn, receive the aggregation of multiple leaf routers and users, which causes a higher variance of the requested contents and a lower probability of having them in the cache.

The relations mechanism improves the usage of the in-network caching because of the higher concentration of requests in the popular contents. In average, the routers enhance their cache hit ratio to 33.5% ($\alpha = 0.7$) and 54.1% ($\alpha = 0.9$). These results are intuitive considering the higher concentration of requests to the most popular contents caused by the relations. Specifically, the relation mechanism reuses some published objects to compose various contents, increasing the popularity of the reused objects and concentrating even more the requests for them. The results of the router tiers follow the same trend described in the analysis of the NDN default scenario but with better results. The routers cache hit ratio varies between 8% and 38% ($\alpha = 0.7$) and from 15% to 61% ($\alpha = 0.9$).

In summary, *the use of relations increases the request rate of the popular objects because of the shared objects used to represent contents, which increases the routers cache hit ratio.* They allow the reuse of content parts (that would be duplicated in each variant), removing the data redundancy and concentrating the requests from all different content variants in a single object. Having very popular objects is beneficial for the content dissemination performance because it improves the efficiency of the NDN caching mechanism.

5.2.2. End Users

This section analyzes the benefits caused by the relation mechanism to the end users: clients and publisher. Particularly, we focus on evaluating the clients download time and the publisher load.

Figure 14 illustrates the download time of clients. The horizontal axis depicts the individual requests from clients while its vertical axis, the download time in milliseconds. This metric represents the period to retrieve the entire content, which begins with the request for either the relation (NDNrel scenario) or the first chunk (default NDN scenario) of an object, and concludes with the retrieval of its last chunk. We expect that the relation mechanism will reduce the download time of clients due to a more efficient usage of the in-network caching to disseminate contents, as a result of the change in the popularity distribution of the catalog.

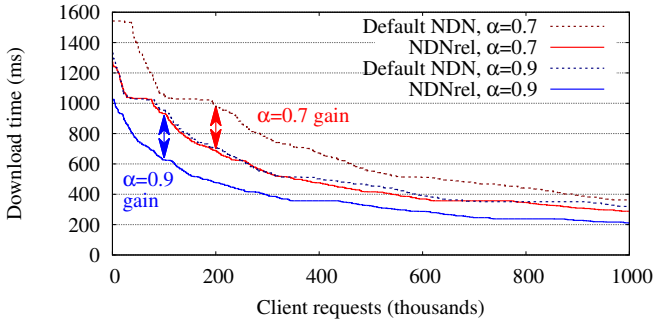
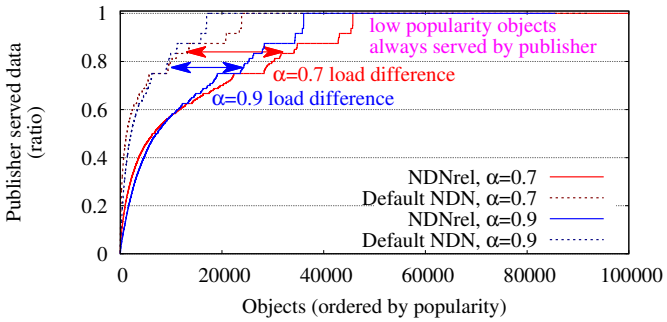


Figure 14: Network traffic

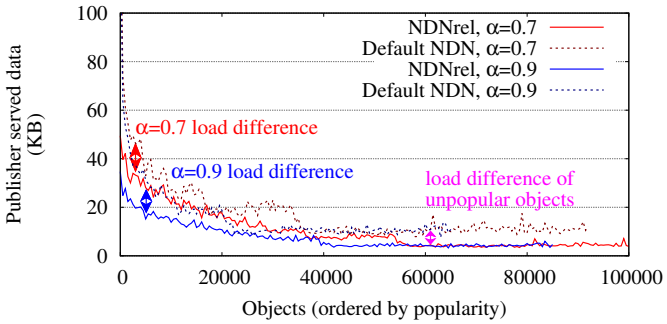
The results show that the download time of clients varies from 300 ms to 1.6 s in the default NDN scenario, with averages of 686 ms ($\alpha = 0.7$) and 531 ms ($\alpha = 0.9$). The observed variation occurs due to the characteristics of each object, especially its popularity and size. Popular contents are more likely to be found in the router caches than unpopular ones due to their higher request rate and NDN caching policies. Because contents cached on-path are closer to clients than the publisher, retrieving data from them reduces the download time. The impact of the size of an object is intuitive because larger ones require more time to be transmitted than smaller contents.

The use of relations reduces the range of clients download times to between 200 ms and 1.3 s. The benefit of relations is more visible in the average download time, which is reduced by more than 25%, achieving 513 ms ($\alpha = 0.7$) and 374 ms ($\alpha = 0.9$). The improvement is a direct result of relations, which increase even more the popularity of popular contents and leverages the NDN caching. Specifically, the routers push the popular objects closer to the clients and satisfy a higher number of requests (discussed later in Section 5.2.3). Regarding the manifest, the average time to retrieve it is only 43 ms (or approximately 10% of the download time), which justifies adding the relations in exchange for the benefits obtained.

The results of the publisher load, presented in Figure 15(b), demonstrate further the benefits of the relation mechanism. The horizontal axes represent each object ordered by the popularity while the vertical axes describe the two metrics measured: the percentage of requests satisfied (Figure 15(a)), and the volume of data (in Kilobytes) transmitted (Figure 15(b)) by the publisher. We expect that the use of relations will reduce the publisher load as a consequence of the higher usage of in-network caches to provide the requested contents, as previously discussed.



(a) Request ratio per object



(b) Data per object

Figure 15: Publisher load

The results of this analysis evidence two groups of objects regarding the providing source: those served by the publisher and in-network caches and those provided exclusively by the publisher. Content provided by an in-network cache reduces the publisher load because it does not need to process the request or send the content data. The results also show that the load reduction is more significant in the popular objects than in the unpopular ones because of their higher request rate.

In the default NDN scenario (for both values of α), approximately 25% of objects are provided partially by the in-network caches, and only 1% have more than half of their requests satisfied by them. The objects that do not benefit from the in-network caching, comprising 75% of the served objects, are provided exclusively by the publisher because of their unpopularity. To complement the analysis, we take a look at the volume of data transmitted by the producer, shown in Figure 15(b). The in-network caching helps alleviate the publisher load, keeping the transmitted data between 100 kB and 20 kB when providing the popular objects. Considering the average size of the objects (10 kB), the publisher sends only up to 10 copies of the popular ones to serve a high number of client requests. In comparison, the unpopular ones generate a traffic of 10 kB,

but they receive a significantly smaller number of requests.

Analyzing the NDNrel scenario, we conclude that the changes in objects popularity increase the in-network caching and, consequently, reduce the publisher load considerably. In Figure 15(a), it is possible to see that caches partially serve almost half of the contents, and around 10% of the objects have less than 50% of the requests provided by the publisher. The impact is also significant in the data traffic (Figure 15(b)), which is reduced to between 40 kB and 10 kB for the most popular contents and only 4 kB for the unpopular ones. Although the objects are smaller in this scenario because of the relations (around 4 kB), they also receive more requests. Therefore, the similar proportion of up to 10 copies sent to provide the popular contents means a more efficient usage of in-network caching to serve the data. Overall, the traffic generated by the publisher is reduced by more than 32%: from 1.6 GB to 1 GB ($\alpha = 0.7$), and from 1.1 GB to 0.7 GB ($\alpha = 0.9$).

In summary, *the use of relations reduces the client download time and the publisher load because of the better usage of the in-network caching to disseminate content.* The benefits are a consequence of the relations, which amplify the advantages of NDN to distribute popular contents in the network.

5.2.3. Network

The last part of the evaluation investigates the effects of the relation mechanism on the network, shedding light on the underlying reasons for the better performance using relations. During the analysis, we discuss the overall network traffic and the content placement in the network.

The total network traffic measured in each link during the content distribution is shown in Figure 16. The horizontal axis presents the topology links ordered by traffic while the vertical axis, the network traffic measured in Gigabytes. We expect that the relation mechanism will reduce the network traffic because of the better efficiency of in-network caching. This result is visible through the higher cache hit ratio and the placement of contents closer to the clients, discussed later in this section.

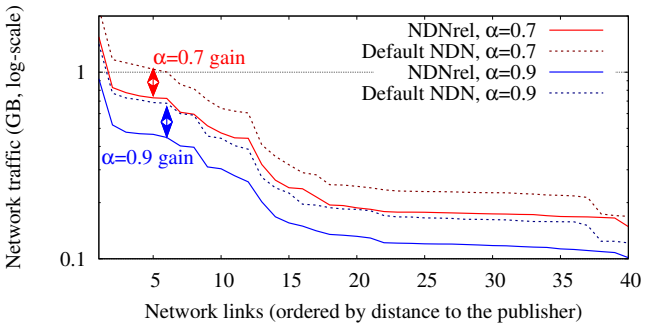


Figure 16: Network traffic

The results show a variation of traffic in the links according to their distance to the content publisher. In the default NDN scenario, the traffic on the links ranges from 2.21 GB to 0.08 GB. Intuitively, because the transmission creates a spanning tree with the shortest path, the links transmit not only the traffic of their local clients but also the aggregation of the lower levels of the tree. Therefore, the closer the link is to the publisher, the higher is its traffic volume. Overall, the content distribution in the default NDN scenario generated a total network traffic of 19.5 GB ($\alpha = 0.7$) and 13.3 GB ($\alpha = 0.9$).

The use of relations reduces the overall network traffic by 26.4% ($\alpha = 0.7$) and 30.7% ($\alpha = 0.9$), down to 14.3 GB and 9.2 GB, respectively. Link-wise, the reduction is proportional to its traffic, being more significant (regarding total volume) in the links closer to the publisher than those farther. The reduced traffic measured in the links spans between 1.55 GB and 0.06 GB and is a consequence of the better usage of in-network caching. Lastly, the retrieval of the relations corresponds to only 4% of the total traffic, which is insignificant given the overall reduction.

Figure 17 shows the distance that data served to requests traveled in the network. The horizontal axis presents each content request from clients while the vertical axis represents the average distance ratio of all interests sent. The distance ratio is calculated dividing the number of hops from the client to the closest content copy by the distance between the client and the publisher. This metric gives the percentage of the worst-case scenario path that a request needs to travel before finding the data. In the experiments, the median distance to the publisher is of 3 hops, with a maximum of 5 hops. We expect that relations reduce the distance from the requester to the closer copy because of the high cache hit ratio, especially on the leaf routers.

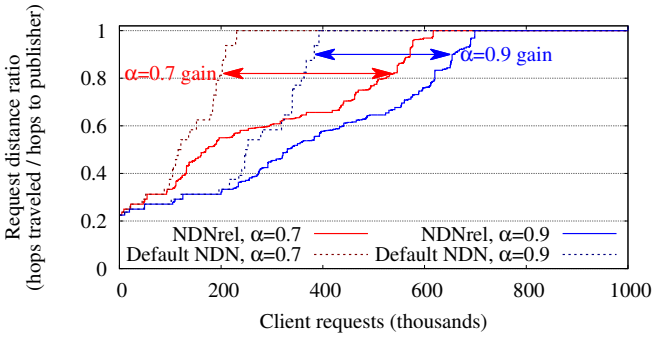


Figure 17: Distance to contents

The results confirm previous observations that the in-network caches provide part of the requests on behalf of the publisher. The average distance ratios of the requests in the default NDN scenario are 0.88 ($\alpha = 0.7$) and 0.77 ($\alpha = 0.9$). Because of the NDN caching mechanism, the contents are pushed closer to clients (especially the popular ones), reducing the distance that requests travel until the data is found. Overall, 23% of requests do not reach the publisher when $\alpha = 0.7$ and 39.2% when $\alpha = 0.9$.

With the relations scheme, the distance ratios reduce

to 0.75 ($\alpha = 0.7$) and 0.66 ($\alpha = 0.9$), as a consequence of the higher percentage of requests that do not reach the publisher. Up to 61.7% ($\alpha = 0.7$) and 69.9% ($\alpha = 0.9$) of requests find the data before reaching the publisher. The distance ratio reduction is another consequence of the higher efficiency of the caching mechanism and helps to explain the improvement of the client performance when using relations.

In summary, *relations reduce the total network traffic because of a more efficient in-network caching that causes objects to be retrieved closer by the clients.* The major part of the gains come from the popular objects, which are shared among different content variants, and leverage the NDN caching to disseminate the data.

6. Related Work

In this section, we first discuss related work that employs the concept of content and semantic relations at the application layer. Next, we focus on relation mechanisms proposed in the context for NDN and other ICN architectures.

The concept of relations is explored in different systems for content distribution. The most popular example are Web pages, which employ relations as a fundamental concept for the creation of complex documents with the use of multiple objects. In the context of multimedia, relations are employed to enable the creation of complex contents based on multiple audio and video channels. The DASH standard [17] employs a manifest file to specify a multimedia content with various components available in different HTTP URLs. The client can select a subset of these components for playback depending on its quality requirements. Also related to multimedia, SMIL [18] is a markup language used for the composition of rich multimedia presentations based on audiovisual elements stored in different objects. The above examples employ a specific format to describe how contents are formed with data acquired from different source objects. However, it is important to note that these formats are created targeting specific application domains and they are not applicable to other types of contents.

The literature of ICN includes three pieces of work (ICN-RE [19], NetInf [20], PSIRP/Pursuit Blackadder [21]) related to our NDNrel proposal.

ICN-RE [19] employs a concept similar to relations for implicit redundancy elimination of object data. In a nutshell, ICN-RE identifies, isolates and publishes byte-identical portions of different contents as a single object. The remaining parts are published individually. The mechanism uses a meta-object that lists the names of all objects that should be downloaded to rebuild the original content. ICN-RE uses the concept of relations in the meta-object to enable redundancy elimination of objects data. However, the format of this meta-object is strictly designed for the mechanism of implicit redundancy elimination. Further, ICN-RE requires modifications in ICN routers.

NetInf [20] introduces the concept of information objects (IOs), which are collections of metadata and pointers to actual data objects. The metadata contained in IOs allows clients to perform semantic queries about published contents. The PURSUIT project proposes a publish/subscribe architecture including an information-centric middleware [21] for the Blackadder prototype based on semantic technologies and metadata. The proposed middleware enables, among other things, establishing relations through common semantic attributes. The fundamental difference of these works on ICN is the focus on ICN architectures other than NDN. Thus, their findings cannot be directly extended to NDN due to specificities in naming (e.g., flat IDs under nested scopes) and routing mechanisms (e.g. separated control and data planes [22]). Further, those studies do not provide a detailed evaluation on the potential network performance gains when leveraging content and semantic relations among data objects.

In summary, our work is novel in exploring the design aspects of introducing backwards-compatible relations mechanisms for NDN to allow publishers modeling their contents in innovative ways. We argue that allowing publishers to use their knowledge about contents to define relations among objects can bring benefits beyond those achieved by mechanisms such as implicit decomposition. While current proposals for relations in content distribution are specific to their application domains (e.g. multimedia, P2P), NDNrel is designed to be generic and allow current and future ICN applications to natively benefit from their intrinsic semantic relations.

7. Final Remarks

NDN is a promising architectural proposal that enables efficient distribution of uniquely named data objects. Nevertheless, its full network performance gains require further investigation of methods to model distributed contents. In this paper, we further our analysis of a model that enables the distribution of contents as sets of related objects. The resulting mechanism, NDNrel, enables features such as content decomposition, object re-use, and efficient content updates. This paper is an extension of previous work [7] and presents additional considerations about the mechanism implementation and additional results about its performance.

To demonstrate the potential of NDNrel we employ extensive simulations on two case studies. The first explores relation-based decomposition in multimedia content distribution. Results have shown that NDNrel improves user performance, reducing client download times in an average of 34% and publisher request load in 56%. In turn, the second explores the use of relations for the composition of HTML contents for Web systems. Results show that, even in a scenario with small data objects and potential overhead from the relations metadata, NDNrel can reduce download times in an average of 28% and the publisher load in 34%. These improvements are a direct result of

higher hit-ratio of in-network caches due to changes in the request distribution of data objects. These effects occur because redundant data spread among multiple objects is eliminated of the network due to the use of relations.

Our simulation results show promising advantages in the use of the proposed relations model. Nevertheless, we intend to further explore these results through experimentation in a global-scale testbed deployment of or prototype implementation. Additionally, we intend to actively contribute with the efforts from the ICN research community to develop a specification for manifests in content distribution. Such specification can greatly benefit our proposal and various other mechanisms for networked applications based on content distribution.

References

[1] G. Zhang, Y. Li, T. Lin, Caching in information centric networking: a survey, Elsevier Computer Networks 57 (16) (2013) 3128–3141.

[2] M. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, B. Mathieu, A Survey of Naming and Routing in Information-Centric Networks, IEEE Communications Magazine 50 (12) (2012) 44–53.

[3] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos, A Survey of Information-Centric Networking Research, IEEE Communications Surveys & Tutorials 16 (2) (2013) 1024–1049.

[4] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon, I tube, you tube, everybody tubes, in: 7th ACM SIGCOMM Conference on Internet Measurement (IMC 2007), 1–14, 2007.

[5] X. S. Wang, A. Krishnamurthy, D. Wetherall, How Much Can We Micro-Cache Web Pages?, in: 12nd ACM Internet Measurement Conference (IMC 2014), 249–256, 2014.

[6] M. Mosko, G. Scott, I. Solis, C. A. Wood, CCNx Manifest Specification, Tech. Rep., Xerox PARC, URL `www.ccnx.org/pubs/draft-wood-icnrg-ccnxmanifests-00.html`, 2015.

[7] R. S. Antunes, M. B. Lehmann, R. B. Mansilha, C. E. Rothenberg, L. P. Gaspary, M. P. Barcellos, CCNrel: leveraging relations among objects to improve the performance of CCN, in: 14th IFIP/IEEE Integrated Network Management Symposium (IM 2015), 199–206, 2015.

[8] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A Survey of Information-Centric Networking, IEEE Communications Magazine 50 (7) (2012) 26–36.

[9] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. C. Schmidt, M. Waehlisch, ICN Research Challenges, Tech. Rep., IRTF, URL `tools.ietf.org/html/draft-irtf-icnrg-challenges-04`, 2016.

[10] Apache Software Foundation, Apache Subversion, URL `subversion.apache.org`, 2015.

[11] M. Mosko, CCNx Publisher Serial Versioning, Tech. Rep., Xerox PARC, URL `tools.ietf.org/html/draft-mosko-icnrg-ccnxserialversion-00`, 2015.

[12] I. Moiseenko, Fetching content in Named Data Networking with embedded manifests, Tech. Rep., NDN Project, URL `named-data.net/wp-content/uploads/2014/09/ndn-tr-25-manifest-embedding.pdf`, 2014.

[13] A. Afanasyev, I. Moiseenko, L. Zhang, ndnSIM: NDN simulator for NS-3, Tech. Rep., NDN Project, URL `named-data.net/wp-content/uploads/TRndnsim.pdf`, 2012.

[14] K. Pentikousis, B. Ohlman, E. Davies, S. Spirou, G. Boggia, P. Mahadevan, Information-centric Networking: Evaluation Methodology, Tech. Rep., IRTF, URL `tools.ietf.org/html/draft-irtf-icnrg-evaluation-methodology-03`, 2015.

[15] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet Topology Zoo, IEEE Journal on Selected Areas in Communications 29 (9) (2011) 1765–1775.

[16] G. Rossini, D. Rossi, Evaluating CCN multi-path interest forwarding strategies, Elsevier Computer Communications 36 (7) (2013) 771–778.

[17] ISO/IEC, Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats, URL `www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=65274`, 2014.

[18] D. Bulterman, J. Jansen, P. Cesar, S. Mullender, E. Hyche, M. DeMeglio, J. Quint, H. Kawamura, D. Weck, X. G. Pañeda, D. Melendi, S. Cruz-Lara, M. Hanclik, D. F. Zucker, T. Michel, Synchronized Multimedia Integration Language (SMIL 3.0), URL `www.w3.org/TR/SMIL`, 2008.

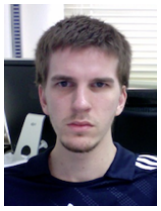
[19] D. Perino, M. Varvello, K. P. N. Puttaswamy, ICN-RE: Redundancy Elimination for Information-Centric Networking, in: 2nd ACM SIGCOMM Information-Centric Networking Workshop (ICN 2012), 91–96, 2012.

[20] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, H. Karl, Network of Information (NetInf) – An information-centric networking architecture, Elsevier Computer Communications 36 (7) (2013) 721–735.

[21] B. Tagger, D. Trossen, A. Kostopoulos, S. Porter, G. Parisis, Realising an application environment for information-centric networking, Elsevier Computer Networks 57 (16) (2013) 3249–3266.

[22] P. Jokela, A. Zahemszky, C. Rothenberg, S. Arianfar, P. Nikander, LIPSIN: line speed publish/subscribe inter-networking, in: 22nd ACM SIGCOMM Conference on Data Communication (SIGCOMM 2009), 195–206, 2015.

Rodolfo Stoffel Antunes is a Ph.D. student at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil. He holds a B.Sc. degree in Computer Science from the UNISINOS University (2009). His research interests include information-centric networks and large-scale content distribution systems. More information can be found at http://inf.ufrgs.br/~rsantunes.



Matheus B. Lehmann is a Ph.D. student at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil. He holds a B.Sc. degree in Computer Science from the Federal University of Rio Grande do Sul (2011). His research interests include information-centric networking, mobility support, peer-to-peer systems, and performance analysis. More information can be found at http://inf.ufrgs.br/~mblehmann.

Rodrigo B. Mansilha is a fourth year Ph.D. student in Computer Science at Federal University of Rio Grande do Sul (UFRGS), Brazil. His research interests and background are in the field of large-scale content distribution systems, specifically ICN (currently) and P2P networks (in the past). For details, see http://inf.ufrgs.br/~rbmansilha.

Luciano Paschoal Gaspary holds a Ph.D. in Computer Science (UFRGS, 2002) and is an Associate Professor at the Institute of Informatics, UFRGS. Prof. Gaspary has been involved in various research areas, mainly computer networks, network management and computer system security. He is author of more than 120 full papers published in leading peer-reviewed publications and has a history of dedication to research activities such as organization of scientific events, participation in the TPC of relevant symposia, and participation as editorial board member of various journals. His research interests include network, service, and application management, network virtualization, software-defined networking, cloud computing, security of large-scale distributed systems (e.g., P2P), IT service management, and intrusion detection and prevention. More information can be found at http://lattes.cnpq.br/3059640410928425.

Marinho P. Barcellos received BSc and MSc degrees in Computer Science from Federal University of Rio Grande do Sul (1989 and 1993, respectively) and PhD degree in Computer Science from University of Newcastle Upon Tyne (1998). Since 2008 Prof. Barcellos has been with the Federal University of Rio Grande do Sul (UFRGS), where he is an Associate Professor. He has authored many papers in leading journals and conferences related to computer networks, network and service management, and computer security, also serving as TPC member and chair. He has authored book chapters and delivered several tutorials and invited talks. His work as a lecturer has been consistently distinguished by graduating students. Prof. Barcellos was the elected chair of the Special Interest Group on Computer Security of the Brazilian Computer Society (CESeg/SBC) 2011-2012. He is a member of SBC and ACM. His current research interests are cloud computing data center networks, software-defined networking, information-centric networks and security aspects of those networks. He is the General Co-Chair of ACM SIGCOMM 2016 and TPC Co-Chair of SBRC 2016. More information can be found at http://www.inf.ufrgs.br/~marinho.