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

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.

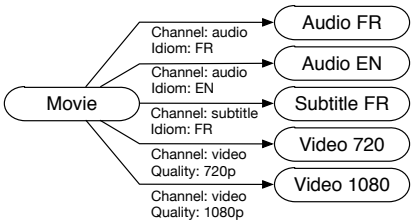

Figure 1. Multimedia modeling example

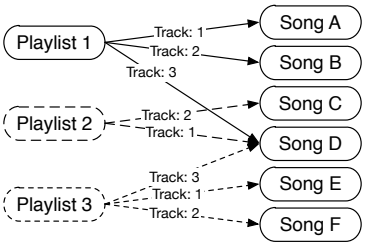

Figure 2. Playlist modeling example

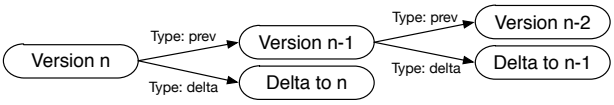

Figure 3. Log file modeling example

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

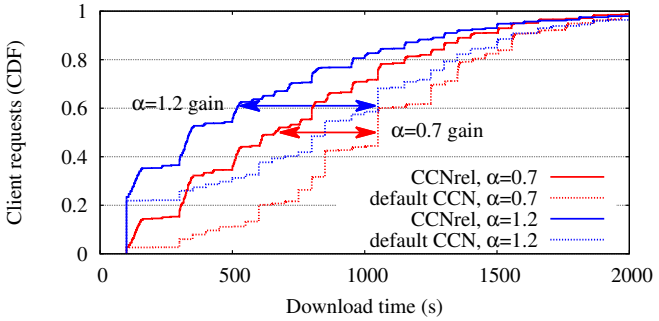

Figure 4. CDF of clients download time

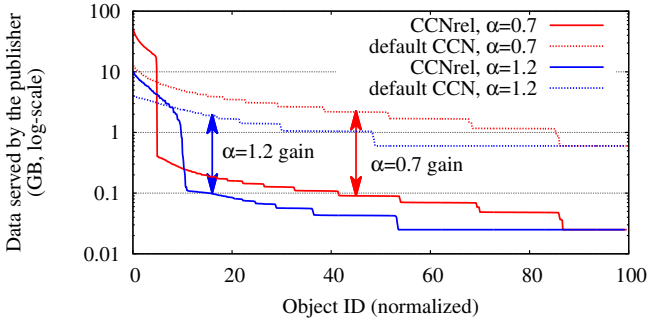

Figure 5. Publisher load

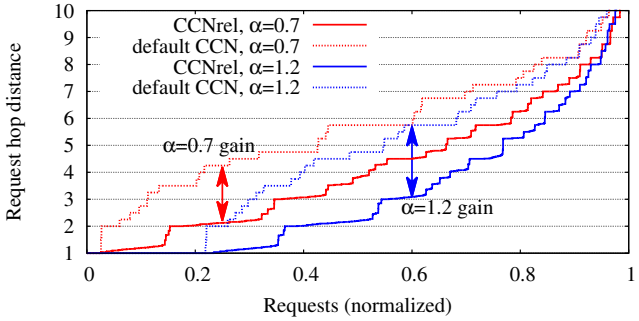

Figure 6. Request hop distance

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.

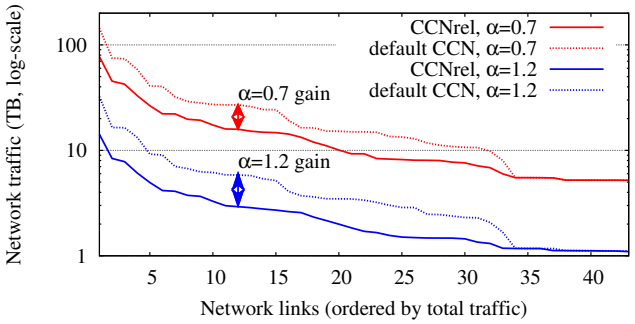

Figure 7. Total network traffic: (average gain of 34% and 42%, shown in log scale)

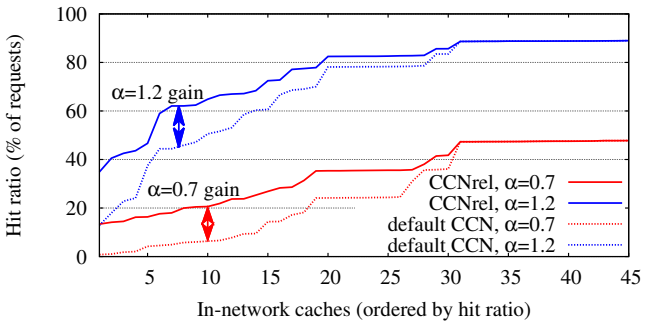

Figure 8. Cache hit ratio

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:

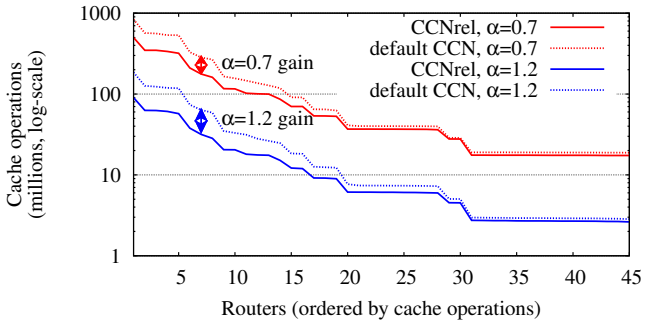

Figure 9. Caching operations: (average gain of 32% and 44%, shown in log scale)

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

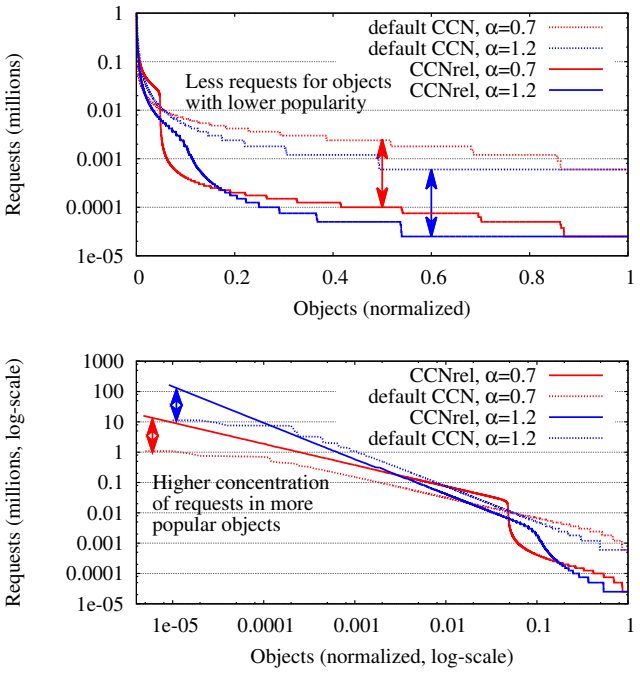

Figure 10. Object request distribution

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.