Imperial College London

Department of Computing

Future Internet: Information Centric Networks

Author:
Henry Williams

Supervisor:
Professor Kin K. Leung

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

September 2016

Abstract

Today's Internet only just works and networking engineers know it. Consumer usage has evolved away from the accessing of identified hosts to an interest in obtaining vast amounts of information, irrespective of its physical location. This shift in purpose, together with the increased need for enhanced mobility and security solutions, has lead to a search for solutions to salvage the future of the Internet. Information Centric Networking (ICN) is one such proposal.

In this paper, we first of all review the current state of Information Centric Networking, condensing the extensive existing research into a manageable survey. In particular, we focus our attentions on caching mechanisms in ICN. Secondly, we seek to rebuff the criticism of ICN's ubiquitous caching as not offering significant performance gains over Internet Protocol solutions. In particular, we refute the proposition that the edge caching provided by Content Delivery Networks, performs just as well as pervasive caching. To do this, we highlight the fact that much of the literature, including these critiques, fail to consider the benefits of optimising in ICN over name-routing and caching simultaneously. We run our own simulations which demonstrate that if you consider the upper bound of ICN capabilities, as opposed to the lower bound, then continued research into ICN is justified. Finally, accepting the possibility of optimal routing in ICN, we create our own novel caching mechanism, Filter First Caching (FFC), that focuses solely on reducing caching redundancy. With FFC, we introduce the concept of multi-layered filtering to ICN in-network caching.

Acknowledgments

Many thanks to Professor Kin K. Leung for leading me towards this fascinating area of research and allowing me complete flexibility on my approach. Your positive attitude is forever an inspiration.

In addition, I wish to thank (indirectly since no contact was made) the ccnSim development team comprising of Andrea Araldo, Raffele Chiocchetti, Emilio Leonardi, Dario Rossi, Giuseppe Rossini and Michele Tortelli. Their open-source simulation tool allowed me to run content-centric simulations and build on their existing code-base.

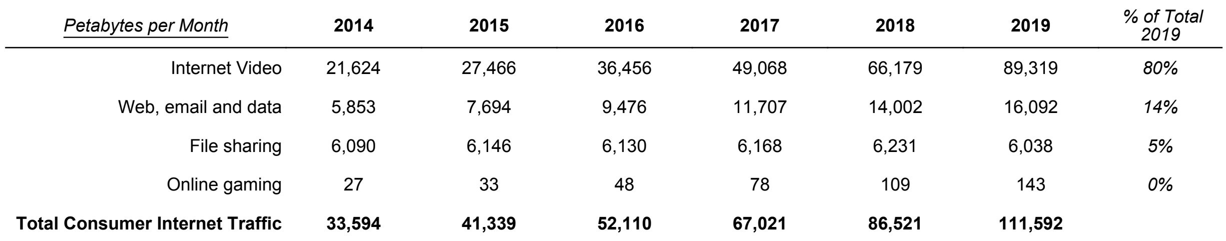
Contents

Chapter 1

Introduction

In the middle of the 20th Century, the Advanced Research Projects Agency (ARPA) initiated the construction of an architecture which the modern world has come to rely on. Their goal was to provide a host to host connection between distinct locations. This would facilitate more reliable communication and the efficient sharing of scarce and expensive resources, such as supercomputers or high-speed tape drives. Those principles developed through the 1960s and 1970s provide the bedrock on which contemporary Internet thrives.

However, society's needs have changed. Nowadays, users are more interested in accessing the published content, in particular web pages and multimedia, rather than interacting with remote devices. In fact, Cisco [7] predicts that video will be 80 percent of all consumer Internet traffic by 2019, up from 64 percent in 2014. What's more, the estimated compound annual growth rate of Internet Video from 2014 to 2019 is 33%. To emphasize the affect this will have on existing architectures, by 2019, nearly a million minutes of video content will cross Internet Protocol (IP) networks every second.



Petabytes per Month	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	*% of Total 2019*
Internet Video	21,624	27,466	36,456	49,068	66,179	89,319	*80%*
Web, email and data	5,853	7,694	9,476	11,707	14,002	16,092	*14%*
File sharing	6,090	6,146	6,130	6,168	6,231	6,038	*5%*
Online gaming	27	33	48	78	109	143	*0%*
Total Consumer Internet Traffic	**33,594**	**41,339**	**52,110**	**67,021**	**86,521**	**111,592**	

Figure 1.1: Cisco Expectations of Global Consumer Internet Traffic (PB per month)

In other words, Internet consumers today are more interested in *what* the Internet offers as opposed to who is providing the content and thus *where* that content is located. The host-centric nature of the prevailing IP no longer suits the content-centric consumption that it seeks to serve. Much research has gone into overcoming the limitations of the classic host-to-host Internet model in the face of increasing demand and incumbent patches include Content Delivery Networks (CDN), Peer-to-Peer architectures, and cloud-based platforms [19]. However, each of these IP overlays require expensive and complex middleware to function and generate their own issues in terms of scalability, security, robustness and mobility [62, 25, 12].

As such, some leading pioneers in networking propose a more clean-slate alternative with the innovative field of Information Centric Networking (ICN). In this approach, it is the content objects themselves that become the central units of information processed: routing is done by uniquely *named-data* rather than to a named-host. Furthermore, ICN provides in-network caching to alleviate the projected strain on the network and the traffic load at the server side. Aside from supporting the increasing request for video exchange and large file transfers, a content-centric framework is more applicable in new areas of research such as the Internet of Things, sensor networks and vehicle networks where multi-homing, multicast and mobility are essential [59].

Despite the many advantages that ICN appears to provide, there remain sceptics who believe that this novel architecture does not justify the wide-spread changes required for implementation. Principally, there are those [24, 28, 31] that point to the minimal performance improvement of in-network caching over the edge caching used in CDN. This central element is one of the key drivers of a content-centric framework so these results shake the foundation of ICN before it has even begun. However, in this paper we argue that this negative research, and indeed most positive research, severely underestimates the capabilities of ICN. We stipulate that only if you couple caching and name-routing decisions can you understand the true benefits available. Moreover, we go on to show that at the upper bound of optimisation, ICN caching offers substantial performance gains. In this way, we support the case for ubiquitous ICN caching over existing edge IP caching overlays such as CDN. We hope that our results justify continued funding and research into ICN, as a solution to the struggling Internet.

Having shown the benefits of in-network caching using existing mechanisms, we seek to improve performance through the introduction of our own novel caching policy. Once we assume that an optimal name-routing policy is available then we can focus the mechanism entirely on lowering caching redundancy and avoid costs for improving caching availability. With this in mind, we develop Filter First Caching (FFC). FFC uses a multi-layered filtering technique, whereby content is only stored in a router's main cache if it's name can be located in a pre-filtering name cache. In this way, rare and unpopular requests are filtered out of consideration, avoiding unnecessary cache pollution. Building on this design, we extend FFC to incorporate benefits of other simple caching mechanisms. For example, Trickle Filter Caching (TFC) combines the trickle-down benefits of an existing policy, Leave Copy Down, with the concept of multi-layered filtering. Likewise, FFC(p) incorporates the successful randomised element of Fix(p) into a multi-layered filtering environment. With these novel solutions, we hope to show that continued innovation and research into ICN can increase the potential gains over existing IP solutions still further.

1.1 Contributions

The main contributions of the paper are:

- We summarize the extensive existing research of ICN in one up-to-date and manageable survey.
- We categorise, address and evaluate the wide range of available caching mechanisms.
- We show that only with an optimisation of name-routing policy and caching mechanism simultaneously can the true benefits of ICN be understood. Having shown the potential for significant performance gains, we justify the continued support of ICN as a solution.
- In an attempt to further optimise ICN in-network caching performance, we propose a novel two-layered caching mechanism focused on lowering caching redundancy.
- We provide real-world simulations to show the validity of the propositions.

Our goal in this paper is not to focus on the specific ICN architectures, but to analyse the benefits arising from the principles underlying ICN. It is fair to say that the significant potential changes to the existing network that ICN will give rise to, can only be justified if they have been shown to offer substantial performance improvements.

Our motivation is to support the ongoing research of ICN and demonstrate it's potential is worth continued innovation. What's more, we hope to introduce the concept of multi-layered filtering as a viable in-network caching mechanism for ICN.

1.2 Report Outline

The remainder of this paper is organized as follows:

In Chapter 2, we summarise the key elements of ICN before outlining the existing proposals and taking an in-depth look into a complete ICN architecture. This enables us to contextualize the individual components and the challenges faced by researchers.

Chapter 3 will expand on some possible caching mechanisms including identification of those suitable for further analysis and the metrics for their evaluation.

Chapter 4 sets out the implementation for a suitable network simulation, the basis of which can form the scenarios run in later sections to justify the key contributions of the paper.

Chapter 5 contains the main contribution to the field of this paper with an analysis of the coupled nature of caching mechanisms and name-routing policies in ICN. In this chapter, we look to show the upper bound potential of ICN that much of the negative literature has failed to consider.

Chapter 6 attempts to optimise performance potential of in-network caching through a novel caching mechanism utilising the benefits of multi-layered filtering.

Chapter 7 concludes the paper evaluating the project success and highlighting future work.

Chapter 2

Information Centric Networking

In this Chapter, we explain how Information Centric Networking (ICN) introduces uniquely named data as the core Internet principle. First of all, we review the basic terminology with which we will talk about ICN initiatives. Then, we give brief descriptions of the key elements of ICN. With this knowledge, we then look at ICN as a whole with an example request for content. Finally, we consider existing architecture proposals and review one in-depth to contextualise the complexities.

To help understand the overarching conceptual difference between IP and ICN, we compare an example of YouTube content distribution in figure 2.1. For IP, note that CDN servers are placed close to the consumers to reduce latency but that the fundamental abstraction remains a host-to-host connection between Producer (source) and Consumer (user). This is illustrated by the unbroken line, representing a TCP-like connection (Transmission Control Protocol). Despite the request for content being satisfied by the nearby CDN server, the main YouTube server is aware of the transaction, since the secure connection is the only way of ensuring the validity of the content in IP.

On the other hand, with ICN, each packet (piece of content) is uniquely named and can be independently validated. As such, any node can answer a request for that packet within the network without notifying the main YouTube server. This transparent process can include communication across Internet Service Providers (ISP) to locate nearer replicas of requested content, illustrated by the connecting line between ISP 5 and 6.

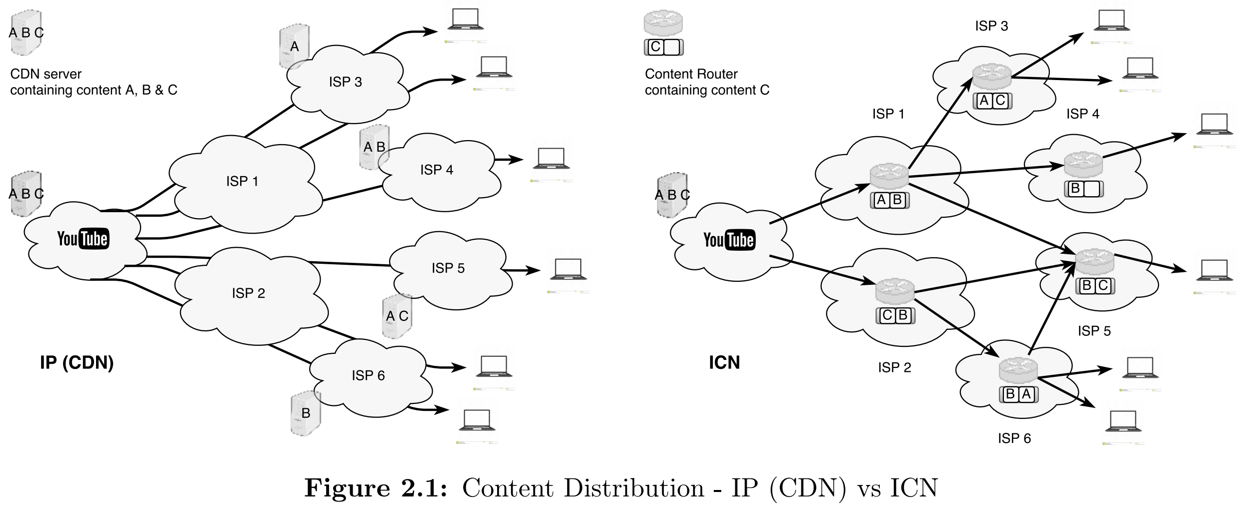


Figure 2.1: Content Distribution - IP (CDN) vs ICN

2.1 Terminology

Since ICN is a growing research area with distinct contributions from around the world, there is a corresponding diversity in terminologies for different design elements [35]. For the sake of this document, we unify those terminologies for their common elements.

There are three central roles in any ICN network. Any device on the network may play one or more of these roles at the same time: although usual circumstances have a single device acting as a single role at a point in time.

1. **Producers** are end applications that originate the content *data* to be made available to the public. These are sometimes referred to as publishers.
2. **Consumers** are end applications that originate the *requests* for data to be sent out over the network to access published content. These are sometimes referred to as subscribers.
3. **Content Routers (CRs)** refer to the in-network equipment that forward both *request* and *data* packets. They are also installed with a cache, enabling them to store content within the network itself. This differs them from standard IP routers, with which they can still operate alongside in many of the proposed architectures.

   For ease of understanding, we divide the roles of the Content Router into the **data plane**, the part responsible for managing incoming *request* and *data* packets, and the **control plane**, the part responsible for populating the routing tables.

Requests for content are also known as *interest* packets [34] as they signify the consumer's interest in the corresponding *data*.

The basic *data* unit transmitted over ICN is called a **Named Data Object (NDO)**. A NDO is a response to a *request* with the matched name. NDOs can be of different granularity (e.g. a piece of content, or packets comprising content). For example, content A may be divided into 5 chunks, named by A/C_1, A/C_2, A/C_3, A/C_4 and A/C_5, respectively. In order to access the content, a Consumer will then issue 5 *request* packets, each of which corresponds to one chunk of the content. Most architectures include an algorithm to collate all required *requests* for selected content. Thus, users that operate Consumer applications appear to make only a single *request*.

2.2 Key Research Areas

There are various existing ICN projects that attempt to design a content-centric alternative to the prevailing host-centric system. These are well surveyed in [11, 35, 44, 60, 64] and collectively they offer an insight into a set of key functionalities for any successful ICN architecture, together with possible solutions. In this section, we seek to understand the basics of all of these core functions, both independently and how they relate to each other, in order to assess any possible initiatives. In section 2.3, we illustrate the totality of these elements with an example request in a simple ICN framework.

2.2.1 Naming

The fundamental characteristic of all ICN architectures is the concept of *named data*, wherein each piece of content is given a unique name which is location independent. The given name must be persistent in terms of spatial and temporal dimensions. Most approaches advocate either a flat naming scheme or a hierarchical scheme.

An ICN **flat naming scheme** was first used in the Data Oriented Network Architecture (DONA) [37] and it has since been adopted by various subsequent architectures [38, 29]. In DONA, every piece of content is directly associated with a Producer and each name is in the form:

$$P : L$$
$$P = \text{cryptographic hash of the Producer's public key}$$
$$L = \text{an unique label assigned by the Producer (a name)}$$

Each piece of content is sent with meta-data, which includes the full public key and digital digest signed by the Producer. In this way, the P element ensures provenance (reliability of source) while the signature in the meta-data ensures the content's integrity itself. Note that it is the Consumer who is responsible for verification of the public key. There is a lot of ongoing research into how ICN can assist in this venture for end-users, who are often naive to security demands.

As you can see, flat names greatly assist in self-certification, since provenance and integrity are validated by the name itself. However, it is also clear that the scheme will suffer in terms of scalability, with limited labels (L) available. What's more, regulation of labels must be strict, since each one must be unique. Finally, given these issues in lack of label flexibility, names will not be human readable. This is highly problematic as the naming scheme now requires an extensive lookup system so as to be functional for actual Internet users (similar to the current Domain Name System). This serves to create a point of weakness as already exists in IP.

A **hierarchical naming scheme**, which is more in line with Uniform Resource Locator (URL) addressing, facilitates human readability and flexibility in the routing strategy. This comes at the cost of additional overheads and potential lack of persistence due to movements in the hierarchy. Figure 2.2 illustrates the multiple components of names, arranged in a hierarchy. This is the scheme used in Named Data Networking (see section 2.4.1). Note that most hierarchical names will implicitly contain a Secure Hash Algorithm digest of the content (typically 256) for resolving ambiguity and ensuring uniqueness.

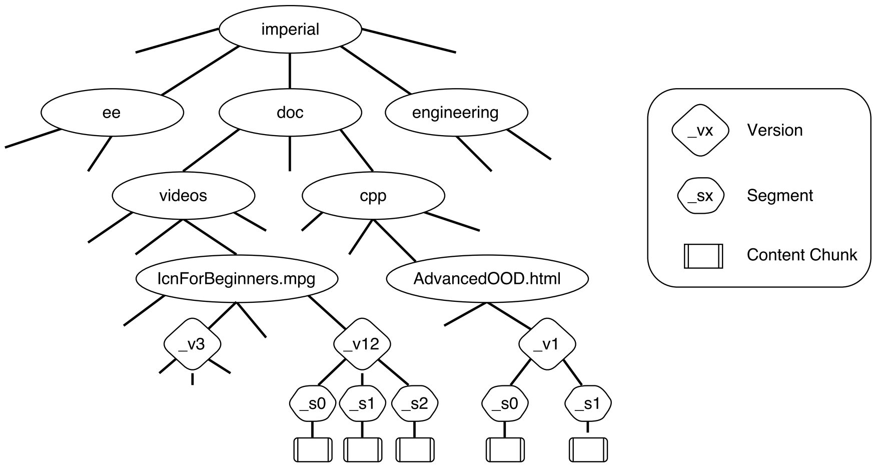


Figure 2.2: Example hierarchical naming scheme used in some ICN architectures

One key advantage of hierarchical naming is the flexibility to make naming decisions on a packet or whole object level. As you can see in figure 2.2, some architectures attach a segment number to the name itself to indicate the NDO granularity. Further to this, version control can be integrated into the naming scheme to ensure that users are experiencing the most recent release of the requested content. Verifying that a user's request is satisfied with the most up-to-date content is one of the more difficult features to overcome with in-network caching.

There have also been proposals advocating a **hybrid** approach. For example, the Scalable and Adoptable Internet Solution (SAIL) [4] ICN architecture uses flat naming for data object matching while it can use a hierarchical naming structure for routing [23]. It is likely that the future of ICN rests in a suitable hybrid approach, since both individual schemes are fallible on their own.

2.2.2 Name Resolution and Data Routing

Name resolution, along with data discovery, involves matching the content name to a source that can supply the content. In some architectures, this involves a request forwarding strategy (e.g. longest prefix match) while others rely on a DNS-like lookup. As with the existing Domain Name System (DNS), lookup name-resolution systems can be problematic and often result in being the most common point of failure [17]. In addition, the selected naming scheme (flat or hierarchical) is clearly integral to name-resolution. While it may seem that hierarchical names are more suitable for both DNS-like and prefix matching resolution, there are promising alternatives for flat naming schemes using multi-level distributed hash tables [23].

Data routing (or data delivery) refers to the construction of a path across the network on which to transfer the content to the user from the identified source. This corresponds to the routing of packets from a server host to an end-user host in an IP context. The difference being that the packet may be originating from a cache within the network as opposed to a dedicated server.

As a whole, the routing steps for a single download in an ICN scenario may consist of:

1. **name resolution step:** translates the unique name of the requested NDO into its locator;
2. **discovery step:** routes the request to a source for the data object based on the locator;
3. **delivery step:** routes the data object back to the Consumer.

We consider these elements together as their functionality can be integrated within the architecture, *coupled*, or operate independently, *decoupled.*

An example coupled approach might reverse the data along the same path on which the request was routed to the source. In the coupled scenario in figure 2.3, we can see that since the request has been routed to the Producer via CR2 (discovery step), then the data is routed back along that same seemingly inefficient path (delivery step).

In the decoupled approach, the name resolution system will not determine the data routing path. This means that the data may be sent over a more efficient route according to an independent routing module. In figure 2.3, we can see that being implemented as the data packet is routed back to the Consumers directly from CR3 to CR1. While decoupled approaches appear to be more efficient, they often require additional middleware, such as a Resolution Handler, or an unrealistic knowledge of intra-domain connections at each content router.

In terms of the actual routing scheme (i.e. identifying the next hop for a request or data packet to travel along) there are a wide variety of algorithms. Many of the architectures propose to tweak

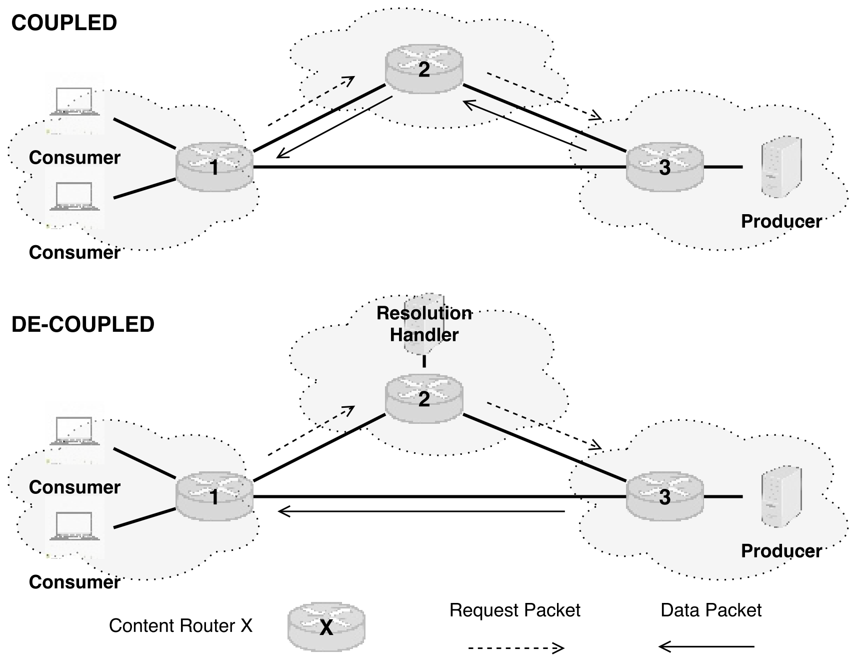


Figure 2.3: Content Distribution - IP (CDN) vs ICN

existing IP routing protocols such as Border Gateway Protocol (BGP) or Open Shortest Path First (OSPF). Most of the concepts range between these two extreme design concepts [22]:

1. **exploitation:** each content router's routing table has complete knowledge concerning the placement of the copies of any content in the network;
2. **exploration:** routing tables have no knowledge and simply flood requests with appropriate Time To Live (TTL) while interest aggregation is assured to avoid looping.

A hybrid approach may mean that routers have partial knowledge allowing some exploitation-based approach, alongside some exploration-based approach. For example, exploration may be used for the k^{th} percentile of the popularity request distribution, since the most popular content is likely to be stored in the network (varying k can be done based on the routing policy, topology popularity distribution etc.).

To actually achieve performance gains under one of these designs, routing algorithms require inter-node communication. This coordination can either be explicit or implicit. Explicit schemes tend to achieve near optimal routing performance but at the cost of considerable overheads in terms of knowledge storage and network transmissions required to maintain that knowledge. On the other hand, implicit schemes achieve performance gains without constant exchange of information. They rely on different criteria to map content requests to nodes storing local replicas such as: static priority, minimum load, fastest response, round-robin, random split, hash based [56]. There is substantial evidence to suggest that implicit schemes are more suited to ICN given the exorbitant overheads of constant communication.

On a final note, the volume of content exchanged in the Internet and the volatility of cached content within the network pose significant challenges to the identification of a scalable routing protocol. In addition, the distribution among caches of content, and therefore the potential for congestion reduction, is clearly impacted by the chosen name resolution and data routing policies [22]. This central feature is intuitively obvious but often overlooked in ICN caching research. Chapter 5 seeks to illustrate that, when considering the benefits of ICN, this is a mistake worth avoiding.

2.2.3 Caching

Since content can be validated by it's unique name, there is no requirement to certify the authenticity of the nodes themselves. As such, every ICN node can act as the content's source as opposed to just the original host server. Based on this fundamental difference to IP, ubiquitous in-network storage is possible with the simple addition of Content Routers to the network infrastructure. Each Content Router, or end-user device, can check whether it has the requested named-content in its co-located cache. This means that any node on the network can satisfy the Consumer requests, potentially avoiding unnecessary additional congestion towards the Producer server.

Put simply, in-network caching can improve network performance by sourcing content from nodes closer to end-users. For static files, this could result in optimal delivery. While for dynamic data, the improved multicast capabilities and enhanced packet retransmission can also reduce the load on the network, particularity in avoiding repeat requests through bottlenecks. For example, once a packet has passed through a congested area, it may well be cached in the network beyond said congestion. Thereby, we avoid repeated strain should retransmission be required.

The potential benefits of in-network caching include:

- **lower latency:** cached content near the Consumer can be fetched quicker than from the Producer server;
- **reduced network traffic:** content traverses fewer links when there is a cache hit and the same object copy can be delivered to multiple Consumers, with no need to propagate numerous requests throughout the entire network to the Producer's server;
- **reduced server load:** with Consumer requests being satisfied by local caches, Producer servers have less workload reducing the likelihood of a bottleneck of service;
- **improved Quality of Service:** retransmission requests could be answered by an on-path cache, potentially on the other side of a disrupted link.

Given the potential benefits, much thought has gone into the optimisation of in-network-caching. Indeed, there are numerous individual caching considerations when constructing an efficient design. These include but are not limited to:

- **caching decision policy / caching mechanism:** which content should be cached at a router (e.g. should the arriving data packet be cached or not);
- **cache replacement policy:** which content should be removed when new content arrives at a full cache;
- **cache placement:** where caching routers should be placed (e.g. edge caching vs ubiquitous caching, nodes with more important links);
- **caching capacity:** how many request and data packets can a Content Router cache at any time;
- **request selection policy:** which requests to reject when the request buffer is full (this is particularly important in consideration of Denial of Service attacks);
- **detecting staleness of cached content:** how can a router verify that it's cached content is the most up-to-date version of that content before potentially satisfying a request with stale data;

- **detecting volatility of content:** caching packets that are less likely to be updated (e.g. the consistent elements of a popular web-page) will reduce overall network traffic and maximise the benefit of in-network caching;
- **implementation feasibility:** all decisions concerning Content Routers must consider hardware and software capabilities and costs.

In theory, the caching process is transparent to both the Consumer and Producer, who are unaware from where a request is satisfied. However, in practice, there may need to be a central notification centre, or a similar supervising device, to monitor content consumption for both legal content use and fee calculation purposes. This hints upon the issues of practical implementation from a business perspective, that we will only mention in passing, but remain one of the more unresolved barriers to progress [43, 53].

Finally, it is worth noting the potential effects of in-network caching on the quality of service functions of the transport layer. For example, out of order delivery is expected where parts of a data object will be returned from within the network. On top of this, existing flow and congestion control protocols require reliable Round Trip Time measurements which may not be possible with intermittent packets being returned from various in-network locations. While these issues are far from insurmountable, it helps to highlight the whole-scale changes required to implement an ICN architecture and the inter-linked nature of each decision.

Note that we illustrate caching in practice in section 2.3.

2.2.4 Mobility

Consumer mobility is inherent in ICN architectures where lengthy TCP subscriptions with end-to-end identifiers are not required. In short, there is no need to bind a layer 3 address to a layer 2 MAC address. This allows intelligent routers to take advantage of multiple interfaces and adapt to rapid changes. Much of the research in this area focuses on maximising the benefits achievable from native ICN primitives, as opposed to identifying solutions.

On the other hand, the effects of **Producer mobility** will depend on the naming and routing schemes. For example, any name resolution system or individual routing tables will need to be updated should the top-level source be re-located. Solutions fall into three main categories [15]:

1. **rendez-vous based:** involving a resolution of identifiers into locators performed by dedicated network nodes;
2. **anchor-based:** where a fixed network node is kept aware of mobile node movements and redirects packets appropriately;
3. **anchor-less:** where the mobile node itself is directly responsible for notifying the network about its movements.

This paper does not address further considerations on mobility which often relate to specific application use, for example in the Internet of Things [59].

2.2.5 Security

Current IP networks trust content based on the server id and connection type used for retrieval, forcing consumers to retrieve content from a prescribed source to trust it. Since ICN does not require host identification, it does not suffer the same IP security issues. For example, since each node can independently validate named-content, object authenticity can be ascertained at ingress gateways preventing standard denial-of-service attacks. Indeed, since the name-content bindings can be validated, the nodes themselves do not require authentication.

Yet, ICN does of course give rise to possible weaknesses of it's own [8, 9, 26]. It is clear that many of these are contingent on the naming scheme. Hierarchical name structures are partially human readable, assisting in usability. However, they also require an established trust relationship to ensure that the content does indeed correspond to the requested name. On the other hand, flat name structures support self-certification but require an alternative trust system to map human-readable names to flat names (similar to DNS but on a far larger scale).

The main areas of ICN security research include but are not limited to:

- **data integrity and origin authentication:** verify content is what it says it is;
- **binding NDOs to real-world identities:** verify publisher of content
- **access control and authorisation:** providing consumers with correct level of access to individual NDOs without end-to-end verification;
- **encryption:** distribution and management of encryption keys (for example required as solution to providing Consumers with access to correct data);
- **interest flooding:** similar to a contemporary Denial-of-Service attack, content routers can be flooded with interests exhausting network resources (without recourse since origin of requests are inherently unknown without end-to-end connection).

This paper only touches on security since most related discussions would need a dedicated study on the subject.

2.3 Overview

Having reviewed the basic principles of ICN, we can diagrammatically walk through a request for content. Figure 2.4 shows an example 7-node network in an ICN setting at arbitrary time *t*. It is assumed that all Content Routers link to Consumers while CR2, CR5 and CR7 link to the the Producers for content A, B and C, respectively. In any simple ICN scenario, each piece of content has a name and at least one Producer, a known location where that content is permanently stored and from which that piece of content can be requested.

At first glance, it would appear that the Consumer at CR1 has made a request for content B, via CR3, which has been satisfied by the Producer at CR5 and B has subsequently been cached in all en-route caches (CR5, CR4, CR3, CR1). This would suggest that the caching decision policy is the default of leaving a copy everywhere. Equally, the Consumer linking to CR2 looks to have made a request for content C. Of course, it may well have been the case that CR4 made the first request for content C. A subsequent request from CR2 for content C would have been satisfied by CR4 having matched the name within it's cache. It is this in-network caching, together with the identification

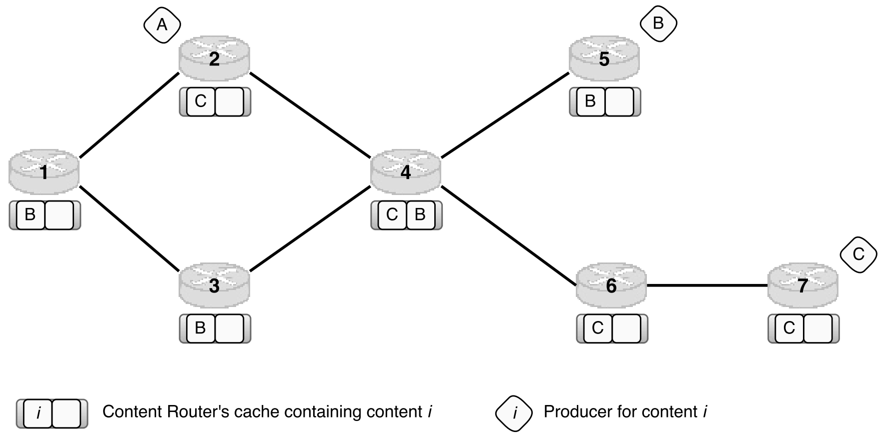


Figure 2.4: ICN scenario at time *t*

of the nearest replica to the Consumer through routing policies, that we seek to optimise in this paper.

Figure 2.5 now illustrates the affect on our simple network after a Consumer at CR5 makes a request for content A between time *t* and time *t + 1* (1). We can see that content C has had to be evicted from the cache at CR4 to make room for the passing-through content A (3-4). The content to be overwritten, in this case content C, would have been selected based on the cache replacement policy. The default policy is Least Recently Used (LRU) since it requires minimal computation and has been proven to work effectively [39].

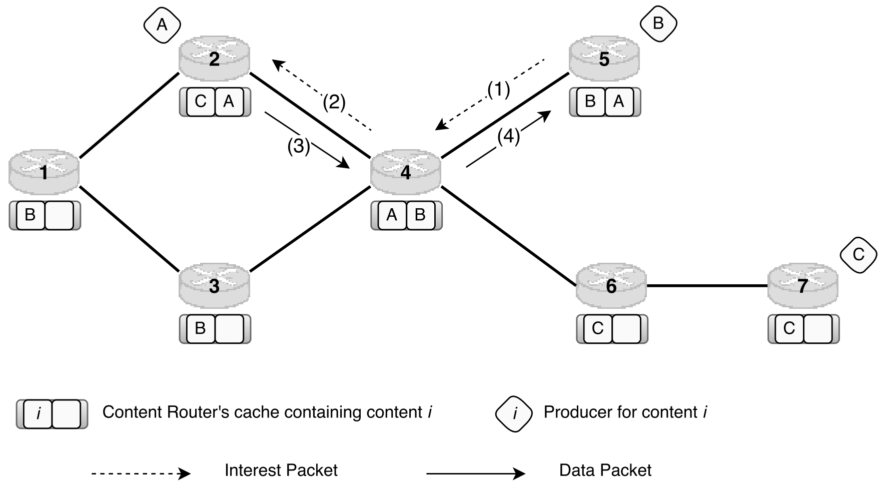


Figure 2.5: ICN scenario at time *t + 1*

To reiterate the value of in-network caching and naively demonstrate the link to name-resolution, we ask you to imagine a request for content A, from a Consumer at CR3, at time *t + 2*. Having now a basic understanding, one would naturally assume that the Consumer's request is sent to CR4 who satisfies it with content A. However, with the most basic name-resolution and caching mechanisms, there is no way that CR3 can be aware that content A is cached in CR4. As such, it may well send it's request to the Producer of content A at CR2 (required to be known as with IP) via CR1 thereby avoiding any potential benefit of in-network caching.

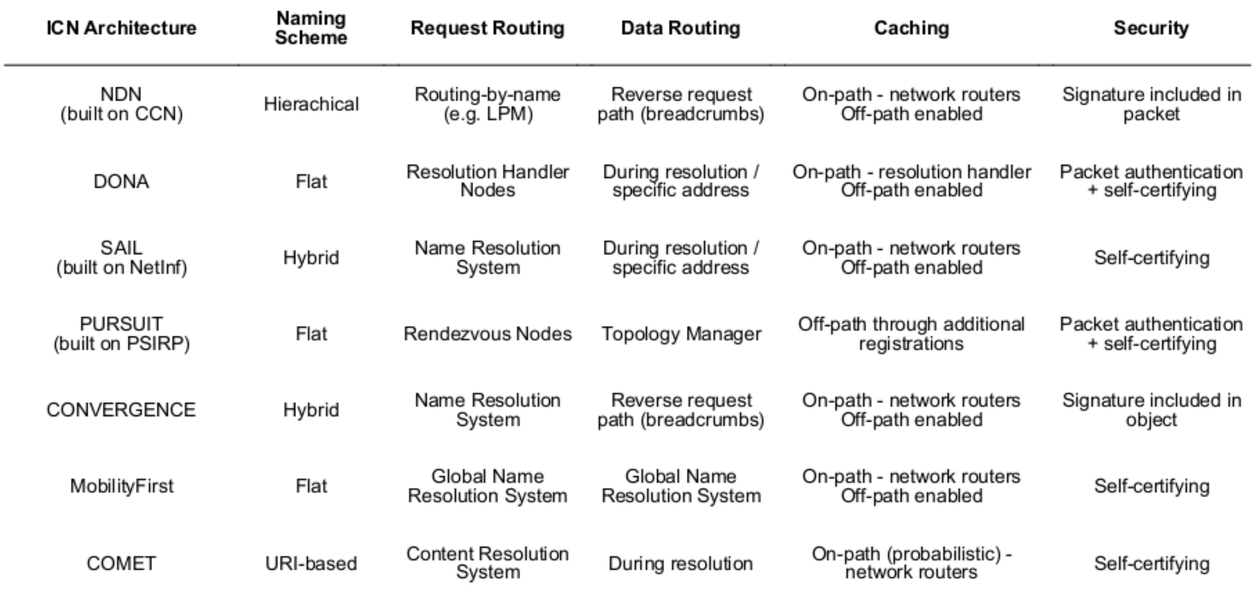
This serves to illustrate the role of *cache availability*, which refers to the knowledge of nearby caches each router has. Improved cache availability can be achieved by the caching mechanism, or more suitably, by the routing policy. In chapter 5, we demonstrate that the improvements available from identifying the nearest replica of content, coupled with non-complex caching mechanisms, help to justify the development of ICN.

2.4 Complete Architectures

Now that we have a basic understanding of the individual components of ICN, we must try to understand them in the context of a complete solution. In this way, the relationship between each element is more clear.

In 1999, the TRIAD Project [6] made the first direct advocacy of extending the Internet with content routing capabilities. Despite the emergence of the subject at the end of the 20th Century, it was not until 2007 and the release of the Data Oriented Network Architecture (DONA) [37], that a complete ICN architecture was proposed. Since then, there have been many potential ICN architectures put forward with a clear lack of consensus on the design [31].

We include a table highlighting the different decisions made for the key features of some of the most popular architectural proposals in Figure 2.6. The table serves to illustrate both the lack of consensus in the field and the difficulty in the aggregation of the individual ICN functions. For example, a decision between a flat or hierarchical naming structure will greatly influence routing and security decisions.



ICN Architecture	Naming Scheme	Request Routing	Data Routing	Caching	Security
NDN (built on CCN)	Hierachical	Routing-by-name (e.g. LPM)	Reverse request path (breadcrumbs)	On-path - network routers Off-path enabled	Signature included in packet
DONA	Flat	Resolution Handler Nodes	During resolution / specific address	On-path - resolution handler Off-path enabled	Packet authentication + self-certifying
SAIL (built on NetInf)	Hybrid	Name Resolution System	During resolution / specific address	On-path - network routers Off-path enabled	Self-certifying
PURSUIT (built on PSIRP)	Flat	Rendezvous Nodes	Topology Manager	Off-path through additional registrations	Packet authentication + self-certifying
CONVERGENCE	Hybrid	Name Resolution System	Reverse request path (breadcrumbs)	On-path - network routers Off-path enabled	Signature included in object
MobilityFirst	Flat	Global Name Resolution System	Global Name Resolution System	On-path - network routers Off-path enabled	Self-certifying
COMET	URI-based	Content Resolution System	During resolution	On-path (probabilistic) - network routers	Self-certifying

Figure 2.6: Implementation of key functionalities in selected proposed ICN architectures

In order to contextualise future discussions, understand the end-to-end considerations, and offer a benchmark on which to research alternatives, this paper now outlines the detail behind one of the more promising proposals. We elect to look at a single architecture in-depth as to look at each would require a paper on its own right. For interested readers, the collective work in [11, 35, 44, 60, 64] can provide more information.

As the only ICN architecture with a complete software ecosystem and the majority of the supporting research, Named Data Networking (NDN) is the forerunner in ICN proposals. As an example of it's success, in 2013, it received substantial funding from the US-based National Science Foundation. It is a clean-slate design, in that it has no dependency on IP. However, it has been designed to be compatible as an overlay to IP for integration purposes. Section 2.4.1 summarises the NDN project reports in [34, 65].

2.4.1 Named Data Networking (NDN)

The minimal functionality of IP is its most important characteristic as it facilitates global interconnectivity while applying little to no constraints on upper or lower layers. It is this flexibility that has allowed the innovation leading to the behemoth that the Internet is today. As you can see from Figure 2.7, NDN is designed to mirror that same flexibility. The architectural differences appear in the *Strategy* and *Security* layers. The former is to allow NDN to take advantage of its simpler relationship with layer 2 transmission by dynamically optimizing over multiple connectivities (e.g. Ethernet, 3G, 802.11). The latter hints at the fact that NDN secures itself through it's naming choices thereby avoiding TCP-like secure connections. In fact, NDN architecture does not require a separate transport layer: data integrity and reliability can be handled by the application layer where appropriate decisions can be made with more information. For example, it is the Consumer application's responsibility to handle retransmission if requests are unsatisfied.

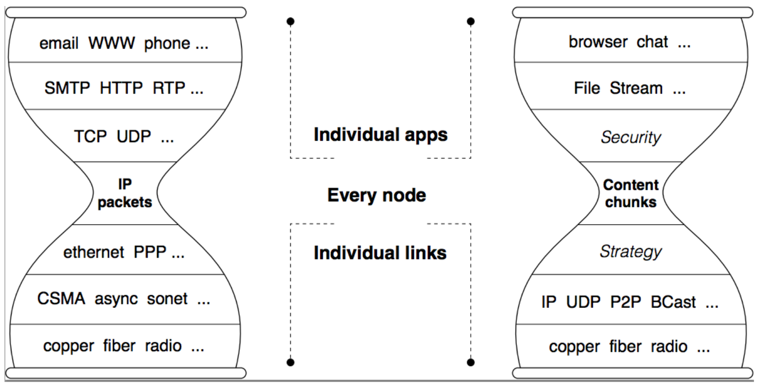


Figure 2.7: NDN closely relates to IP architecture

In Named Data Networking, the granularity of the NDO is at the packet level (known as "content chunks"). Each packet is given an unique name as opposed to each object.

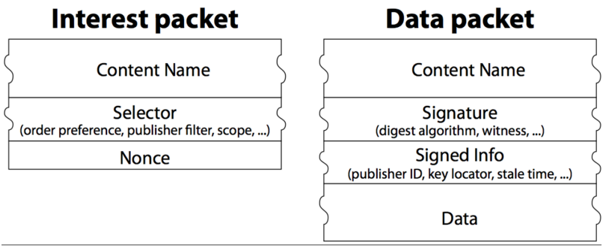

Interest packet
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

Figure 2.8: NDN packet types

As Figure 2.8 illustrates, there are two distinct NDN packet types: *Interest* packets, which correspond to Consumer requests; and *Data* packets, which are responses to the matching requests with the sought-after content. Please find below brief descriptions of the individual components that comprise the two packet types.

- **Content Name:** same for corresponding *Interest* and *Data* packets, see Naming subsection below.
- **Selectors (optional):**
  - min-/maxSuffixComponents: indicator for length of Name.
  - PublisherPublicKeyLocator: way for the *Interest* to select answers from a particular publisher.
  - Exclude: filter possible prefix matches.
  - ChildSelector: express preference for multiple data matches in Content Store.
  - MustBeFresh: if selected, router should not return *Data* packet whose FreshnessPeriod (see below) has expired.

  - InterestLifetime: indicates the approximate time remaining before the *Interest* packet times out (default is 4 seconds).

- **Nonce:** the Nonce carries a randomly-generated 4-octet long byte-string. The combination of Name and Nonce should uniquely identify an *Interest* packet. This is used to detect looping requests.
- **Signature:** proves the validity of the *Data* packet, and sometimes its source.
- **Signed (Meta) Info:**
  - ContentType: packet may be standard data payload or may contain public key etc.
  - FreshnessPeriod: indicates how long a node should wait after the arrival of this data before marking it as stale - routers associate each piece of data with a staleness bit.
  - FinalBlockId: indicates the identifier of the final block in a sequence of fragments.
- **Data:** the packet content.

Content Router Design and Procedures

To manage both the data and control planes (see Terminology in section 2.1), NDN envisages Content Routers to be comprised of 3 distinct components:

1. **Forwarding Information Base (FIB)** is a table that associates prefixes of content names to one or multiple neighbouring routers. This table decides the next router to be sent an unsatisfied request.
2. **Content Store (CS)** is similar to a buffer in an IP router but it also caches Data packets after they have been forwarded, according to an algorithm, in order to serve future requests.
3. **Pending Interest Table (PIT)** keeps track of the content items that have recently been requested and not yet served. Requests for the same content are then aggregated within a single PIT entry and thus not forwarded on. This table maintains the reverse path(s) for data to travel on back to Consumers.

The basic operations of a NDN Content Router are similar to an IP router with added caching. An essential feature is that all requests are consumed by their data responses on a one-to-one basis, or discarded based on their life-time, thereby avoiding floating packets.

Please see below pseudo-code outlines for the procedure upon *Interest* or *Data* packet arrival, and a flow diagram of the *Interest* processing in figure 2.9.

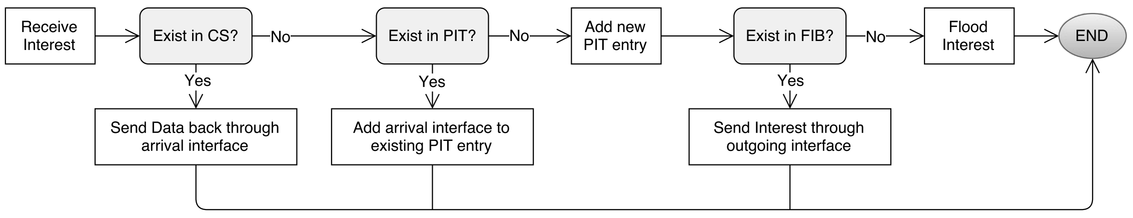

Receive Interest
Exist in CS?
No
Exist in PIT?
No
Add new PIT entry
Exist in FIB?
No
Flood Interest
END
Yes
Yes
Yes
Send Data back through arrival interface
Add arrival interface to existing PIT entry
Send Interest through outgoing interface

Figure 2.9: Processing an Interest packet in a NDN content router

```
Receive Interest packet with Content Name (N) on interface (I);
if N exactly matches Content Name of object chunk in Content Store then
    send the requested data back on interface I;
    depending on replacement algorithm being used (e.g. random, LFU, LRU), update index table
    to reflect changed state (e.g. used at this time);
else if N exactly matches Content Name of entry in PIT then
    update PIT entry to include request from interface I;
    // do not forward Interest Packet on (aggregating requests);
else
    create new PIT entry;
    check FIB table for forwarding information using longest prefix match;
    if there is a prefix match then
        forward Interest packet to one or multiple interfaces determined via FIB;
    else
        flood Interest packet to neighbouring nodes;
    end
end
```

```
Receive Data packet with Content Name (N);
if N exactly matches Content Name of entry in PIT then
    forward Data packet to all requesting interfaces listed in PIT;
    remove request from PIT;
    store Data in CS (based on caching decision and replacement policies);
else
    discard Data packet;
    // either the request was already satisfied or an unsolicited Data packet was received;
end
```

Naming

NDN design assumes hierarchically structured names. For example, segment 2 of version 12 of a video produced by Imperial College's Department of Computing may have the name:

/imperial/doc/videos/IcnForBeginners.mpg/12/2

The '/' merely indicates the boundary between names for binary encoding to assist in human readability. As previously highlighted, NDN uses NDO granularity at a packet level (object chunks) and as such includes segmentation in the naming scheme. In addition, a version or time-stamp is added to the data name to ensure the most up-to-date (or the requested) information is received. Please refer back to figure 2.2 for an illustration of the hierarchy of multiple components.

Note that there is a deterministic algorithm to match end-user requests on partial names to Producer's published content such that a request for */imperial/doc/videos/IcnForBeginners.mpg* would suffice to receive the complete latest version of the video.

Name Resolution and Data Routing

In line with the ICN modus operandi, NDN routes and forwards *Interest* and *Data* packets based on names. This eliminates IP issues such as scalable address management and NAT traversal, since hosts do not need to expose their address in order to offer content.

However, in terms of functionality at the *control plane* level, routing is not dissimilar to IP. A Content Router announces the name prefixes, instead of IP prefixes, that the router is willing to serve. The announcement propagates through the network via a network protocol and every router builds its FIB based on these announcements. For initial deployment, existing intra-domain protocols (e.g. OSPF) and inter-domain protocols (e.g. BGP) can be adapted to route on name prefixes. In the long term, new routing protocols will be required to cater for the potential size of router tables given the number of data names. For example, there is some research into a protocol that aggregates at a top-level within each ISP, reducing the FIB size to be proportional to the number of ISPs rather than users.

Routers treat names as a sequence of opaque components. They do component-wise longest prefix match of the *Content Name* from a packet, against the routing table in the FIB. Using our original video example, the name would match with */imperial/doc* but would only be routed on the interface(s) listed under */imperial/doc/videos* if it were also available. Another area of extensive research is ensuring fast name lookups since these tables could be too large for current fast-lookup technology: many of the most likely eventual results will rely on improved hardware [43].

Finally, it is important to highlight that NDN inherently supports multipath routing. The combination of the unique *Content Name* and *Nonce* element of the *Interest* packet, ensures that duplicate requests are discarded and new requests are aggregated. Therefore, NDN does not require the restrictive Spanning Tree Protocol, necessary in IP, since looping is not possible. Aside from vastly improved security against prefix hijacking, this supports an area of research for identification of best-path routing through the measurement of performance over multiple interfaces.

Caching

NDN does not make specific requirements for cache management and replacement policies since they will be diverse across the network and subject to ISP policies. This paper seeks to identify appropriate caching mechanisms to pursue including our novel caching mechanism, Filter First Caching.

Mobility

Unlike IP, NDN packets cannot loop thanks to the aggregation and one-to-one consumption of *Interest* packets and as such are not restricted to forwarding on spanning trees. Furthermore, there is no layer 2 to 3 binding as there is with an IP address to a MAC address. Given this, optimisation can be achieved at the *Strategy* "layer", identified in figure 2.7, to avoid restrictions on the conventional layers.

Security

In NDN, all content is digitally signed and private content is encrypted. If we consider the make-up of the Data packets in Figure 2.8, the unique human-readable *Content Name* identifies the content and what question it answers, proving content pertinence. While the *Signed Info* states whom is providing the content, proving content provenance. Therefore, the *Signature* validates that the content is complete and authenticates the binding of the name, its origin and the content.

As mentioned before, we will only touch on security related issues when necessary since each element would require thorough discussion in itself.

Chapter 3

Caching Mechanisms

In this chapter, we summarise the existing caching mechanisms for ICN and select the most appropriate for our analysis. The given performance metrics and categorisations are the most prevalent in the literature as suggested by the authors in [66]. Further detail can be found in their extensive survey.

3.1 Categorisation

We classify the existing caching mechanisms into broad categories in order to facilitate cross-comparison and aid in understanding. The categorisations are those used in much of the literature including in [66] for continuity of reference.

On-Path Caching - Off-Path Caching

In on-path caching, data packets are only cached along the path from the content source to the Consumer who has made the request. This may be the path that the corresponding request packet travels. However, that will depend on the name resolution and data routing policies, further illustrating the coupled nature of these two elements for any complete ICN architecture.

Off-path caching facilitates the caching of content on routers other than those on the data packet routing path. This attribute generally requires a centralised management device, such as a topology manager.

Figure 3.1 illustrates how in on-path caching, if a content router CR3 is not on the path between Consumer and Producer then it will not cache any content. To ensure that it assists with the total in-network caching capacity, a resolution handler could send data to an off-path cache and subsequently route requests to that cache should that specific content be requested. Clearly, the overheads of the resolution handler in terms of information storage and network transmissions could be substantial and even outweigh possible benefits.

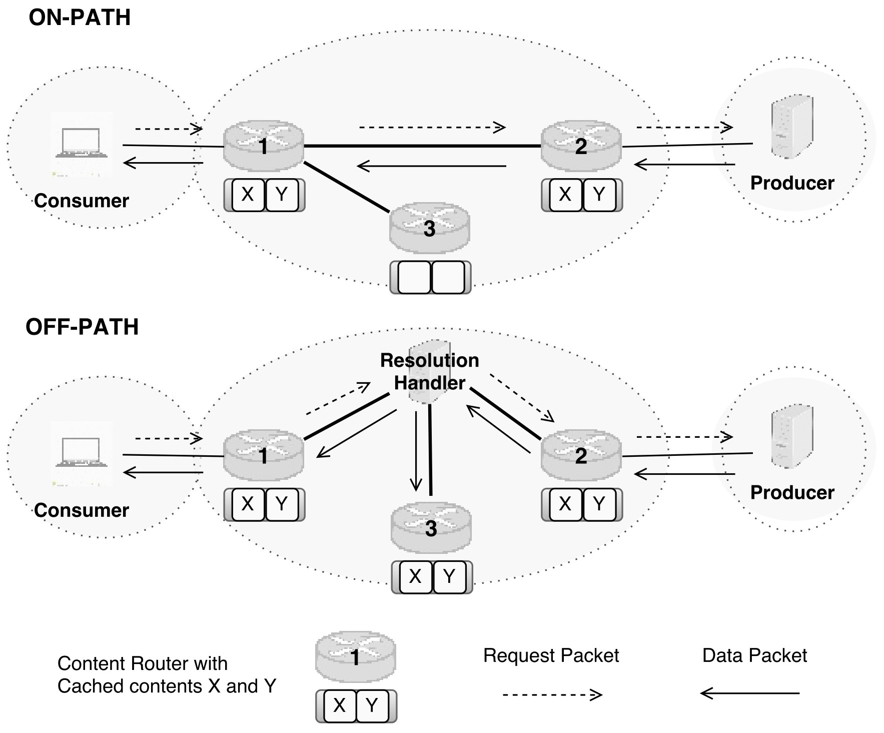


Figure 3.1: Example On-Path Caching vs Off-Path Caching

Cooperative Caching - Non-Cooperative Caching

This category focuses on the existence of any teamwork between the routers. Content routers that make caching decisions independently, and do not advertise their state to other routers are said to deploy non-cooperative caching.

Furthermore, cooperative caching can be implicit or explicit. With implicit cooperative caching, routers are able to coordinate their content store state without an additional advertisement mechanism. This can be achieved through additional operations such as leaving a caching trail at each router on a path for future requests.

On the other hand, explicit cooperative caching is achieved through content routers advertising their cache state to other content routers upon any changes. To do this, they must send additional information packets around the network, thereby reducing potential gains from traffic reduction.

Homogeneous Caching - Heterogeneous Caching

With homogeneous caching, all content routers along the data routing path have the same cache provision and cache all the Data packets that pass through them.

In heterogeneous caching, content routers along a given path may, or may not, cache the passing Data packet, based on various criteria. Alternatively, the content routers may exhibit heterogeneous caching provision whereby strategically placed routers may have a greater cache size than more peripheral routers.

Means of Optimisation

There are two primary methods by which an in-network caching mechanism can optimise performance:

1. **Reducing Caching Redundancy (RCR):** Caching unpopular content that will not be looked up often consumes storage space with little benefit. What's more, caching the same content throughout the network reduces total network caching capacity to that of a single router's capacity.

   For example, we can study the states of a simple network in figure 3.2. We assume that A is the most popular content and H is the least popular content with all other content appropriately distributed in-between, relative to their name in the alphabet. It is clear that the High Cache Redundancy scenario will be unlikely to satisfy the next request unless it is for the less popular contents E or F. Whereas, in the Low Cache Redundancy scenario, any request for content, from A to H, could be satisfied in-network. What's more, the most popular requests for content (A and B) will be returned after a single hop, drastically reducing response time and network traffic.

   Caching mechanisms that address RCR are seeking to ensure that there is both a diversity of cached content in the network and that the most popular content finds it's way closer to the relevant Consumers.

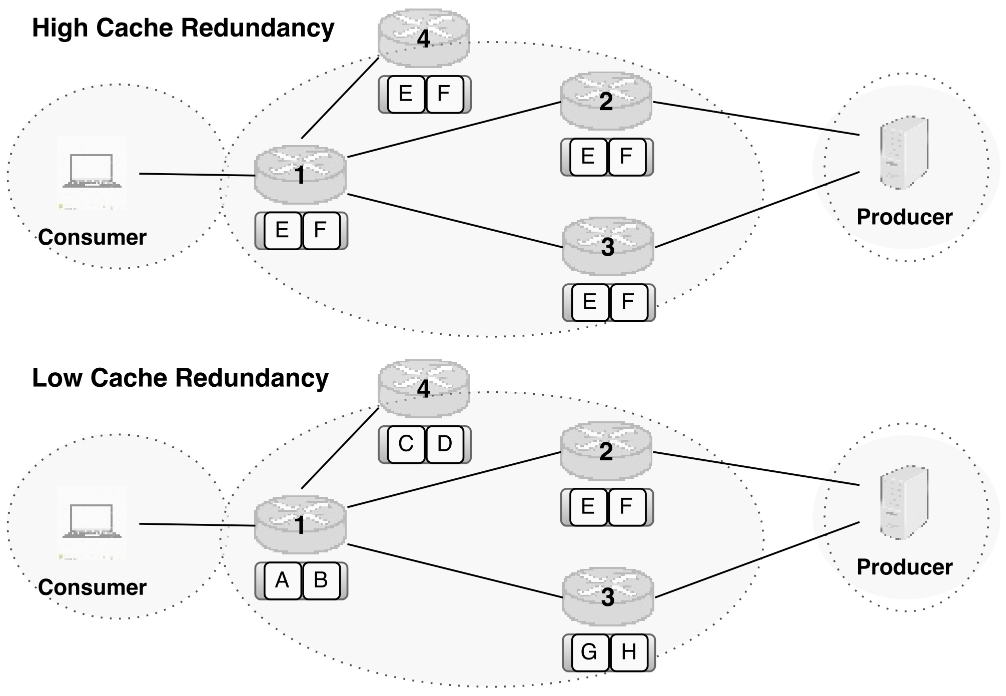


Figure 3.2: Example of high and low caching redundancy

2. **Improving Cache Availability (ICA):** Some mechanisms seek to be able to identify replicas of content that are not on the path to the Producer Server.

   Using the Low Cache Redundancy example of figure 3.2, should the Consumer make a request for content C or D then with no knowledge of the cache availability at CR4, the request would most likely be routed back to the Producer (on-path). However, if CR1 is aware of CR4's cached contents, then the request can be routed to CR4 (off-path), thus improving response time, reducing network traffic and avoiding unnecessary strain on the Producer server.

3.2 Proposed Solutions

Table 3.1 categorises a non-exhaustive list of existing caching mechanism proposals. The table is sorted by whether the mechanism supports on-path caching vs off-path caching followed by the degree of inter-router cooperation required for implementation.

Note that of the caching concerns listed in section 2.2.3, the majority of these mechanisms are solely focused on the caching decision policy. In other words, whether or not to cache the content. Some offer part-solutions for other issues by removing unwanted content (assisting the replacement policy) or specifically utilising other considerations, such as cache placement or cache capacity. In section 3.3, we weigh-up the potential costs and benefits with respect to caching in general.

Short Name	**On- vs Off-path**	**Cooperation**	**Homo- vs Hetero-geneous**	**RCR**	**ICA**	**Ref.**
LCE	On-path	None	Homogeneous	Yes	No	[34]
Fix(p)	On-path	None	Heterogeneous	Yes	No	[40]
HCS	On-path	None	Hetero Provisioning	Yes	No	[49]
Breadcrumbs	On-path	Implicit	Homogeneous	No	Yes	[47]
LCD	On-path	Implicit	Heterogeneous	Yes	No	[39]
MCD	On-path	Implicit	Heterogeneous	Yes	No	[39]
WAVE	On-path	Implicit	Heterogeneous	Yes	No	[36]
Probcache	On-path	Implicit	Heterogeneous	Yes	No	[45]
CBC	On-path	Implicit	Heterogeneous	Yes	No	[63]
CLS	On-path	Implicit	Heterogeneous	Yes	Yes	[41]
Intra-AS Co	On-path	Explicit	Homogeneous	Yes	Yes	[61]
CATT	On-path	Explicit	Heterogeneous	Yes	Yes	[27]
ICC	Off-path	Implicit	Homogeneous	No	Yes	[46]
CINC	Off-path	Explicit	Heterogeneous	Yes	Yes	[42]
DCM	Off-path	Explicit	Heterogeneous	Yes	Yes	[54]

Table 3.1: Categorised Caching Mechanisms

For the sake of concision, we will only briefly describe each of the most widespread caching mechanisms with more emphasis on those identified as suitable for coupling with name resolution techniques (see section 3.3). Further information on many of the mechanisms is available in [66].

LCE - Leave Copy Everywhere: LCE is the most simple caching mechanism and is the default solution for NDN. In LCE, each content router along the path that the data packet traverses back to the user, caches the content.

Fix(p): Fix(p), also known as Prob(p), is a randomized version of LCE. An intermediate cache on the path from the location of the hit down to the requesting client will cache a local copy with probability p, and does not keep a copy with probability 1 - p. Fix(1) is identical to LCE. Intuitively, the most requested content (more popular) is more likely to have been cached in the network.

HCS - Heterogeneous Cache Sizes : HCS is similar to LCE in that each content router caches all passing-by data packets. It differs by allocating additional caching capacity to the more "important" routers in the network. A router's importance is determined by 6 different metrics that focus on the interconnected nature of the router in terms of neighbouring links, hop counts and delays.

Breadcrumbs: Breadcrumbs caches a data packet at every content router along the downloading path, just like LCE. Its additional functionality comes with storing a one step caching path at each router to assist in future interest routing. In short, it facilitates the potential identification of a nearer in-network replica downstream of a Consumer, rather than only being able to go upstream to a Producer. Breadcrumbs exclusively improves cache availability.

LCD - Leave Copy Down: Under LCD a new copy of the requested object is cached only in the router that resides immediately below the location of the hit on the path to the requesting client. With LCD, it requires multiple requests to bring an object towards a leaf cache, with each request advancing a new copy of the object one hop closer to the Consumer. In this way, only the regularly requested content for an area is cached in the closest content routers to that area. The staggered caching is achieved through a flag bit added to each Data packet's header, which is reset after the first hop.

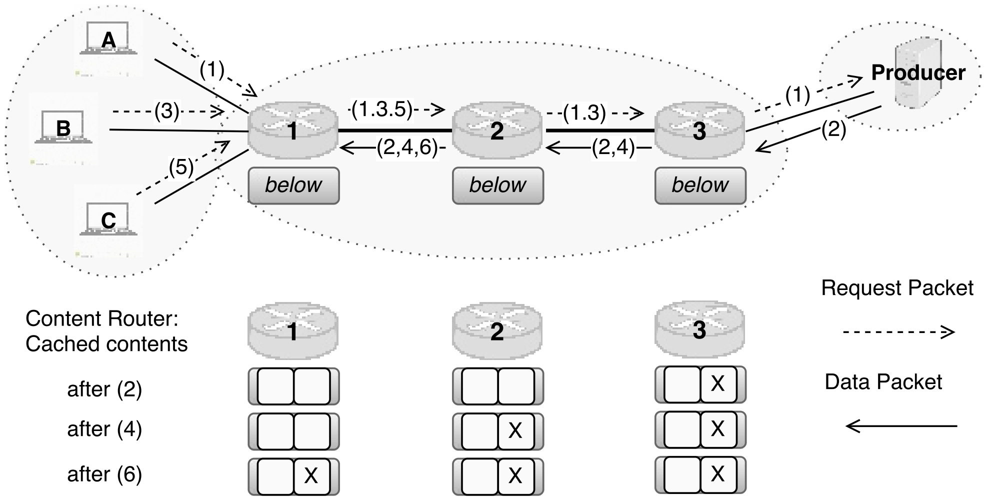


Figure 3.3: Example LCD operation

In figure 3.3, we assume Consumer A requests content X (1), to which the Producer replies with the relevant data packet (2), with the caching flag bit set to 1. The first downstream content router CR3 caches the data packet since the flag bit is set, after which CR3 resets the bit to zero. All subsequent content routers on the path of (2) back to Consumer A do not cache the content since the flag bit is not set. Now, another Consumer B requests the same content X (3). The request is routed towards the source Producer but is satisfied by the in-network cache at CR3. CR3 sends the data packet back towards Consumer B with the flag bit set (4), and as such CR2 caches content X and resets the flag bit. Finally, Consumer C also requests content X (5). The request is satisfied by CR2 (6) and the content is cached in CR1. As you can see, the content only reaches the most important nodes nearest to the

Consumers if it is requested often by those Consumers. This is a large improvement with regards to caching redundancy compared to LCE.

MCD - Move Copy Down: Similar to LCD with the difference that once content is cached a level further down the path, it can be deleted from the router where the hit originated from (excluding the origin server of course). This should reduce the number of replicas for the same object on the path between the Consumer and the Producer. However, one can see a potential conflict of interest with MCD when coupled with improved routing policies. For example, if a request is routed downstream (with regards to the Producer server) from a different Consumer to the in-network replica, then the content will be pulled upstream once again.

WAVE: WAVE works similarly to LCD but views each chunk (packet) of the same content in a united way. The strategy uses a specifically designed Chunk Caching Algorithm to determine the number of chunks of each object to store at a given router. In short, the more requests for the content, the more chunks of the content are stored in the network. What's more, they will be stored closer to the Consumers requesting it.

Probcache: Probcache is a probabilistic caching mechanism similar to Fix(p). However, probability p differs for each content router based on their location along the downloading path.

This is implemented through a Time Since Inception (TSI) field in the request packet header, and a TSI field and a Time Since Birth (TSB) field in the data packet header. When a user sends out an request packet, the TSI value is set to be zero. When a content router receives an request packet, it increments its TSI value. Similarly, when a content source sends out a data packet, it sets its TSB value to zero and sets its TSI value to match that of the corresponding request packet. When a content router receives a data packet, it increments its TSB value, while keeping the TSI value unchanged. The Cache Weight (CW) is calculated at each router as the value *CW = TSB / TSI*. Figure 3.4 illustrates this meta-data management in practice.

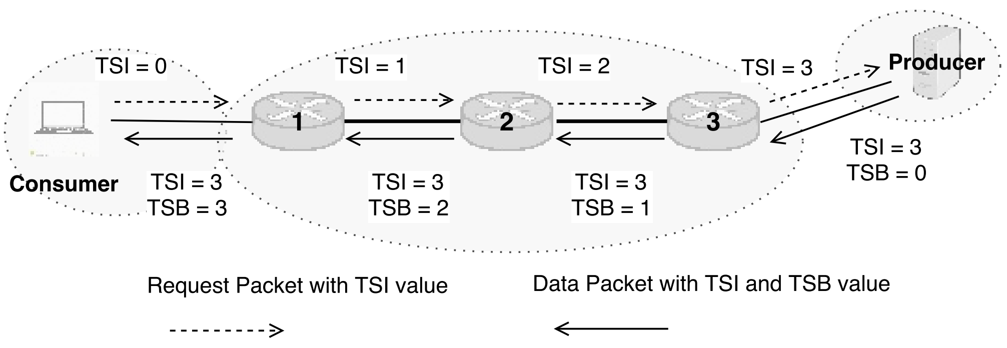


Figure 3.4: Example Probcache meta-data management

In addition, on packet arrival, a router will calculate it's TimesIn factor which estimates the remaining caches available on the download path to the Consumer. Ultimately, a content router then caches an incoming data packet with probability equal to *Prob = CW * TimesIn.* This should result in popular content quickly reaching edge nodes, closer to the Consumers.

CBC - Centrality-Based Caching: CBC caches data at the content routers on the downloading path which have the greatest betweenness centrality (which we denote as $C_B(x)$ for CRX). Betweenness centrality values for a domain are pre-calculated off-line and inserted into the routers. They reflect how often the router lies on the shortest path between two routers. The equation is simply:

$$C_B(x) = \sum_{x \neq y \neq z \in X} \frac{n(y,z,x)}{n(y,z)} \tag{3.1}$$

where

$$n(y,z,x) = \text{no. of shortest paths from CRY to CRZ through CRX} \tag{3.2}$$

$$n(y,z) = \text{no. of shortest paths from CRY to CRZ} \tag{3.3}$$

To illustrate an example of betweenness centrality in practice, we can consider figure 3.5. It is clear the content router with the largest betweenness centrality along the path CR1 to CR3 is CR2. Therefore, for requests from our Consumer at CR1 for data from our Producer at CR3, the returning data packets will only be cached at CR2. In this way, should a Consumer (not illustrated) at CR4 request the same content then the in-network copy will be in the most appropriate place for rapid response.

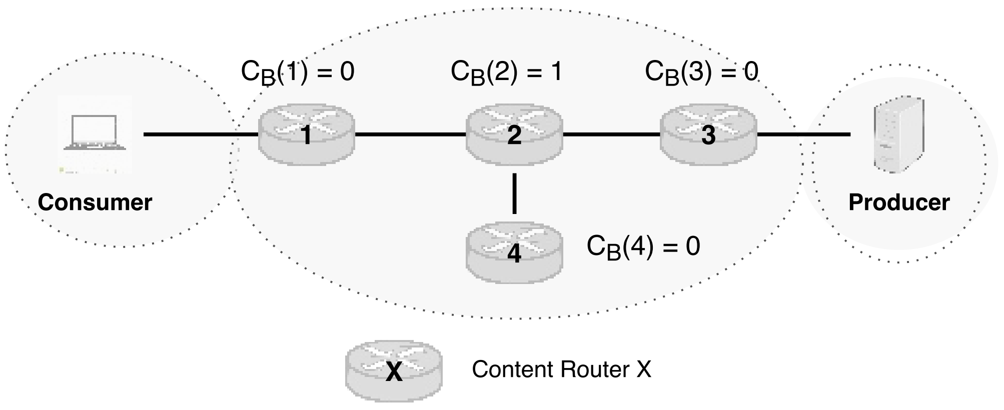


Figure 3.5: Example Betweenness Centrality

The obvious flaw with this mechanism is the over-use of "important" routers, including frequent replacement, while other nearby routers may be left idle. In our example, CR2 would be the only router to cache any content.

What's more, any update to the network structure would require some form of central manager to propagate the corrected betweenness centrality values to all routers. Thus, unlike many of the other mechanisms, this proposal becomes dependant on the design of the topology. Simple networks such as in figure 3.5 simply would not suit CBC.

CLS - Caching Location and Searching scheme: CLS seeks to improve on the concept of MCD that there be at most one copy of a chunk cached on the path between a Producer and Consumer. The key concept being that a cache hit pulls the requested chunk down to the Consumer and an eviction moves a banished chunk back towards the Producer. In addition, CLS integrates a caching trail similar to that of Breadcrumbs. This is all implemented by CLS content routers maintaining a table with upstream and downstream information on all cached content.

Intra-AS Co - Intra Autonomous System Cache Cooperation: An Intra-AS Co content router cooperates with it's one-hop neighbouring nodes by periodically sharing a summary of currently cached items (similar to distance vector routing). With this information, nearby nodes can implicitly cooperate to serve each other's requests. In addition, the domain can utilise a greedy algorithm to reduce caching redundancy: deleting content already stored in neighbouring nodes will improve cache hits by allowing more distinct content to be cached in-network.

CATT - Cache Aware Target Identification: CATT bases a caching mechanism around Potential Based Routing (PBR). In short, with PBR, a content router shares with it's neighbours a combination of the availability of content and the proximity of themselves to that content's source. The combination of these two factors decide the content's quality rating (it's potential). When a request for content arrives at the router, the highest quality rating for that content will be looked up in the PBR table (replacing the standard FIB) and the request will be forwarded to that router.

ICC - Incentive Compatible Caching: Primarily designed for inter-domain cache cooperation in response to perceived flaws in the DONA architecture, ICC is a replacement for possible uses of Border Gateway Protocol implementation in ICN. In short, ICC proposes that central Resolution Handlers (RHs) not only cache network content but advertise the content that they cache to nearby RHs. The optimisation technique is to identify caching availability as opposed to lowering caching redundancy.

CINC - Cooperative In-Network Caching : CINC adds two tables in each content router: a Collaborative Router Table (CRT), which records every content router in the network domain; and a Collaborative Content Store (CCS), which records the names of the content cached at each of the content routers stored in the CRT. Furthermore, a content router only caches based on a comparison with the sequence number of a data packet and the id of the content router within the network. Using the CRT, a data packet can be forwarded to the correct router in the network for it to be cached (sequence number vs router id). Using the CCS, a request packet can be forwarded to the correct router to satisfy the request. In this way, CINC addresses both caching redundancy and cache availability successfully. However, the overheads in terms of coordination packets and caching capacity limits are substantial.

DCM - Distributed Cache Management : The central concept of DCM is a Cache Manager (CM) that manages all the caches within it's domain (hierarchical mechanism). In addition, CMs communicate with nearby domain CMs to share their cache state in an attempt to maximise cache availability in a given location. All caching decisions are made by the CM with complete knowledge of the content router contents in it's and neighbouring domains. This comes at the cost of efficiency and increased overheads.

Please note that this is not an exhaustive list. Some mechanisms have been omitted due to over similarity with included proposals, while others due to their vast overheads not-befitting the necessary streamlined nature of ICN (see section 3.3).

3.3 Evaluation

In this section, we present the performance and cost metrics for evaluating the benefits of a caching mechanism. Ultimately, we use these metrics to narrow down our caching mechanism selection to those suitable for use in our simulations in chapters 5 and 6.

From a high-level perspective, the potential benefits of in-network caching include reducing latency for users accessing content and diminishing overall network traffic [39]. In order to evaluate the performance of caching mechanisms in reaching these goals, we identify metrics by which to compare the success and shortcomings of each.

- *Cache Hit Ratio*: a cache hit means that a request for a content chunk can be satisfied by a cached copy within the network, in contrast with a cache miss which must be satisfied by the Producer. A higher cache hit ratio means reduced load on the Producer's server and suggests improved request response time.
- *Cache Hit Distance*: the average distance from the user to the content router that services the request in terms of number of routers passed through, with consideration of all user requests made.
- *Response Latency*: in simple terms, this is the delay in seconds in delivering the content back to the Consumer after a request has been made. This relates to flow and congestion control, potentially requiring a congestion-aware caching mechanism [16], which we are not looking to analyse in this paper. Thus, it will not be a primary evaluation metric for our purposes.

In addition to perceived benefits of each caching mechanism, it is important to evaluate the potential costs of each strategy in terms of network load, packet header size and content router space or complexity. While the impact is difficult to measure, they should be considered when comparing efficacy of the caching mechanisms.

- *Packet Cost*: some caching mechanisms may require additional information to be sent with each content chunk increasing the packet header size. Additional header fields range from single flag bits to recording router history.
- *Router Cost*: for certain policies, the content router complexity is increased with added tables and / or packet management. For example, a content router may be required to maintain additional routing tables thereby increasing its potential lookup time and necessary storage capacity.
- *Network Cost*: some caching policies require additional packets to be sent to regulate the decision policy thereby increasing network traffic. For example, a content router may be required to send a message to a central hub, or its neighbouring nodes, advertising the contents of it's cache.

In table 3.2, we summarise the additional costs of each of our described caching mechanisms. Using this we then identify those solutions which are inappropriate for consideration in this paper below.

Short Name	**Packet Cost**	**Router Cost**	**Network Cost**
LCE, Fix(p), HCS	n/a	n/a	n/a
LCD, MCD, WAVE	flag bit (Data)	n/a	n/a
Probcache	TSI (Interest) TSI & TSB (Data)	n/a	n/a
CBC	max value (Interest)	n/a	n/a
Breadcrumbs	n/a	caching trail	n/a
CLS	flag bit (Data)	caching trail	n/a
Intra-AS Co	n/a	neighbouring caches	advertise cache
CATT	n/a	PBR replaces FIB	advertise cache
ICC	n/a	data from RH	advertise cache
CINC	flag bit (Interest)	CRT & CCS	advertise cache
DCM	n/a	other CMs' state	advertise cache

Table 3.2: Costs of Caching Mechanisms

Network Cost

Most off-path caching mechanisms require a centralised manager or extensive communication between routers in order to function. This adds unnecessary strain to the network in terms of traffic and complexity which often outweigh the potential benefits with regards to caching and routing. Indeed, the existence of a centralised device often places restrictions on the efficacy of the architecture as a whole [45]. For this reason, it is often reserved for inter-domain policies, which we are not specifically reviewing in this paper.

Furthermore, there is evidence to show that the benefits of mechanisms reliant on explicit cooperation do not outweigh the additional messages required for coordination [45].

These concerns specifically apply to: Intra-AS Co, CATT, ICC, CINC, DCM.

High Router Cost

One of the most pressing barriers to entry for ICN proposals is the limitations of existing hardware [43]. Each content router must be capable of line speed lookup over a cache with sufficient capacity to make the in-network caching worthwhile (cache size relative to total Internet content with respect to popularity distribution). As such, those mechanisms that require the use of *large* amounts of storage capacity within each content router are not realistic for practical implementation.

These concerns specifically apply to: Intra-AS Co, ICC, CINC, DCM.

Focus on Improving Cache Availability

Since this paper stipulates that ICN benefits can only be understood when optimising over caching and routing simultaneously. Those mechanisms that only improve *cache availability*, usually at additional router cost, are not necessary for consideration. For example, Breadcrumbs and CLS enhance LCE and LCD respectively, with improved cache availability. This enhancement is not necessary in our optimal routing scenario, and thus, these mechanisms are superfluous to our analysis.

These concerns specifically apply to: ICC, Breadcrumbs, CLS.

Complexity to Accurately Simulate

While heterogeneous provisioning will most likely be included in many complex networks (decided by the ISP), it is not suitable for this paper's concerns. Indeed, simulations of provisioning schemes are specific to each topology: network layouts with a greater proportion of linking will require greater cache capacities. As such, accurately simulating provisioning schemes, and specifically comparing their results with other mechanisms, is only possible in certain scenarios.

These concerns specifically apply to: HCS, CBC

Similarity to other Mechanisms

Finally, there are those mechanisms that we have mentioned that are not significantly diverse from more simple solutions and do not necessarily add value. For example, MCD has been shown to add complexity to the LCD mechanism without improving performance [39].

Furthermore, there are schemes that require specific implementation choices in order to be relevant. For example, WAVE relies on considering object chunks independently. However, as you will see below we intend to use single chunk objects to simplify the simulations and focus on the relationship with routing.

These concerns specifically apply to: MCD, WAVE.

Final Selection

Given the concerns with certain mechanisms we have outlined, we make our final selection as suitable for the simulations in this paper. On the whole, they are simple mechanisms with a focus on reducing cache redundancy, making them more comparable with our similarly focused novel caching mechanisms in chapter 6. They also reflect the most represented caching policies in the literature.

Selected Caching Mechanisms
no caching
LCE
Fix(p)
ProbCache
LCD
FFC, FFC(p), TFC, WFC(p) (see chapter 6)

Table 3.3: Selected Caching Mechanisms for use in simulations

Chapter 4

Implementation

Name	Values	Brief Description
Network Topology	see section 4.2	Includes number and location of Producers and Consumers (and degree of replication)
Routing Strategy	{SPR, NRR}	Shortest Path / Nearest Replica Routing
Caching Mechanism	see table 3.3	see sections 3.2 and 6.1
Cache Replacement	LRU	Least Recently Used algorithm
NDO Granularity (G)	1	Each object from the catalogue consists of a single chunk (packet)
Catalogue Size (N)	10^5	Represents the cardinality of the catalogue, expressed in number of contents
Cache Size (C)	10^2	Number of packets that can be cached in a router (homogeneous cache provision)
Name Cache Size (NC)	10^2	Number of names that can be cached in a router's Name Cache, see section 6.1
Zipf Exponent (α)	[0.5, 1]	Relative probability of a request for the ith most popular page is proportional to $1/i^\alpha$
Control Window (cW)	60s	Stabilisation metric, see section 4.6
Sampling Time (sT)	0.1s	Stabilisation metric, see section 4.6
Variance Threshold (V)	0.05	Stabilisation metric, see section 4.6
Partial Nodes (pN)	{0.5, 1}	Stabilisation metric, see section 4.6
Request Rate (λ)	20Hz	User requests per second for each client
Sim Time (T)	600s	Post stabilisation simulation time

Table 4.1: Scenario Parameters

In this Chapter, we outline the necessary parameters for a network simulation, which will be required to support contributions made in the following Chapters. In particular, we seek to justify any assumptions we make and explain the technical work required to achieve the results. We summarise the key parameters for the sake of readability in table 4.1.

4.1 Simulator

The authors in [57] identify four simulators as the most widespread and comprehensive of the numerous software tools available for ICN analysis. Crucially, they emphasise that all four provide similar results for similar configurations implying that scenarios are well calibrated across each simulator. We discuss their relative merits to justify the final choice.

CCNPL Sim [1] is exclusively designed to assess flow and congestion control in an ICN environment; it is not suitable for a routing / cache mechanism analysis.

ndnSIM [10] is built on ns-3 [3] with a focus on completeness and offers an accurate representation of Named Data Networking. As such, it offers little flexibility on name resolution and data routing which is simulated in line with NDN specifications.

Icarus [52] is a Python written simulator that imports from networking libraries such as NetworkX and Fast Network Simulation Setup (FNSS). It's primary focus is on the issues of scalability and energy efficiency in ICN networks but offers a wide degree of flexibility with no ties to any specific ICN architecture. It's flaws lie in the simplified nature of it's network structures, including ignoring link capacities and router buffers.

ccnSIM [48, 58] is a chunk-level simulator written in C++ built under the Omnet++ framework. While it has some gaps in implementation, including a lack of multipath routing, it's open structure facilitates the modification of the code itself, in order to suit a user's needs. In this way, it's flexibility makes it the clear choice for wide-ranging simulations.

In reviewing each of the most commonly used simulators, the inability to simultaneously experiment over varying caching mechanisms and routing policies was further highlighted. Only ccnSim and Icarus contemplate the functionality at all and even then with limited usability in the unmodified code.

Having experimented with all of the above, aside from CCNPL Sim, it became clear that **ccnSim** was the best suited for the purposes of this paper. It offers the requisite framework to assist large scale simulations, alongside the flexibility to adjust name-routing policies and add novel caching mechanisms.

Furthermore, ccnSim implements the CCN architecture which is the predecessor to the NDN architecture outlined in section 2.4.1. As such, the reader should be familiar with terminology such as interest packet, Content Store (CS), Pending Interest Table (PIT) and Forwarding Information Base (FIB).

4.2 Network Topology

Intuitively, one would suspect that the design of the network topology will dictate the potential results of any simulation. However, there is evidence to suggest that in fact, the impact of the topology is limited [48, 50]. Indeed, provided that heterogeneous latencies are taken into account, variations in relative performance of different routing and caching strategies across various topologies, are minimal. Having accepted this finding, we can consider topologies to specifically illustrate our intended purpose, so long as we implement uniform link delays (for our synthetic networks we use 1ms homogeneously). We do include an analysis over multiple real-world topologies both for cross-comparison purposes and to validate this theorem.

In table 4.2, we report the main characteristics of the considered topologies, including the diameter which represents the shortest distance between the two most distant nodes in the network (in terms of hop count).

Topologies	**Nodes**	**Consumers**	**Producers**	**Diameter**
Ternary Tree	40	27	1	4
Bus	10	10	1	10
Abilene (a)	11	11	1	6
Tiger (b)	22	22	16	5
Geant (c)	22	22	16	4
Dtelecom (d)	68	32	16	3
Level3 (e)	46	32	16	4

Table 4.2: Characteristics of selected topologies for use in simulations

We expand on the individual layout of each topology in the subsections below together with an illustration of the real world topologies corresponding to their associated letter in figure 4.4.

Note that we simulate with no overlapping content in Producers. In other words, the object catalogue (cardinality of 10^5, see section 4.4) is randomly distributed across the number of Producers with no Producer sourcing the same content as another. For example, in a simulation with 16 Producers, each Producer will hold roughly $\frac{10^5}{16} = 6250$ unique pieces of content.

Ternary Tree with Probabilistic Linking

Our primary topology is a 4-level ternary tree with 40 total nodes as represented in figure 4.1. The 27 edge content routers all act as Consumers (one can think of it as having a stream of Consumers making requests within that router's sub-domain). The sole Producer is at the top of the tree (CR1).

Of course, real networks seek to avoid single link connections since it equates to a single point of failure, after which the node cannot be reached. As such, to recognise the realistic need for fault tolerance, we add probabilistic links between uncle and nephew nodes.

This is broadly in line with the work in [51]. However, we note that the analysis in their paper is actually flawed due to the design of their binary tree. Figure 4.2 helps to illustrate that by creating links between all nephew nodes to a single uncle node, one of the parent nodes may be inactive. This characteristic is brought about by the nature of ccnSim which prevents multipath routing and

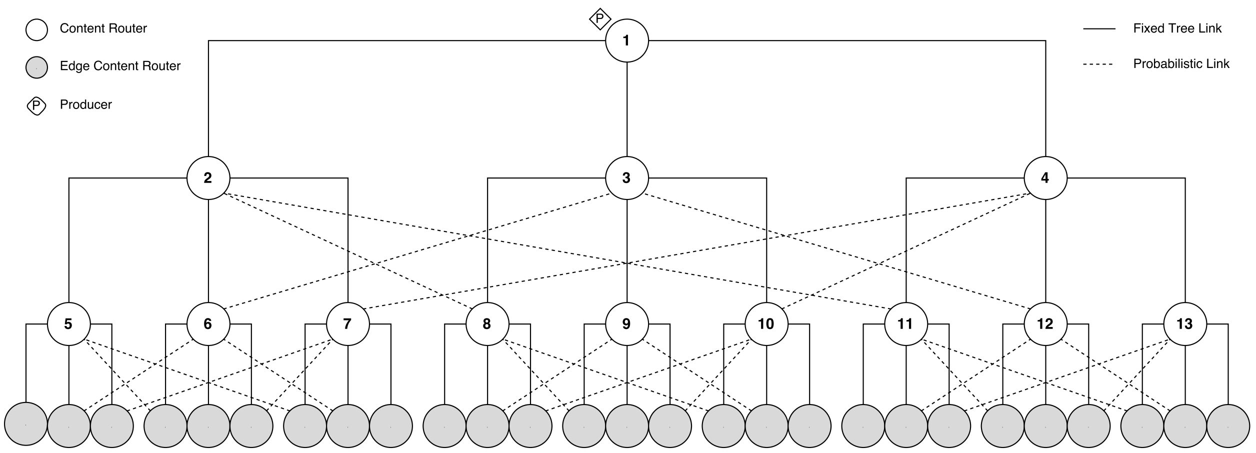


Figure 4.1: 4-level ternary tree topology with probabilistic links present with probability p

therefore, packets always elect to go through the same router if possible. Once the node is inactive like this, it will receive no packets, caching no content, misrepresenting the value of ICN.

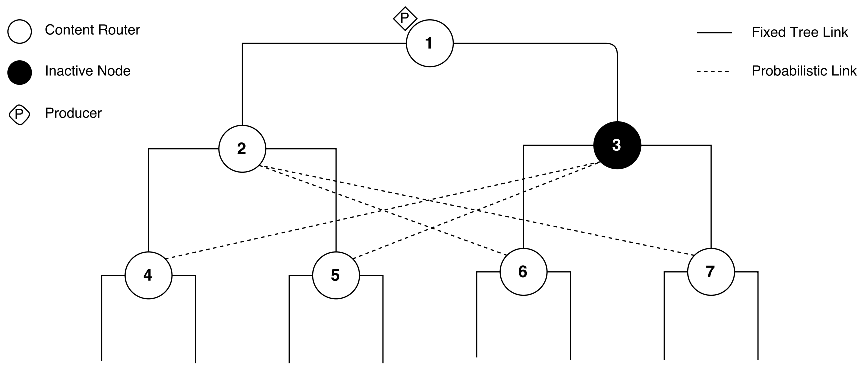


Figure 4.2: Flaw with binary tree topology in [51] - due to the ability to route all requests through CR2 - CR3 is inactive

To solve this problem, we designed our more robust ternary tree whereby nephews and uncles can only create a probabilistic link if they are in the same branch of their triplet. For example, CR6 is already bound to it's parent, CR2, and can only probabilistically link to it's uncle CR3 (and not to CR4). In this way, each node must receive some packets and therefore will not go inactive, and will cache content which can be located by a suitable routing policy to satisfy requests.

In order to generate a fair representation of the existence of the probabilistic links with probability p. We created 20 different networks for each 0.1 step increment of p in the range [0,1]. As an example, at $p = 0.5$, we create 20 different networks with the core fixed tree links in place and a random selection of $0.5 * 24 = 12$ distinct probabilistic links. We then run the simulation on all networks for that value of p and the average performance metric is calculated and reported. This should serve to reduce any effect of anomalous link set-ups.

Finally, we also check effects of varying the ternary tree's size. While the primary performance analysis is done on our 4 level ternary tree, we make it possible to build ternary trees with up to 8 levels. In the interest of practicality of simulations, we construct our larger tree networks with $p \in \{0, 1\}$. This allows us to make comparisons across tree size at the extremes of probabilistic linking.

Note, for more observation, we make available our C++ program for generating the requisite ternary tree networks in the attached archive. Indeed, to run scenario A or D in this paper, one would have to compile and run the program to add the .ned files to the ccnSim directory.

Bus

For initial evaluation of our novel caching mechanisms in chapter 6, we consider a simple bus topology of 10 nodes. Each of the 10 nodes acts as a Consumer while only a single end node (CR1) acts as a Producer. The simplicity of the topology will allow a further look at effects of variations in the Zipf-alpha exponent (see section 4.5) in the context of Filter First Caching.

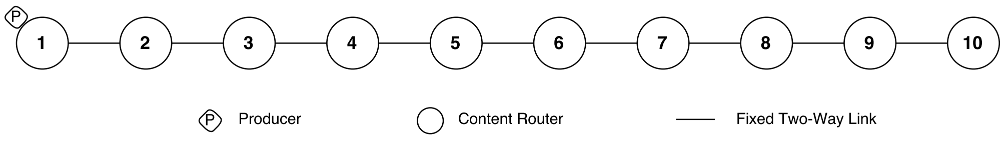


Figure 4.3: 10 node bus topology with single Producer repository

Real World Topologies

To assist in cross comparison of our novel caching mechanisms, we also consider real network topologies that are publicly available through Rocketfuel [5, 55]. For the sake of feasibility, we consider only the backbone of Autonomous Systems. This should have no consequence since the stream of Consumer requests from clients in the network can simulate multiple Consumer applications at each edge node. We elect to use the pruned design constructed by ccnSim creators since this better facilitates accurate replication of this work.

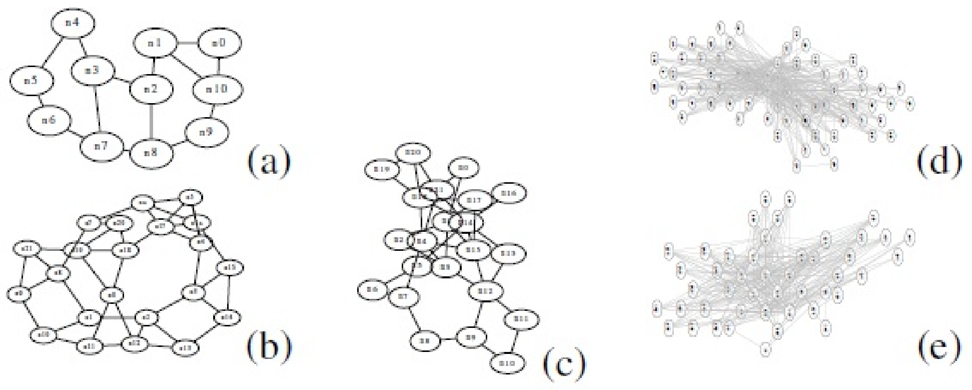


Figure 4.4: Representations of 5 pruned real-world topologies [48]

Figure 4.4, taken from [48], gives representations of the structure of each chosen topology. The topologies correspond to the given letters in table 4.2.

Being real-world networks, they have diverse link delays which could suggest a difference in performance. However, the performance metrics we have chosen to use (see section 4.7) do not require a measure of latency. For example, measuring hop count is distinct of link delay provided the routing strategy is seeking to optimise over number of nodes, rather than round trip time.

Where we do have an issue is in the **placement of Consumers and Producers** within our real-world topologies. In order to roughly illustrate the backbone of a distribution network, we would ideally have all nodes acting as Consumers and Producers. However, ccnSim uses the placing of 1 bits in a variable's bit representation to randomly distribute both content to a Producer's repository, and request creation to Consumers. Owing to this, we are restricted to a maximum of 16 Producers and 32 Consumers in a network. While initial work to fix this problem was attempted, the interconnected nature of the functionality would have virtually required the creation of a new simulator in itself.

As a solution, we opt to use the maximum available for each topology. For example, we assign all 22 content routers as Consumers in our Geant topologies and 32 of the 68 routers in our Dtelecom topologies will be Consumers.

Given this random assignment, we appreciate the potential for varying results. For example, if the Consumers are assigned next to the Producers in a corner of the network, there may be skewed results. To avoid the effects of this eventuality, we take the average of 10 arbitrary distributions of Producers and Consumers, with regards to each real-world topology. During the actual simulation, we found there to be some variance between the individual versions but not to a degree where it affected relative performance of the caching mechanisms.

4.3 Caching and Strategy-Layer

Name Resolution and Data Routing

In chapter 5, we seek to demonstrate the potential benefits of ICN in-network caching are worth the continued research. To do so, we use simple caching mechanisms, which we will later test against our own caching solutions, coupled with improved routing policies. The underlying contribution is the concept that ICN benefits can only be understood when optimising over caching and routing simultaneously. Given the objective of demonstrating the potential of ICN, we can simply demonstrate it's lower and upper bound performance scenarios, thereby demonstrating that future research will simply help push actual performance towards the upper bound.

A majority of the literature on ICN caching fails to consider performance gains from improved routing policies. Most of the research is satisfied with using the default policy, **Shortest Path Routing (SPR)**, as the underlying request forwarding strategy [34, 36, 39, 40, 45]. With SPR, the request is simply routed on the shortest path to the nearest Producer server of the matching content. Aside from random routing and random repository routing, this is the lower bound of routing policies available to ICN. In fact, it negates many of the advantages of in-network caching since requests can only be satisfied if the content has been cached on the path between Consumer and Producer.

The upper bound of routing policies is **Nearest Replica Routing (NRR)**. With NRR, requests will be sent to the nearest possible replica of the content requested, which could be an on-path or off-path in-network cached copy, or within the Producer server. For our purposes, proximity is assessed in least number of nodes to pass through (hop count) since that correlates with our primary performance measure (see section 4.7). Note that there has been some promising research getting close to nearest replica routing [22, 51], so it is not an unattainable measurement of the potential benefits of ICN.

Caching Mechanism

We repeat the table of selected caching mechanisms from section 3.3. Please refer back for justification of the selection.

We include no caching as an example of performance in IP without any caching enhancements. To demonstrate the performance of existing Content Delivery Networks (CDN) in IP networks, we use the LCE caching mechanism applied solely to the edge routers (we refer to this as spr-Edge in our performance graphics). In addition, we test the capability of intelligent CDNs (which we refer to as nrr-Edge). With nrr-Edge, we apply our optimal routing policy to the CDN scenario in IP to provide a fair comparison across IP and ICN. It is not infeasible for CDN servers to be able to re-route requests to neighbouring ISPs known to be storing the desired content.

Note that as with SPR as the lower bound of routing, we view LCE as the performance lower bound for ICN caching, due to it's poor cache redundancy solutions and high eviction rates. We seek to identify an upper bound in chapter 6, alongside our novel caching mechanisms.

Selected Caching Mechanisms
no caching
LCE
Fix(p)
ProbCache
LCD
FFC, FFC(p), TFC, WFC(p) (see chapter 6)

Table 4.3: Selected Caching Mechanisms for use in simulations

Cache Replacement Policy

We can imagine a caching policy of being composed of three central elements. The first two of which are the caching decision strategy and the routing technique employed. The final element is the cache replacement policy and substantial research has gone into seeking it's ideal solution. Indeed, research into replacement policies has been going on for some time [13]. However, ICN has two defining characteristics which make the simpler policies more appropriate despite prevailing sentiments:

1. **consecutive requests are correlated** since they often represent subsequent chunks of the same object
2. **replacement must happen at linespeed** to avoid any increase in latency

With these two considerations, the complexity of the policy is very limited. Four simple replacement policies have dominated the majority of the literature [21]:

LRU the least recently used chunk is replaced

LFU the least frequently used chunk is replaced

FIFO the oldest chunk is replaced (First In First Out)

Random a chunk selected uniformly at random is replaced

This list does not include Most Recently Used and Most Frequently Used since they have been shown to be entirely unsuitable for ICN. What's more, additional replacement policies that include popularity calculations of content, for example, are not considered since they are not implementable at linespeed. In fact, LFU is the upper limit on complexity with regard to computational requirements and even then some authors believe that LRU should be considered the upper limit for complexity [48].

In terms of performance, the authors in [30] have indicated that LRU sit he superior replacement policy. They do note that in areas that require high-speed caching (particularly at the higher-level caches that serve more requests) the Random policy may be more suitable since it requires minimal computation and performs almost as well as LRU. Given our focus on the performance of routing and the caching decision policy together, **we will elect to use the optimal linespeed policy, LRU, for our simulations**. We recognise the potential for variation of the replacement policy in future work.

4.4 Content Distribution

NDO Granularity

First of all, NDO granularity is set to 1 meaning that a complete object is represented by a single packet. This is unrealistic given the multimedia nature of web content, but it helps to simplify the model allowing a focus on the key contribution made in this paper. Furthermore, the effect is only to rule out mechanisms or policies that rely on chunk-level differentiation (such as WAVE) since there would be no perceivable change in the statistics should the ratios outlined below remain constant.

Catalogue Size, Cache Size - Cache to Catalogue Size Ratio

Cache size (C) is homogeneously set to 10^2 while catalogue size (N) is set to 10^5. While these values are minute compared to real-world scenarios, this simulates a realistic cache to catalogue size ratio (C/N) of 0.1%. In reality, any ICN system should be prepared to handle at least 10^{12} objects (based on the current size of the web) [31]. The chosen cache to catalogue size reflects much of the prevailing literature, both in terms of ICN and IP [51].

Intuitively, if we maintain popularity distribution and $C/N = 0.1\%$ but increase the values of C and N, then most caching mechanisms will perform better (reducing average hop count). This is because despite the ratio being maintained, each cache is able to store more of the most popular content and satisfy more of the most regular requests in-network. Crucially, the relative difference across caching mechanism with varying C and N is small and as such their values are less important than the ratio itself. We choose low values of C and N, 10^2 and 10^5 respectively, since it is easier to graphically show the over/under performance where there is greater absolute disparity.

Name Cache Size

Finally, we fix Name Cache size as capable of caching the same number of names (10^2) as the cache itself is capable of caching content. This is required for the multi-layered filtering technique used in our novel caching mechanism, Filter First Caching (see chapter 6).

Cache Homogeneity vs Heterogeneity

We elect to use homogeneous cache sizes to delineate the caching mechanisms into the agreed upon strategies above. Mechanisms that exhibit heterogeneous caching provision techniques were removed from consideration in section 3.3 due to their complexity, difficulty in simulation and idiosyncratic nature, in terms of distinct ISPs.

Cache Placement

Cache placement will be ubiquitous unless otherwise stated. For example, to illustrate the effect of caching solely at edge nodes, cache placement will be reserved for those leaf nodes only. In all other scenarios, every content router will be capable of caching, with the same capacity (C).

4.5 Content Popularity

Substantial research has gone into attempting to understand the distribution of web requests and construct a suitable model to replicate real-world consumer request patterns in a simulated environment. Indeed, there is evidence to suggest that the catalogue and popularity settings play by far the most crucial role in the outcome of an ICN simulation [50]. Note that this observation applies to *absolute* performance in network scenarios: *relative* performance of differing ICN set-ups remains mostly constant. It is now commonly accepted that the distribution of page requests generally follows a Zipf-like distribution [18] which is defined as:

$$p(X = i) = (1/i^{\alpha})/Y \tag{4.1}$$

where,

$$Y = \sum_{j=1}^{O} (1/j^{\alpha}) \tag{4.2}$$

and i is the rank of the ith most popular file in a catalogue of size O.

That is to say that the relative probability of a request for the ith most popular page is proportional to $1/i^{\alpha}$. The request distribution is said to be strictly following Zipf's law when $\alpha = 1$. To summarise the effect of varying α, the higher the value, the greater frequency of requests for the most popular pieces of content relative to less popular content.

While we can pursue a Zipf-like distribution provided by the content distribution class of ccnSim, there is no consensus for the correct settings for it. Throughout the literature, the value of α varies in a wide range from 0.5, to over 2 in some frameworks [20].

Furthermore, it is intuitively clear that the selection of the content popularity factors will greatly affect potential results [49]. If we consider the percentage of requests for the set of k most popular contents would be defined by:

$$\sum_{i=1}^{k} p(X = i) \tag{4.3}$$

We can perceive that as α increases, the cache size required to satisfy the vast majority of Consumer requests becomes unrealistically small and indeed the caching mechanism virtually irrelevant. For example, if we assume our catalogue size of 10^5 and boldly look to cache the files required to satisfy 90% of user requests in a single cache. In Figure 4.5, we show the values of k, and therefore the requisite cache size, when varying alpha over these assumptions.

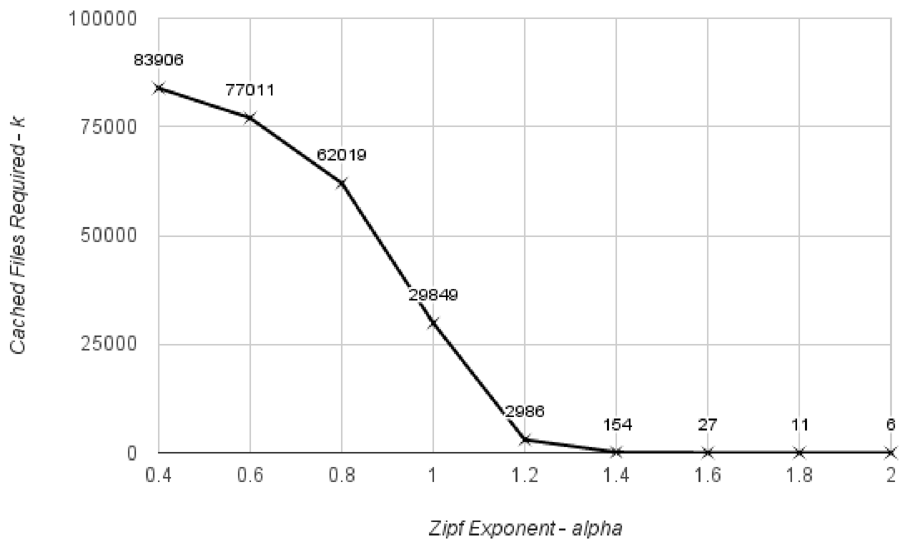


Figure 4.5: Cached files required to satisfy 90% of user requests under varying settings of alpha for a Zipf-distribution to simulate content popularity - catalogue size of 10^5

As you can see, at values of $\alpha \geq 1.4$, the cache would be required to hold fewer than $k = 154$ files to satisfy 90% of user requests (since $\sum_{i=1}^{154} P(X = i) > 0.9$ for $\alpha = 1.4$). Given our cache size (C) of 100 files, the expected cache hit of every caching mechanism regardless of other scenario parameters would be exceptionally high. From a real-world hardware perspective, the analysis done on YouTube traffic in [32] suggests that only 10% of the videos requested are larger than 21.9 MB. So if we cautiously assume average file size of 20MB, then content routers of 20MB * 154 = 3.08GB could service 90% of user requests at $\alpha = 1.4$. Bearing in mind a conservative estimate of content router cache size of 10GB [14], an α of 1.4 for our Zipf-distribution, clearly does not accurately reflect reality.

We observe a sharp shift to more realistic cache size required to fulfil 90% of user requests as α tends to 1. Indeed, we believe that a realistic picture of popularity distribution can be obtained with $\alpha \in [0.5, 0.6, 0.7, 0.8, 0.9, 1]$ where 0.5 and 1 act as boundary figures. What's more, the analysis on YouTube traffic in [32] proposes an α of 0.56 and original discussions in [18] hint at an α between 0.64-0.83. Thus, we feel this range of settings fairly reflects both the literature and our own findings.

For feasibility purposes, we elect a static α for the most computationally expensive scenarios. We choose $\alpha = 0.9$ as it lies at the upper fringe of our observed range, thereby facilitating greater absolute values and better highlighting *relative* performance differences. We recall that varying α will affect absolute performance, but relative performance of our routing and caching combinations should remain unaffected.

Please note, we make available our C++ program and relevant script in the attached archive for generation of the underlying graph data should an interested reader wish to pursue this central topic further.

Finally, we observe that using a synthetic workload is sub-optimal in terms of real-world replication. However, it does provide a technically sound stream of requests and would most likely be a conservative replication given they neglect temporal request correlation (i.e. end-users request same trending content at similar times).

4.6 Set-Up

Starting State

We start each simulation from a cold starting point, in other words with each in-network cache empty. However, performance metrics are only recorded after the cache hit ratio for the active caches reaches a steady state. In this way, we fill the caches with a fair representation of the Zipf distribution specific to the simulation before taking statistics. This further assists in the replication of real-world scenario where caches will neither have hand picked cache contents (at least not for our non-specific ISP) or have empty caches.

The stabilisation process begins when the proportion of total nodes specified by the value, Partial Nodes (pN), is full (or empty for inactive nodes). We set $pN = 1$ since we want all active nodes to be full. However, in edge caching scenarios we use $pN = 0.5$ since many of the nodes are without caches and the simulation would fail to proceed past the fill phase.

Stabilisation is achieved when the variance of the cache hit probability of pN proportion of the content routers is less than the Variance Threshold (V). This is implemented by sampling the hit probability of each node, and then calculating the variance of the samples collected. A sample is collected every sT (Sampling Time) seconds with the variance of those samples calculated every cW (Control Window).

For our scenarios, we will collect a sample every 0.1 seconds ($sT = 0.01$) with variance calculated every 60 seconds ($cW = 60$). Thus, the variance is repeatedly calculated with 600 samples until it is lower than the threshold (cautiously set to $V = 0.05$), at which point that node is considered stable. We select these values for stabilisation on the recommendation of ccnSim creators in previous scenarios. They offer a fair balance between accurate starting points for performance analysis and speed of simulation.

Running State

After stabilisation has been reached, each simulation will run for a Sim Time of 600s ($T = 600$) so that performance metrics can be collected. Furthermore, Consumer requests are exponentially distributed across all Consumers, modelled as a Poisson process, with intensity $\lambda = 20$ requests per second. This roughly equates to each Consumer sending 12,000 requests for content (following our afore-mentioned Zipf-distribution).

To put that in perspective, for a **single** strategy-layer combination in our ternary tree topology outlined in section 4.2 we will be simulating:

20 repetitions * 10 link states * 27 consumers * 12,000 requests = 64.8 million requests

Note that that is only for the simulation time after stabilisation, which for strategies like Filter First Caching (see chapter 6) can take $T > 12000$.

4.7 Performance Metrics

First of all we reiterate the measurable benefits of ICN that we outlined in section 2.2.3:

1. Lower content delivery latency
2. Reduced traffic
3. Alleviate Producer server load

Now we outline the metrics we will use for our simulation to evaluate the success of each caching mechanism in achieving these benefits. We hinted at these in section 3.3.

Hop Count

Intuitively, data that travels through fewer nodes, has come back quicker and caused less overall network congestion. Therefore, in order to assess the effect of each caching mechanism on (1) and (2) above, we use the *hop count* performance metric (at time t).

$$hopcount, hc(t) = \frac{\sum_{c=1}^{C} hop_c(t)}{C} \tag{4.4}$$

where

$$C = \text{no. of Consumers making requests in network} \tag{4.5}$$

$$hop_c(t) = \frac{\sum_{d=1}^{D} hops_d(t)}{D} \tag{4.6}$$

$$D_c = \text{no. of data packets received at Consumer } c \tag{4.7}$$

$$hops_d(t) = \text{no. of hops data packet } d \text{ has travelled} \tag{4.8}$$

As mentioned in section 3.3, we do not directly address response latency since that requires an inspection of link delay aware routing mechanisms among other additional concerns. We recognise this as future work.

Inner Hit Ratio

In order to assess the reduction in Producer server load, (3) from above, we calculate the *inner hit ratio* (at time t). This is the whole network proportion of total downloads made by Consumers that are satisfied by in-network caches, and not by Producer servers.

$$innerhitratio, ih(t) = \frac{\sum_{c=1}^{C} downloads_c(t) - \sum_{p=1}^{P} serverlookups_p(t)}{\sum_{c=1}^{C} downloads_c(t)} \tag{4.9}$$

where

$$downloads_c(t) = \text{no. of satisfied requests for Consumer } c \tag{4.10}$$

$$P = \text{no. of Producers with content servers} \tag{4.11}$$

$$serverlookups_p(t) = \text{no. of times Producer } p \text{ has accessed server} \tag{4.12}$$

Chapter 5

Coupling Name Resolution and Caching

In this Chapter, we expand on work done in [51] by demonstrating that the benefits of ICN can only be understood where caching and routing implications are intrinsically linked. Using our novel probabilistic topology, outlined in section 4.2, we are able to show that ubiquitous-caching can significantly outperform existing naive edge-caching where caching mechanism and name-resolution are optimised simultaneously.

5.1 Outline

The perceived benefits of in-network caching are one of the main arguments for shifting the underlying currency of the Internet to uniquely named data. Yet, there are those that argue those benefits are over-exaggerated [24, 28, 31]. They attempt to prove that ubiquitous caching does not substantially improve performance over caching at the edge of the network, as with existing CDN. Further, they provide results which support the fact that Edge caching with an optimal forwarding policy (NRR, see section 4.3) can perform similarly to ubiquitous caching algorithms. However, while they elect an optimal forwarding policy for their simulation, they fail to consider improvements in the caching mechanism. They only consider the default Leave Copy Everywhere (LCE) mechanism and, as such, they grossly underestimate the potential performance benefits of ICN. Indeed, this lack of simultaneous optimisation is rife through the literature. We look to support the claims made in [51] that coupling caching and forwarding is essential for understanding the benefits of ICN.

As we mentioned in section 4.2, while [51] makes the correct stipulations, we find flaws in the design of it's topology and therefore, the simulation itself. In scenario A we run a similar simulation with our corrected topology of a ternary tree with probabilistic links. Further to this, we can use the same topology to test our novel caching mechanism in scenario D, Chapter 6.

5.2 Scenario A

Purpose

To illustrate that the benefits of ICN's ubiquitous caching, over CDN-like edge caching, can only truly be understood with a combination of optimisation over name-routing policy and caching mechanism.

Parameters

Scenario A uses our purpose-built ternary tree topology detailed in section 4.2 which includes probabilistic linking.

Since the purpose is to compare the benefits of ICN caching mechanisms with and without routing optimisation, we include simulations over both SPR and NRR as examples of lower an upper performance bounds respectively.

Our selection of caching mechanisms are the most simplistic devices ranging from the default of LCE to the randomised Fix(p). Given their simplicity, we can focus on the value of combining them with routing as opposed to a cross comparison which comes in chapter 6.

Edge caching is simply the LCE mechanism implemented at only those nodes closest to the consumers (see edge node indication in figure 4.1) while all other nodes have caching capacity of 0. This roughly reflects the current state of Content Delivery Networking in contemporary Internet.

Finally, as we stated in section 4.5, we fix the Zipf-exponent of our content popularity distribution within our observed range at $\alpha = 0.9$. This is necessary for simulation feasibility.

Name	Values	Additional Comments
Network Topology	Ternary Tree Topology	4-level and 5-level
Routing Strategy	{SPR, NRR}	Shortest Path / Nearest Replica Routing
Caching Mechanism	{Edge, LCE, LCD, FIX(p)}	p = {0.1, 0.01}
Zipf Exponent (α)	0.9	For practical purposes, we fix α

Table 5.1: Parameters for scenario A that vary from general parameters (table 4.1)

Results

First of all, we assess the sensitivity of each caching mechanism to improved routing policies and improved network connectivity.

Figure 5.1 illustrates the change in hop count of each caching mechanism across both SPR and NRR with varying presence of probabilistic links. For each of our chosen mechanisms we can see that Nearest Replica Routing, together with improved network connectivity, improves performance by up to 10%. Bearing in mind the scale of Internet requests per second (in the order of a millions), an improvement of 10% is considerable.

The other key observation from our initial sensitivity test is the out-performance of fix(0.01) over fix(0.1). We henceforth exclude fix(0.1) from our analysis for ease of representation.

In figure 5.2, we can see the hop count comparison of spr-Edge (naive CDN-like edge caching) and nrr-Edge (intelligent CDN-like edge caching) with our routing-caching combinations. As the authors in [24, 28, 31] suggested, there is little difference in performance between ubiquitous caching and edge caching where the routing-caching combination chosen is spr-lce. At this lower bound of ICN capabilities, the performance gain is only 3.7%. However, if we consider the optimal combination

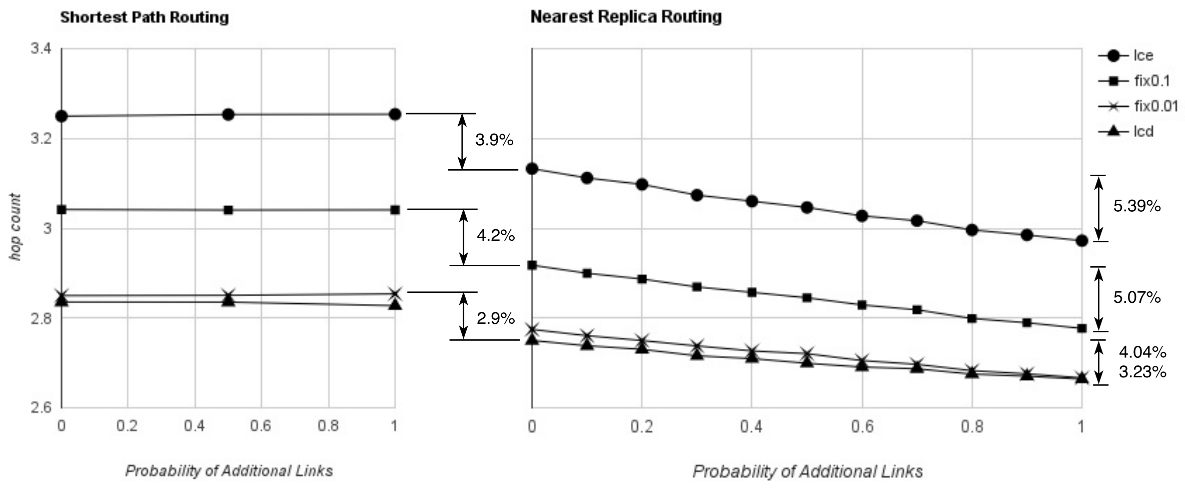


Figure 5.1: Sensitivity of Routing-Caching pairing to presence of probabilistic links

(upper bound), then we observe a substantial improvement of circa 20%. Given the context of Internet traffic, that represents truly monumental improvements, particularly in terms of reduced congestion.

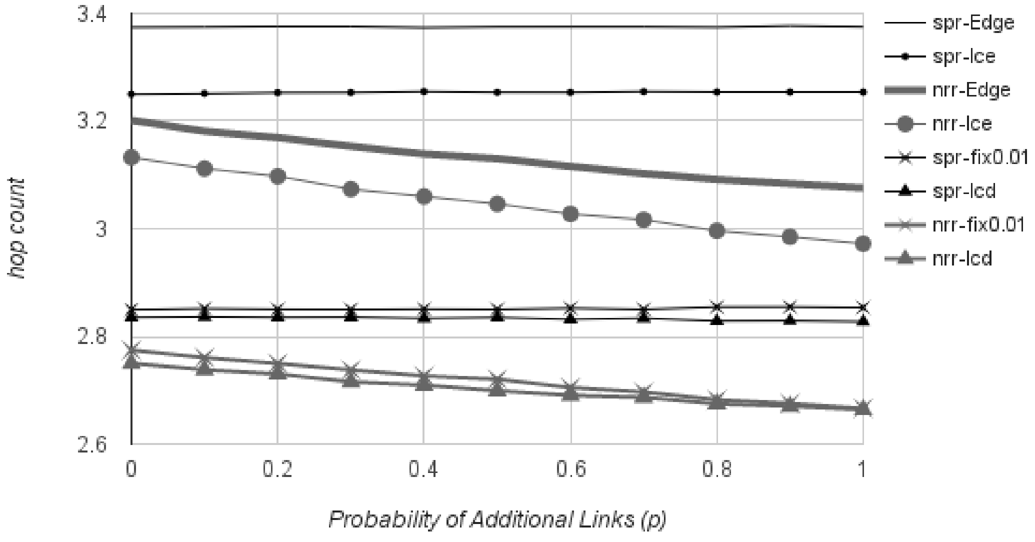


Figure 5.2: Performance (**hop count**) of CDN vs ICN strategies with various routing-caching combinations, as a function of presence of probabilistic links (p)

Next, we consider our other performance metric, inner hit ratio, to understand the true benefits of ubiquitous caching with regards to alleviating the Producer server load. Intuitively, one would expect this to be reduced by there being more in-network caching capacity than simply caching at the edge. The results in figure 5.3 affirm these thoughts emphatically. The chart depicts the relative performance gain, in terms of inner hit ratio, of each of our routing-caching combinations against the CDN-like naive edge caching combination (spr-Edge).

In the optimal scheme, inner hit ratio is improved by over 200%. Indeed, at $p = 1$, fix(0.01) has an inner hit ratio of circa 47.5%. This means that servers deal with almost half as many requests than with no in-network caching, thereby drastically reducing the bottleneck scenarios common in IP. It is worth noting that while fix(0.01) and lcd perform similarly in terms of hop count reduction, fix(0.01) out-performs in terms of inner hit ratio. This is because the random-based policy caches some of the less popular content throughout the network by chance, thereby increasing the total

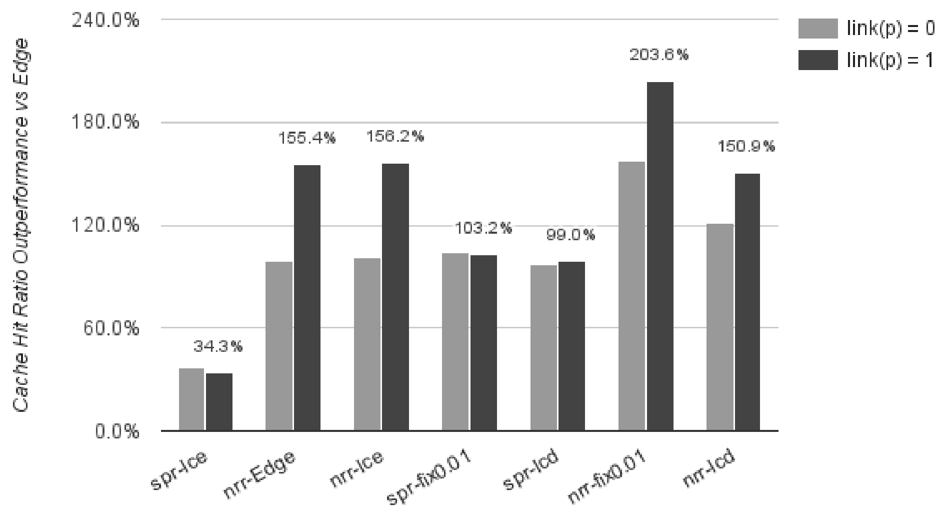


Figure 5.3: Percentage performance gain (**inner hit ratio**) of each of our caching-routing combinations against CDN-like edge caching (spr-Edge)

percentage of the catalogue that can potentially be satisfied in-network. What's more, fix(0.01) has a more strict policy on the node closest to the Producer server: with lcd, the node one hop from the Producer has an lce-like policy.

Figure 5.3 also highlights the effect of improved network connectivity ($p = 1$ vs $p = 0$): in some cases the relative gain increases by over 50%. As you would expect, the more paths on which a router can identify a nearer replica, the less likely that router will have to send the request all the way back the Producer server. Since most real-world networks are far more inter-connected than our tree topology, this finding helps support the theory that ICN would thrive in the existing Internet infrastructure.

Effects of Ternary Tree Size

Now that we have seen the benefits of ubiquitous ICN caching over CDN-like edge caching, we investigate the effect of extending the network size. The above analysis is done on our 4 level ternary tree, the construction of which we outlined in section 4.2. In table 5.2, we summarise the performance gains of our various routing and caching combinations against edge caching with Shortest Path Routing.

	Shortest Path Routing			Nearest Replica Routing			
link p	lce	fix0.01	lcd	Edge	lce	fix0.01	lcd
0	3.7%	15.5%	16.0%	5.1%	7.2%	17.8%	18.5%
1	3.8%	15.4%	16.2%	8.9%	11.9%	21.0%	21.1%

Table 5.2: Percentage performance gain (**hop count**) vs CDN-like edge caching strategy with *4-level* ternary tree

As you can see, ICN caching with Nearest Replica Routing can improve network performance by up to 21% when the network shows a high degree of fault tolerance with additional links ($p = 1$).

In table 5.3, we show the same performance gain but for our ternary tree with 5 levels (121 nodes).

We observe that performance gain is enhanced across all routing and caching combinations. Intuitively, one would expect the greater the network size, the more value of in-network caching. This is because there are more nodes to cache varied content and more value in ensuring the most popular content is filtered towards the edge routers (e.g. the caching redundancy optimisation of lcd and fix(p)). This simple comparison helps confirm that intuition.

	Shortest Path Routing			Nearest Replica Routing			
link p	lce	fix0.01	lcd	Edge	lce	fix0.01	lcd
0	4.6%	16.6%	17.1%	7.8%	9.9%	20.4%	20.5%
1	4.6%	16.7%	17.6%	13.0%	16.2%	25.0%	23.9%

Table 5.3: Percentage performance gain (**hop count**) vs CDN-like edge caching strategy with *5-level* ternary tree

The other observation is that as the network size increases, the value of Nearest Replica Routing is increased more than with Shortest Path Routing. This reflects the fact that with a greater topology size and an optimal routing policy, Consumer's can expect to identify cached replicas of their content in-network more often.

Furthermore, we can see from comparing tables 5.2 and 5.3 that as the network size increases the performance gain of random-based policies such as fix(0.01) increases. This is because the total in-network capacity is increased so if some more unpopular content is randomly cached nearer to the Consumers then it does not have such an effect on the caching redundancy. In fact, occasionally this less popular content will be re-requested thereby improving performance over those strategies that are very unlikely to cache unpopular content near to the Consumer at all (lcd). We bear this in mind with the construction of FFC(p) in chapter 6.

5.3 Evaluation

What we have illustrated in this chapter is the regular underestimation of ubiquitous ICN caching in the literature. We stipulate that those seeking to demonstrate that ICN does not offer significant gains must be able to show so with consideration of optimal routing and caching policies simultaneously. Indeed, we have taken a step to showing the possible gains of identifying a routing policy that can perform close to Nearest Replica Routing is worth the additional network complexity. Ultimately, we provide proof that ICN in-network caching is better than the current edge-caching solutions of Content Delivery Networking.

We appreciate that our analysis is not complete, and is certainly not an optimisation, but it serves to illustrate the potential of ICN. It is this upper bound potential, a possible performance gain of over 20% in a conservative environment, which justifies the ongoing research into this possible future internet architecture.

Chapter 6

Filter First Caching

In this chapter we submit that suitable ICN routing policies should be able to optimise cache availability. As such, we attempt to identify a caching mechanism that focuses solely on reducing caching redundancy. We run multiple scenarios to test the efficacy of our solution to existing mechanisms and evaluate it's worth.

6.1 Outline

Building on the ideas of multi-layered caching suggested by the work available in [2], we propose the caching mechanism Filter First Caching (FFC) and variations upon it.

The basic concept is the addition of a *name cache* (also LRU) to each content router. Every time data passes through the router, it's name is added to the name cache, or it' s position in terms of LRU is updated. If that data's name is present inside the name cache, then it can be considered for caching in the main cache. In this way, unpopular content, in particular what are known as "singleton" requests, will be filtered out of the caching process, greatly reducing caching redundancy.

Table 6.1 shows the four novel caching mechanisms categorised in line with existing mechanisms in section 3.2. We outline the differences between the variations of Filter First Caching below.

Short Name	Homogeneous vs Heterogeneous	Cooperation	On-path vs Off-path	Ref.
FFC	Heterogeneous	Non-cooperative	On-path	n/a
FFC(p)	Heterogeneous	Non-cooperative	On-path	n/a
TFC	Heterogeneous	Implicit	On-path	n/a
WFC(p)	Heterogeneous	Implicit	On-path	n/a

Table 6.1: Categorising Filter First Caching

As we can see from table 6.2, the need for a name cache adds additional router cost to our caching strategies. However, we must note that the name cache is only required to store the content names, not the data itself. As such, it's storage capacity needs to only a minute fraction of the main cache itself. What's more, the name cache size can vary from router to router to better reflect the content popularity distribution of each domain. One could envisage an intelligent router being able to adapt the partition of memory reserved for name caching and data caching based on packet analysis.

Short Name	Packet Cost	Router Cost	Network Cost
FFC	n/a	Name Cache	n/a
FFC(p)	n/a	Name Cache	n/a
TFC	flag bit (Data)	Name Cache	n/a
WFC(p)	flag bit (Data)	Name Cache	n/a

Table 6.2: Costs of Filter First Caching

Variations

While this multi-layered filtering technique is not unique in hardware, combining it with existing low-cost caching solutions for in-network caching is a novel solution to ICN. Based on the findings we made in chapter 5 and preliminary testing, we identify Fix(p) and LCD as the most promising of the pre-selected caching mechanisms. Now, we attempt to combine the caching redundancy benefits of those strategies with the multi-layered filtering of Filter First Caching.

In figure 6.1, we illustrate the mechanics of a multi-layered filtering cache. For each of our variations on Filter First Caching, we supply the requisite implementation to complete the process in the subsections below. Note that should a request for data be satisfied from the router's main cache, then that content's position is updated in both the main and name caches (in terms of LRU). The implementation variations are relevant when a router receives a data packet which has *not* already been stored in the main cache.

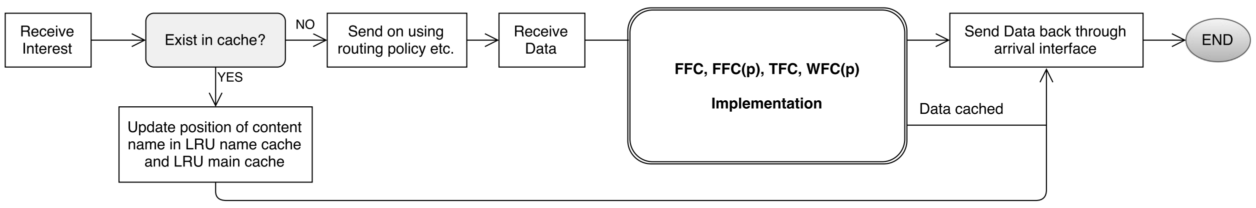

Receive Interest
Exist in cache?
NO
Send on using routing policy etc.
Receive Data
FFC, FFC(p), TFC, WFC(p) Implementation
Send Data back through arrival interface
END
YES
Update position of content name in LRU name cache and LRU main cache
Data cached

Figure 6.1: COMMENT

FFC

Filter First Caching (FFC) utilises the multi-layered filtering system of the name cache but puts no further restrictions on caching (similar to Leave Copy Everywhere). Should a data packet arrive with a content name that can be found in the name cache, then it can be added to the main cache (should it not be already). Figure 6.2 illustrates the basic implementation of FFC.

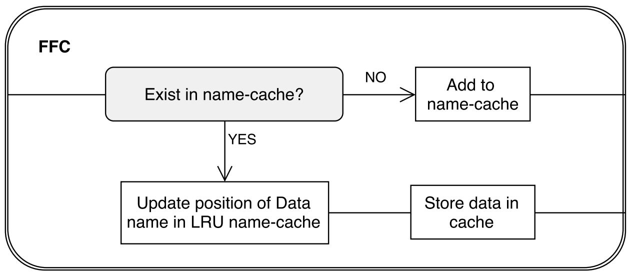


Figure 6.2: Implementation of FFC to complement the mechanics in figure 6.1

FFC(p)

Filter First Caching(p) (FFC(p)) extends FFC with the addition of a random-based element, similar to Fix(p). The intention is to reduce caching redundancy still further. Even if an arriving data packet matches it's content name to a record in the name cache, it is only cached if a generated random number is less than a fixed value for p. Note that FFC(1) is the same as FFC.

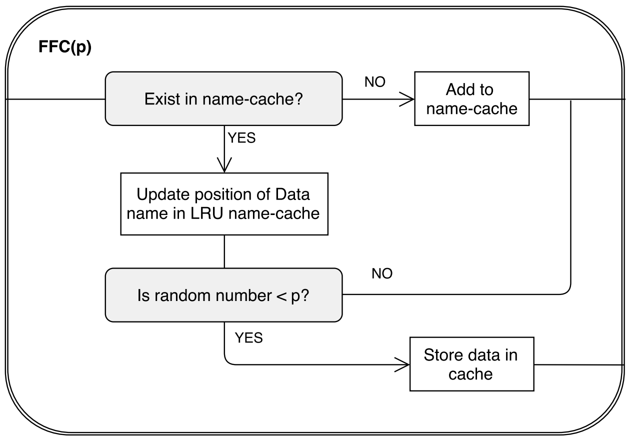


Figure 6.3: Implementation of FFC(p) to complement the mechanics in figure 6.1

Figure 6.3 demonstrates the implementation. Note that even if the data is not stored in the main cache, it's LRU position is updated in the name cache. This is to make it more likely for popular content to at least achieve a Fix(p)-like chance of being cached.

Given the multiple layers of caching redundancy, we envisage the optimal value of p for FFC(p) to be far greater than that of Fix(p). With the name cache functionality able to filter out unpopular content requests, the second random-based filter should only be for anomalous request patterns.

TFC

Having explored the values of Fix(p) with our multi-layered caching, we pursue the potential of LCD-like second layer filtering. Trickle Filter Caching (TFC) is designed to only cache the truly most popular content close to the Consumer, virtually ignoring the remaining distribution of requests.

Figure 6.4 illustrates the implementation of TFC. If an arriving data packet matches it's name within the name cache, then it will be cached provided it has only traversed a single hop. In this way, the most popular content will trickle it's way down to the end Consumers while all other content will only be cached at those nodes nearer to the Producer servers.

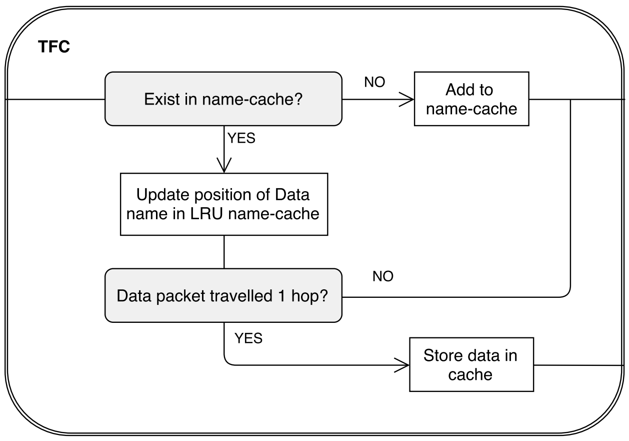

TFC
Exist in name-cache?
NO
Add to name-cache
YES
Update position of Data name in LRU name-cache
Data packet travelled 1 hop?
NO
YES
Store data in cache

Figure 6.4: Implementation of TFC to complement the mechanics in figure 6.1

WFC(p)

In chapter 5, we observed that Fix(p) and LCD perform similarly with our hop count performance metric. However, in terms of inner hit ratio, Fix(p) significantly outperformed LCD. This is due to the fact that LCD's trickle down nature ensures that only the most popular content is cached near to Consumers, but it is only that most popular content that is cached in the network. All non-popular content is only available at the Producer server, or perhaps, at those nodes closest to it.

In an effort to increase inner hit ratio of our trickle down multi-layered caching mechanism (TFC), we introduced Weak Filter Caching(p) (WFC(p)). The concept is the same as that of TFC, in that the truly most popular content finds it's way to the nodes closest to the Consumer. However, we also add a weak filtering characteristic to allow some non-popular content to be randomly cached through the network (in line with Fix(p)). Note that WFC(0) is the same as TFC.

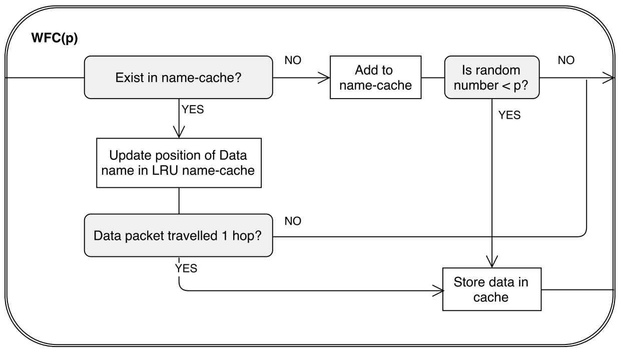


Figure 6.5: Implementation of WFC to complement the mechanics in figure 6.1

In figure 6.5, we can observe the implementation of WFC(p). In short, if data is not present in the name cache, it still has a p% chance of being stored in the name cache. In this way, we expect the inner hit ratio to be increased since the content routers should show greater diversity of cached content.

With FFC(p), our random element applied to the second layer of filtering. Here, the random element of WFC(p) applies to the first layer of filtering (the name cache). As such, we expect the value of p to be much lower, more in line with Fix(p). We still do not want unpopular content overly polluting our content routers' main caches.

6.2 Scenario B

Purpose

In this scenario, we test the performance of our multi-layered caching mechanisms in a simple Bus Topology against our pre-selected caching mechanisms. The simplicity of each simulation allows us the opportunity to explore the effects of varying α and validate the use of $\alpha = 0.9$ as our fixed Zipf-exponent for our popularity distribution.

Parameters

Scenario B uses our simple bus topology detailed in section 4.2.

Having ascertained that Nearest Replica Routing demonstrates the potential of ICN caching, we proceed to test our novel caching mechanisms under that routing strategy only.

Owing to simplicity of topology, it is feasible for us to include all chosen caching mechanisms. What's more, we can further test the full extent of our observed range for α with regards to content popularity distribution (see section 4.5).

Name	**Values**	**Additional Comments**
Network Topology	Bus Topology	1 repetition due to no random elements
Routing Strategy	NRR	Nearest Replica Routing
Caching Mechanism	{LCE, Probcache, LCD, FIX(p), FFC, FFC(p), TFC, WFC(p)}	p = {0.25, 0.1, 0.05, 0.01, 0.005, 0.001}
Zipf Exponent (α)	[0.5, 1]	increments of size 0.1

Table 6.3: Parameters for scenario B that vary from general parameters (table 4.1)

Results

First of all, we look to assess the optimum value of p (p_{opt}) for Fix(p), FFC(p) and WFC(p) in our bus topology. In figure 6.6, we illustrate the performance of each strategy with different values of p, over varying values of α within our observed range.

As you can see, p_{opt} varies for each strategy.

Fix(p): $p_{opt} \approx 0.005$

FFC(p): $p_{opt} \approx 0.2$

WFC(p): $p_{opt} \approx 0$, equivalent to TFC

Zipf-Alpha:	0.5	0.6	0.7	0.8	0.9	1
fix0.1	5.417	5.315	5.091	4.682	4.027	3.205
fix0.05	5.401	5.285	5.029	4.595	3.948	3.119
fix0.01	5.368	5.195	4.921	4.428	3.751	2.964
fix0.005	5.335	***5.164***	***4.882***	***4.388***	***3.717***	***2.919***
fix0.001	***5.331***	5.176	4.965	4.498	3.991	3.196
ffc	5.284	5.108	4.814	4.362	3.725	2.966
ffc0.25	***5.257***	***5.101***	***4.774***	4.286	3.638	2.868
ffc0.2	5.288	5.111	4.793	***4.285***	3.634	2.852
ffc0.1	5.311	5.165	4.825	4.287	***3.628***	***2.837***
ffc0.05	5.327	5.174	4.944	4.371	3.672	2.871
ffc0.01	5.399	5.258	5.091	4.622	3.959	3.189
wfc0.25	5.439	5.356	5.156	4.780	4.175	3.355
wfc0.1	5.416	5.312	5.087	4.673	4.024	3.203
wfc0.05	5.399	5.288	5.035	4.591	3.938	3.111
wfc0.01	5.362	5.199	4.906	4.439	3.786	2.988
wfc0.005	5.333	5.168	4.870	4.389	3.730	2.956
wfc0.001	***5.301***	***5.133***	***4.840***	***4.349***	3.688	2.915
tfc	5.343	5.189	4.884	4.366	***3.683***	***2.903***

Figure 6.6: Hop count analysis of varying p for fix(p), ffc(p) and wfc(p) with differing Zipf alpha

Using these approximations for p_{opt}, we then compare our novel caching mechanisms with existing solutions. The hop count performance comparison, using Nearest Replica Routing, for increments of α can be seen in figure 6.7.

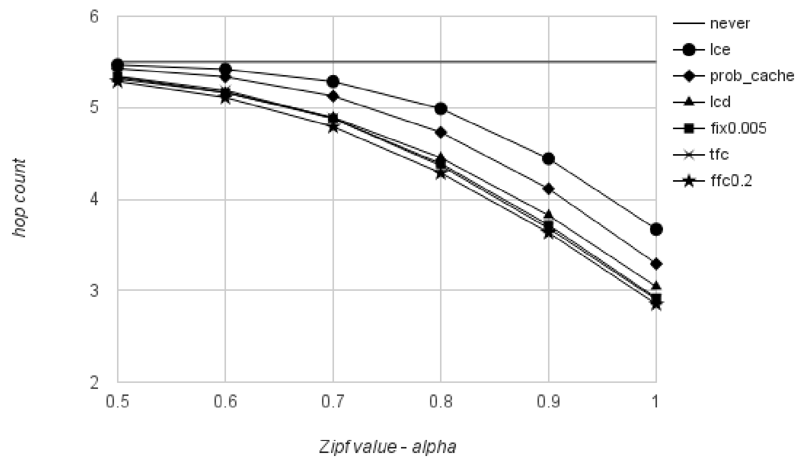


Figure 6.7: Hop count analysis of our selected caching mechanisms, using Nearest Replica Routing, with differing Zipf alpha

As predicted, the value of α has a drastic effect on the efficacy of in-network caching. Indeed, at $\alpha = 0.5$, even our optimal routing-caching combination (nrr-ffc(0.2)) only improves hop count by 3.86% over no caching at all. In contrast, that same combination provides a 48.15% gain over never caching at $\alpha = 1$. We show the percentage hop count improvement of each caching mechanism over never caching in table 6.4. Importantly, the relative performance of each routing-caching combination with respective to each other remains close to constant through different values of α. These findings support our decision to fix $\alpha = 0.9$ for more complex simulations.

With regards to the relative performance of our caching mechanisms (in terms of hop count), we make the following observations from figure 6.7 and table 6.4:

- as expected, Leave Copy Everywhere (lce) caching is the lower bound of ICN in-network caching;

	Existing Mechanisms				Novel Mechanisms	
α	lce	probcache	lcd	fix0.005	tfc	ffc0.2
0.5	0.59%	1.32%	3.36%	2.99%	2.86%	**3.86**%
0.6	1.47%	2.97%	6.09%	6.10%	5.66%	7.08%
0.7	3.90%	6.78%	11.03%	11.23%	11.21%	12.86%
0.8	9.29%	13.99%	19.06%	20.21%	20.61%	22.09%
0.9	19.21%	25.20%	30.49%	32.43%	33.05%	33.92%
1	33.27%	40.10%	44.67%	46.93%	47.22%	**48.15**%

Table 6.4: Percentage performance gain (**hop count**), vs no caching, of our selected caching mechanisms with Nearest Replica Routing, in a bus topology

- probcache substantially under-performs the other caching mechanisms in this simulation, as was the case in other preliminary testing scenarios not detailed here;
- the remaining four mechanisms, namely lcd, fix(0.005), tfc and ffc(0.2), perform similarly in this simple topology and therefore require further testing;
- the top performing mechanism in this topology is our multi-layered novel caching mechanism, Filter First Caching with additional randomised filtering, ffc(0.2).

Table 6.5 includes the inner hit ratio, our alternative performance metric, of our chosen caching mechanisms in the bus topology. The relative results are similar to those we identified with regards to hop count. Once again, the top performing mechanism at all values of α is ffc(0.2); at $\alpha = 1$, an impressive 50.84% of Consumer requests will be satisfied from within the network. The only noticeable difference comes from the increased outperformance of fix(0.005) over lcd: at high values of α in our range, circa 5% more requests are being satisfied in-network. This reinforces previous indications that while lcd and random-based mechanisms perform similarly in terms of reducing latency, the randomised solutions are superior in terms of alleviating the strain on Producer servers.

	Existing Mechanisms				Novel Mechanisms	
α	lce	probcache	lcd	fix0.005	tfc	ffc0.2
0.5	0.69%	1.69%	3.48%	3.82%	3.91%	5.05%
0.6	1.72%	3.65%	6.19%	7.42%	7.44%	8.74%
0.7	4.34%	7.86%	11.19%	13.42%	13.29%	15.17%
0.8	10.05%	15.52%	18.96%	22.92%	22.82%	24.66%
0.9	20.22%	27.37%	30.06%	35.40%	35.57%	36.62%
1	34.60%	42.59%	44.20%	49.93%	49.52%	**50.84**%

Table 6.5: Inner hit ratio (as a percentage) of our selected caching mechanisms with Nearest Replica Routing, in a bus topology

6.3 Scenario C

Purpose

Scenario C extends the performance analysis on FFC and it's variants to 5 real-world topologies. We do this so that interested parties can easily perform cross comparison on Filter First Caching. What's more, we hope to reinforce our conjecture that the relative impact of the network topology is limited (see section 4.2). Finally, the chosen topologies provide a realistic scenario of inter-connectivity in contrast to the bus topology of scenario B.

Parameters

Scenario C compares performance across 5 distinct real-world topologies, detailed in section 4.2.

Having ascertained that Nearest Replica Routing demonstrates the potential of ICN caching, we proceed to test our novel caching mechanisms under that routing strategy only.

From scenario B, and other preliminary testing, we found that Probcache is an inferior solution in most circumstances and as such, is not a worthy comparison for our novel mechanisms. We proceed to test FFC and it's variants with it excluded from the analysis.

In the interest of feasibility, since some of the real-world topologies are particularly large (up to 68 nodes), we limit the values of p to those indicated in table 6.6. These values follow the indications of optimal performance from figure 6.6 in scenario B. Furthermore, we exclude WFC(p) from the simulation with indications in scenario B that it does not improve performance over TFC (same as WFC(0)). We seek to affirm these indications in scenario D.

Finally, we confirmed in scenario B that relative performance remains mostly constant across varying α in the range [0.8, 1]. As such, we validate the use of $\alpha = 0.9$ for our more complex simulations since it offers a realistic distribution of popularity in line with the literature, together with providing sufficient disparity in performance between our caching mechanisms.

Name	**Values**	**Additional Comments**
Network Topology	Real-World Topologies	{Abilene, Tiger, Geant, Dtelecom, Level3}
Routing Strategy	NRR	Nearest Replica Routing

Caching Mechanism	{LCE, LCD, FIX(p), FFC, FFC(p), TFC}	Fix(p), p = {0.05, 0.005}
		FFC(p), p = {0.2, 0.1, 0.05}

Zipf Exponent (α)	0.9	For practical purposes, we fix α

Table 6.6: Parameters for scenario C that vary from general parameters (table 4.1)

Results

In figure 6.8, we can see the outcome of our simulation on the 11 node Abilene topology. The results clearly reflect a similar pattern to that found with our simple bus topology.

First of all, we note that ffc(0.2) is the optimal caching mechanism once again. The results show that ffc(0.2) offers a 15.66% gain in hop count and a 55.69% gain in inner hit ratio, over lce. However, the difference in performance over fix(0.005) is far less obvious. In fact, ffc(0.2) only improves hop count by 2.73% and inner hit ratio by 4.52% when compared with fix(0.005). This suggests that should the optimal value of p for each mechanism be identifiable; the increased multi-layering may not add significant value to random-based strategies.

Interestingly, tfc substantially outperforms lcd proving the effects of the multi-layered filtering are non-negligible in some circumstances. Recall that tfc implements lcd-like trickle caching behind the name cache filtering inherent in all our novel caching mechanisms in this chapter.

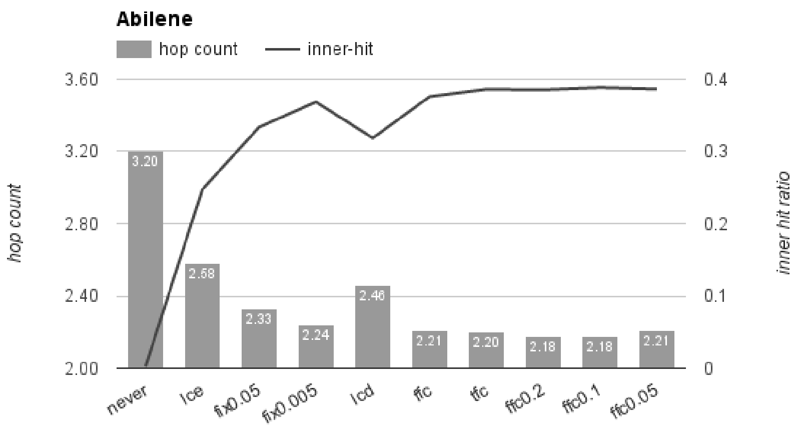


Figure 6.8: Performance (hop count and inner hit ratio) of our selected caching mechanisms, with Nearest Replica Routing, on the Abilene topology detailed in section 4.2

Subsequently, we can see in figure 6.9 that each of the other four topologies produced similar results. Indeed, these findings support the theory that network topology has minimal impact on performance metrics where link latency is uniform [48, 50]. In our topologies, link latency is not uniform but both of our selected performance metrics ignore the effects of delay.

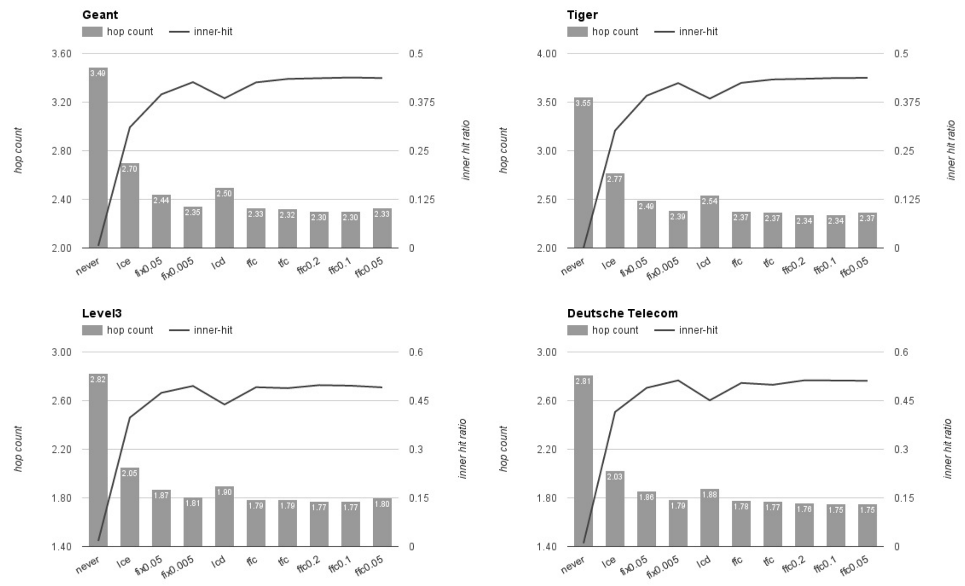


Figure 6.9: Performance (hop count and inner hit ratio) of our selected caching mechanisms, with Nearest Replica Routing, on the Geant, Tiger, Level3 and Deutsche Telecom topologies, detailed in section 4.2

Finally, the fact that all five topologies demonstrate similar results for our selected mechanisms further reinforces the universal efficacy of Filter First Caching combined with Nearest Replica Routing. In all of our tested scenarios, ffc(0.2) has been the optimal caching decision policy.

6.4 Scenario D

Purpose

In the ultimate scenario, we test the efficacy of our novel caching mechanisms in our novel ternary tree topology with probabilistic linking, which we used in scenario A (section 5.2).

We can then see if Filter First Caching demonstrates further benefits of ICN ubiquitous caching over IP edge caching with coupled caching and routing optimisation.

Parameters

Scenario D uses our purpose-built ternary tree topology detailed in section 4.2 which includes probabilistic linking.

Once again, we seek to illustrate the potential of ICN with optimal routing and caching over default alternatives and thus, we simulate with both SPR and NRR routing.

For fix(p), we use the same value of p that we identified as optimal in scenario A. For our novel caching mechanisms, we seek to identify optimal values of p for both FFC(p) and WFC(p).

Name	Values	Additional Comments
Network Topology	Ternary Tree Topology	4-level
Routing Strategy	{SPR, NRR}	Shortest Path / Nearest Replica Routing

Caching Mechanism	{LCE, LCD, FIX(p), FFC, FFC(p), TFC, WFC(p) }	Fix(p), p = {0.01}
		FFC(p), p = {0.3, 0.25, 0.2}
		WFC(p), p = {0.01, 0.001, 0.0001}

Zipf Exponent (α)	0.9	For practical purposes, we fix α

Table 6.7: Parameters for scenario D that vary from general parameters (table 4.1)

Results

In order to identify the optimal p value for our random-based novel caching mechanisms, FFC(p) and WFC(p), we did some preliminary testing (not reported). Following that, we simulated the performance of some selected values of p at the extremes of probabilistic linking (link p = {0,1}). The results, in terms of hop count, are shown in table 6.8.

First of all, we note the confirmation of our belief that the optimal value of p for FFC(p) is around 0.2, as with scenarios B and C. The minute outperformance of ffc(0.25) over ffc(0.2) is within the possible standard deviation from anomalous simulations. Nevertheless, we proceed with ffc(0.25).

Similarly, in previous scenarios, we have found that the weak filtering element of WFC(p) does not add additional value to the functionality of TFC. Once again, we find the same truth that as the p in WFC(p) tends to 0, it's performance improves. As such, we proceed solely with TFC, ruling out WFC(p) as a candidate for optimisation.

	FFC(p)				WFC(p)			
link p	ffc	ffc(0.3)	**ffc(0.25)**	ffc(0.2)	wfc(0.01)	wfc(0.001)	wfc(0.0001)	**tfc**
0	2.691	2.659	**2.658**	2.659	2.740	2.678	**2.667**	2.669
1	2.599	2.573	**2.572**	2.573	2.641	2.596	2.594	**2.591**

Table 6.8: Performance (**hop count**) in 4-level ternary tree with probabilistic linking (link probability = p)

Having finalised our caching mechanisms, we simulate them on the same ternary tree topology we used in scenario A for cross comparison. Table 6.10 shows the hop count performance of our scenario D simulations together with selected routing-caching combinations from scenario A for comparative purposes. Having demonstrated justification for continued research into ICN based on the performance over CDN-like edge caching, we show here further potential gains with our novel caching mechanisms. As has been the case throughout our simulations, FFC(p), coupled with Nearest Replica Routing, reduces the hop count the most of our caching mechanisms. Indeed, with all probabilistic links in place ($p = 1$), nrr-ffc(0.25) outperforms spr-Edge by 23.8%.

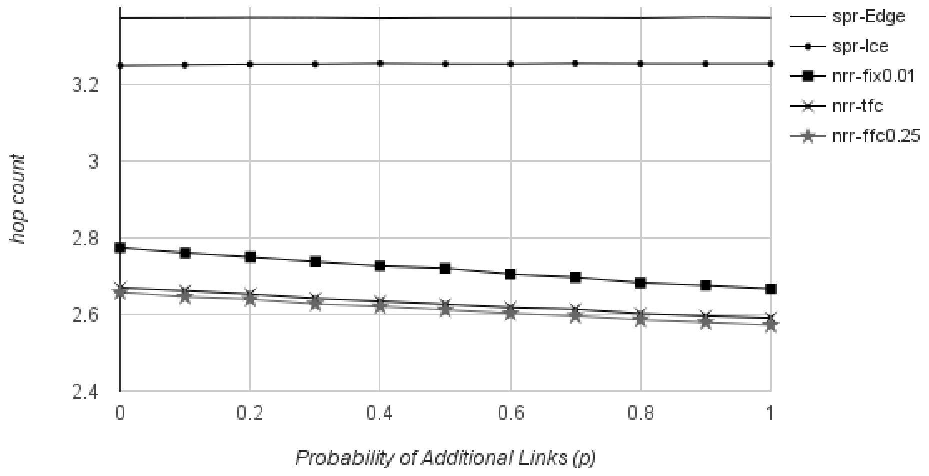


Figure 6.10: Performance (**hop count**) of CDN vs ICN strategies with various routing-caching combinations, including our novel caching mechanisms, as a function of presence of probabilistic links (p)

However, we also note that nrr-ffc(0.25) only outperforms the more simple strategy of nrr-fix(0.01) by 3.5%. What's more, if we consider our second performance metric in table 6.11, inner hit ratio, we notice that nrr-fix(0.01) is in fact the optimal strategy.

We recall that our performance analysis does not directly consider the effects of additional router capacity required for the name cache in Filter First Caching. It is possible that this additional cost (while manageable) could result in a small reduction in performance. Therefore, while our multi-layered caching mechanism is substantially superior to the lower bound of ICN performance (spr-Edge), one could argue that a simple random-based caching mechanism, such as Fix(p). coupled with optimal routing represents a better upper bound.

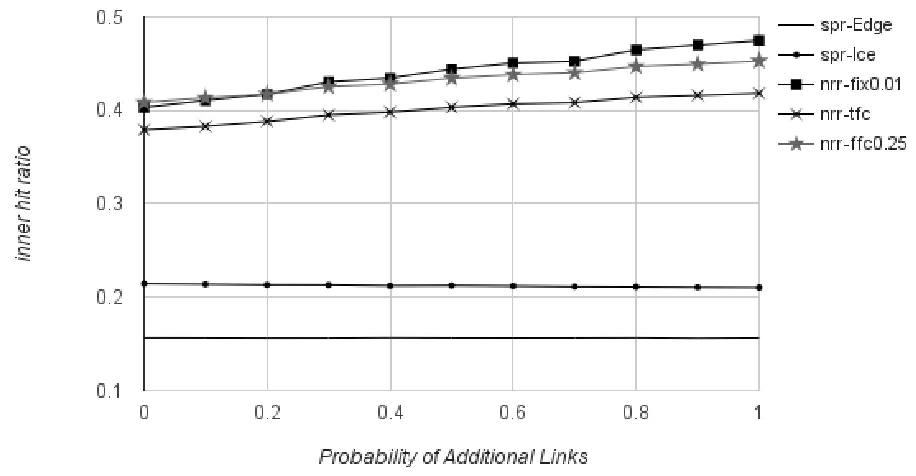


Figure 6.11: Performance (**inner hit ratio**) of CDN vs ICN strategies with various routing-caching combinations, including our novel caching mechanisms, as a function of presence of probabilistic links (p)

6.5 Evaluation

Throughout our analysis in this chapter we identified the potential benefits of Filter First Caching. For all of our simulations, the top performing caching mechanism in terms of hop count was always FFC(p). This serves to illustrate the potential in multi-layered filtering which we proposed here.

In terms of the other variants, TFC and WFC(p), we found them to consistently underperform the more simple FFC designs. Intuitively, one can see that they are either too strict in terms of rejecting content, in the case of TFC, or simply inefficient, in the case of WFC(p). They both seem to suffer from the same lack of diversity of in-network content that we identified between LCD and Fix(p) in chapter 5.

As we came to realise towards the end of our analysis, increasing the enhancements of a mechanism often does not outweigh the possible gains. Indeed, our discovery in scenario D that the upper bound of ICN caching may well be achieved by the simple Fix(p) appears to be strange. However, particularly after taking into account the potential costs of an apparently enhanced strategy such as FFC, we can see that keeping it simple in the case of ICN makes sense.

Since we have shown that multi-layered filtering does indeed add value to ICN in-network caching, it is worth further consideration and additional testing. Yet, as suggested, testing should include a consideration of the cost of the name cache, both in terms of caching capacity and lookup latency.

Integration into ccnSim

In this subsection, we outline the material provided to assist in understanding the integration of our multi-layered filtering mechanisms into ccnSim. Interested parties can then develop further work on this topic using the material made available in the attached archive. See chapter 7 for suggested future work.

In appendix A, we present a snapshot class diagram, in Uniform Modelling Language (UML), of the key elements of ccnSim required for understanding the integration of FFC.

In appendix B, we give pseudo-code outlines of the two central functions of our FFC class in ccnSim. This should assist in the understanding of the functioning of the DecisionPolicy class referenced in the aforementioned UML diagram.

In appendix C, we list some of the key changes to ccnSim that we made besides the integration of FFC that might assist a future user.

Finally, please find installation details of our archive material into the ccnSim framework in the included README file. Note, we provide our own versions of certain files to be patched into the ccnSim source code as the following:

- *include/ffc_policy.h*
- *src/node/core_layer.cc*
- *src/node/cache/base_cache.cc*
- *src/statistics/statistics.cc*

If you follow the included README, our pre-built scripts will perform the integration of the archive into your ccnSim directory.

Chapter 7

Conclusion

In this paper, we set out to make three main contributions. First of all, we wanted to condense the vast and conflicting ICN research arena into an up-to-date and manageable survey of the state of ICN. In particular, we looked to focus on the existing caching mechanisms that have been proposed in the literature, identifying their strengths and weaknesses. In respect of this objective we feel that the initial chapters of this paper offer a fair representation. Nevertheless, we recognise that the survey is not entirely comprehensive and the following areas of ICN research could be added to what we have offered:

- detail on security aspects;
- detail on Producer mobility;
- congestion and rate control in varying network environments (e.g. wired vs wireless);
- commercial incentives;
- software and hardware feasibility;
- specific application of ICN, e.g. Web, Video Streaming, Internet of Things.

Secondly, we sought to rebuff the negative research on ICN suggesting that it's ubiquitous caching offered little or no gains over edge caching provided by contemporary Content Delivery Networks. In the majority of ICN literature, including these pessimistic analyses, we noted that the default parameters for ICN were invariably used. Indeed, even in the occasions where authors recognised the potential for improved caching mechanisms, they failed to consider improvements to the default routing policy of Shortest Path Routing. Given this, we attempted to show that if the routing and caching policies are optimised simultaneously, then the true capabilities of ICN can be understood. With a more realistic picture of the potential of ICN ubiquitous caching, we hoped to show that the continued research into improved routing and caching policies was justified.

To this end, we feel that the simulations performed and reported in chapter 5 were a success. We showed that with an optimal, yet attainable, routing policy coupled with simplistic caching policies we could achieve 20%+ gains in network performance using ICN pervasive caching over CDN-like edge caching. In the context of the millions of packets traversing the network at any given time, that gain is truly substantial. What's more, we demonstrated that as the network size would grow to a more realistic representation of the vast Internet, performance gains were likely to increase as opposed to level out.

While we feel that we were able to support the continued research of ICN in the face of criticism, there are of course ways in which we felt we could have extended the analysis with more time available. These include:

- varying the caching replacement policy from LRU;
- additional performance metric considering actual latency, with consideration of link delays in a topology;
- varying the cache to catalogue ratio, to represent effects of differing content distributions.

Finally, we introduced the concept of multi-layered filtering to ICN in-network caching with our variants of Filter First Caching. We sought to improve the potential benefits of ICN through further reduction of caching redundancy with minimal extra cost to the router. The success of this section of our project was in the identification of multi-layered filtering as the potential basis for further analysis. However, as we hinted in our evaluation in chapter 6, the performance of FFC and it's variants over the simple random-based Fix(p) mechanism likely do not justify the added cost to router capacity. We failed to show sufficient evidence to suggest that Filter First Caching is an optimal solution. This is not definitive since we were unable to accurately test the cost to the router of the additional name cache. We consider this the next step in future work which includes:

- factor router, packet and network costs into evaluation of mechanisms in measurable way;
- if the above evaluation is available, repeat simulations including caching mechanisms with higher network costs for comparison.

In conclusion, we were able to summarise key aspects of ICN in a single survey, support it's continued research with a demonstration of it's potential benefits, and introduce multi-layered filtering in the context of ICN. Our work is made available in the attached archive for future work. We hope that our findings can help in the development of a much needed solution to the growth of content consumption facing the Internet today.

7.1 Future Distinct Projects

Finally, we express some more general ideas that were incubated during the creation of the concepts outlined in this paper. We mention these to assist in the continued ICN research since each could represent an entirely distinct project.

We propose the creation of a caching mechanism that has a consideration of the content type, in particular it's likelihood of change. For example, you could add a measure of how likely the content is to be updated (volatility byte) to a data packet's header. A content router would only cache the content near to the Consumer if it was below a certain threshold. In this way, only content that was unlikely to be changed (e.g. persistent html elements of a website or popular static video) would be cached closer to the Consumers avoiding unnecessary regular replacement in the key caches. To test this policy, one would have to create a popularity distribution that reflects content volatility and then accurately model the regularity of updates to that distribution. What's more, there would need to be a form of policing on the volatility byte to avoid Producers manipulating it to ensure better service for their content to Consumers. We leave these considerations to interested parties.

Throughout our simulations in this paper, we noted that the Leave Copy Down (LCD) successfully pulled the most popular content to the leaf nodes closest to the Consumers. Equally, we identified Fix(p) as successfully reducing caching redundancy through the spreading of diverse content through the network. With consideration of both mechanisms' successes, we propose the construction of domain-specific hybrid ICN caching approaches. This concept was proposed with more complex caching algorithms in [33]. However, we see no reason why the combination of a backbone of simple LCD nodes interspersed with branches of simple Fix(p) nodes could not achieve superior results. As we concluded in our evaluation, when it comes to caching, simple is often better.

Appendices

Appendix A

Partial Class Diagram of ccnSim

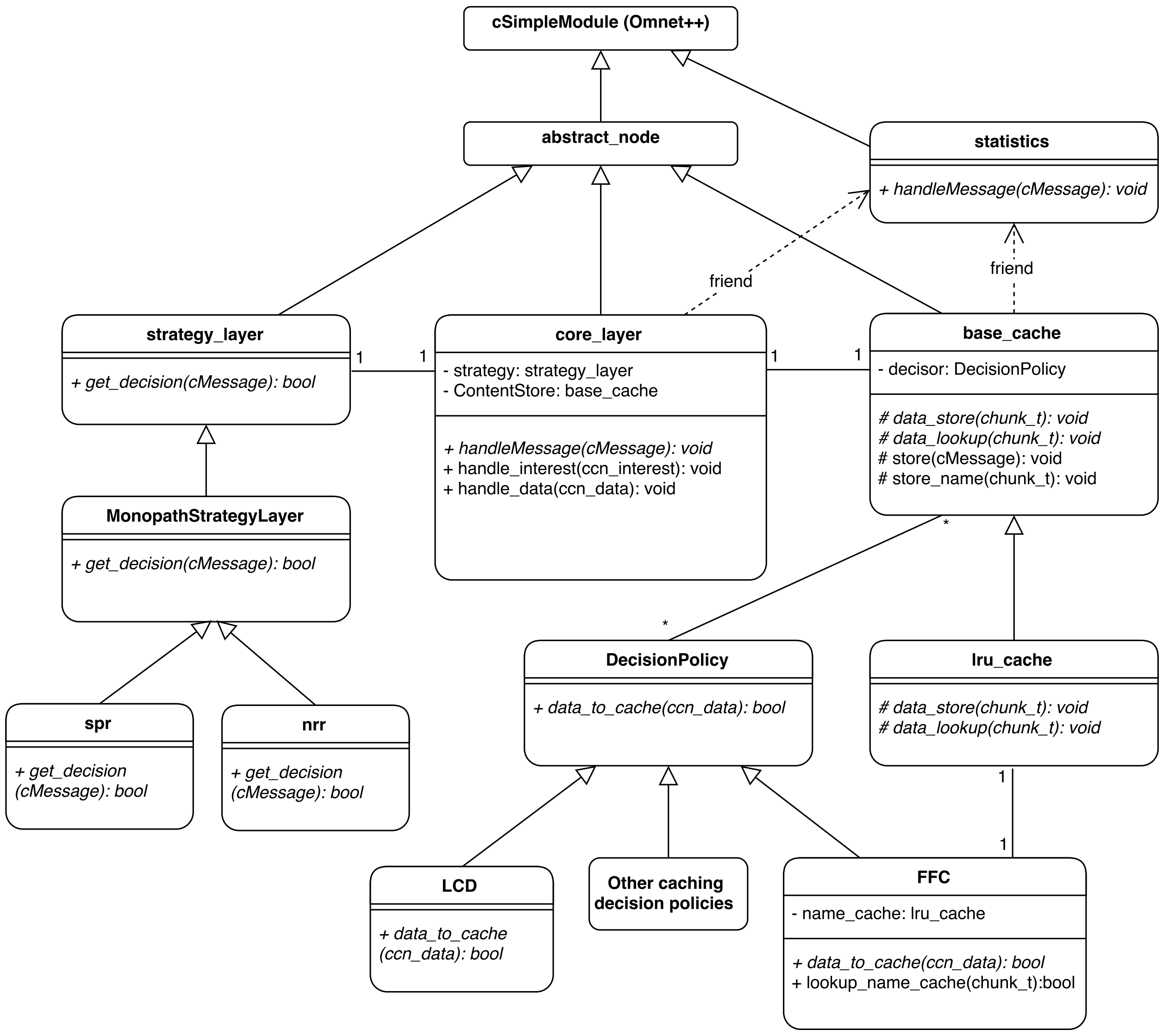

cSimpleModule (Omnet++)
abstract_node
statistics
+ handleMessage(cMessage): void
friend
friend
strategy_layer
+ get_decision(cMessage): bool
1
1
core_layer
- strategy: strategy_layer
- ContentStore: base_cache
+ handleMessage(cMessage): void
+ handle_interest(ccn_interest): void
+ handle_data(ccn_data): void
1
1
base_cache
- decisor: DecisionPolicy
# data_store(chunk_t): void
# data_lookup(chunk_t): void
# store(cMessage): void
# store_name(chunk_t): void
MonopathStrategyLayer
+ get_decision(cMessage): bool
*
*
DecisionPolicy
+ data_to_cache(ccn_data): bool
lru_cache
# data_store(chunk_t): void
# data_lookup(chunk_t): void
spr
+ get_decision (cMessage): bool
nrr
+ get_decision (cMessage): bool
1
1
LCD
+ data_to_cache (ccn_data): bool
Other caching decision policies
FFC
- name_cache: lru_cache
+ data_to_cache(ccn_data): bool
+ lookup_name_cache(chunk_t):bool

Figure A.1: Partial UML diagram of core elements of ccnSim to illustrate the integration of our novel Filter First Caching (FFC) mechanisms

Appendix B

Pseudo-code for key FFC functions

ffc_policy::lookup_name_cache(chunk)

```
if chunk is in Name Cache then
    update chunk name position in Name Cache (in terms of LRU);
    return true;
else
    if Name Cache is full then
        remove least recently used name from Name Cache;
    add chunk name to Name Cache;
    return false;
end
```

ffc_policy::data_to_cache(msg)

```
if msg name matches entry in name cache then
    if FFC(p) then
        if random number $<$ p then
            return true;
        else
            return false;
        end
    if TFC or WFC(p) then
        if msg->getHops() == 1 then
            return true;
        else
            return false;
        end
    else
        return true;
    end
if WFC(p) then
    if random number $<$ p then
        return true;
else
    return false;
end
```

Appendix C

Changes to ccnSim

Here we list some changes made to ccnSim that are not necessarily related to Filter First Caching but assist with the general use of the simulation tool.

- **Improved debugging:** In addition to moving unnecessary output into the existing debugging state (*SEVERE_DEBUG*), we created our own debugging state (*STREAM_DEBUG*). With *#define STREAM_DEBUG* added to include/ccnsim.h, a user can see the requests and content reaching each node and the decision whether to cache in both the name and main cache. In this way, you can assess any changes made to Filter First Caching, or any other caching mechanism, and the effects is has on the decision to cache.
- **Fixed error with stabilisation process:** The partial_n variable that specifies the number of nodes in the network required to be filled, and then subsequently stable, was being incorrectly rounded (in *src/statistics.cc*).
- **Ensure that simulation does not attempt to stabilise with never caching mechanism:** If partial_n was greater than 0 for the never caching mechanism then stabilisation was only achieved after a lengthy time-out period.
- **Memory Leaks:** ccnSim has multiple memory leaks with dynamic memory not accurately cleaned up. The notable sources that would need to be addressed in order to repeat simulations in this paper:
  - *src/node/strategy/nrr.cc*: add "MonopathStrategyLayer::finish();" to the "void nrr::finish()" function;
  - *src/statistics/statistics.cc*: add "delete stable_check;" to the "void statistics::finish()" function;

Bibliography

[1] CCNPL software 2014 [Online] URL: http://systemx.enst.fr/ccnpl-sim. pages 31

[2] Dario Rossi : Software / CcnSim [Online] URL: http://perso.telecom-paristech.fr/˜drossi/index.php?n=Software.CcnSim. pages 47

[3] ns-3: discrete-event network simulator for Internet systems [Online] URL: https://www.nsnam.org/. pages 31

[4] SAIL Project [Online] URL: http://www.sail-project.eu/about-sail/netinf/. pages 7

[5] UW CSE | Systems Research | Rocketfuel. pages 34

[6] Stanford University TRIAD Project [Online] URL: http://www.gregorio.stanford.edu/triad/, 2000. pages 13

[7] Cisco Visual Networking Index: Forecast and Methodology, 2014 to 2019. 2015. pages 1

[8] M. Aamir and S.M.A. Zaidi. Denial-of-service in content centric (named data) networking: a tutorial and state-of-the-art survey. *Security and Communication Networks*, 8(11), 7 2015. pages 11

[9] E.G. AbdAllah, H.S. Hassanein, and M. Zulkernine. A survey of security attacks in information-centric networking. *IEEE Communications Surveys &amp; Tutorials*, 17(3), 2015. pages 11

[10] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. ndnSIM: NDN simulator for NS-3. *NDN*, 2012. pages 31

[11] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7), 7 2012. pages 5, 13

[12] Dewan Tanvir Ahmed and Shervin Shirmohammadi. Design issues of peer-to-peer systems for wireless ad hoc networks. *Proceedings of the Sixth International Conference on Networking, ICN'07*, 2007. pages 1

[13] Alfred V Aho, Peter J Denning, and Jeffrey D Ullman. Principles of Optimal Page Replacement. pages 36

[14] Somaya Arianfar, Pekka Nikander, and JÃűrg Ott. Packet-level Caching for Information-centric Networking. 2010. pages 39

[15] Jordan Augé, Giovanna Carofiglio, Giulio Grassi, Luca Muscariello, Giovanni Pau, and Xuan Zeng. Anchor-less Producer Mobility in ICN. pages 10

[16] Mikhail Badov, Anand Seetharam, Jim Kurose, Victor Firoiu, and Soumendra Nanda. Congestion-Aware Caching and Search in Information-Centric Networks. pages 26

[17] Alcardo Alex Barakabitze, Tan Xiaoheng, and Guo Tan. A Survey on Naming, Name Resolution and Data Routing in Information Centric Networking (ICN). *International Journal of Advanced Research in Computer and Communication Engineering*, 3(10), 2014. pages 7

[18] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. *Proceedings - IEEE INFOCOM*, 1:126–134, 1999. pages 38, 39

[19] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello. From content delivery today to information centric networking. *Computer Networks*, 57(16), 11 2013. pages 1

[20] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. On the performance of bandwidth and storage sharing in information-centric networks. 2013. pages 38

[21] Cheng-Yun Ho, Cheng-Yuan Ho, and Chien-Chao Tseng. A case study of cache performance in ICN - various combinations of transmission behavior and cache replacement mechanism. *2015 17th International Conference on Advanced Communication Technology (ICACT)*, 2015. pages 36

[22] Raffaele Chiocchetti, Dario Rossi, Giuseppe Rossini, Giovanna Carofiglio, and Diego Perino. Exploit the known or explore the unknown? Hamlet-like doubts in ICN. *ICN'12 - ACM Proceedings of the Information-Centric Networking Workshop*, pages 7–12, 2012. pages 8, 35

[23] Matteo D 'ambrosio, Christian Dannewitz, Holger Karl, and Vinicio Vercellone. MDHT: A Hierarchical Name Resolution Service for Information-centric Networks. 2011. pages 7

[24] Ali Dabirmoghaddam, Maziar Mirzazad-Barijough, and J J Garcia-Luna-Aceves. Understanding Optimal Caching and Opportunistic Caching at The Edge of Information-Centric Networks. pages 2, 42, 43

[25] Jie Dai, Fangming Liu, and Bo Li. The disparity between P2P overlays and ISP underlays: Issues, existing solutions, and challenges. *IEEE Network*, 24(6):36–41, 12 2010. pages 1

[26] S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch. Information-Centric Networking (ICN) Research Challenges. 7 2016. pages 11

[27] Suyong Eum, Kiyohide Nakauchi, Masayuki Murata, Yozo Shoji, and Nozomu Nishinaga. CATT: Potential Based Routing with Content Caching for ICN. 2012. pages 22

[28] S.K. Fayazbakhsh, Yin Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B.M. Maggs, K.C. Ng, V. Sekar, and S. Shenker. Less Pain, Most of the Gain: Incrementally Deployable ICN. *Computer Communication Review*, 43(4), 10 2013. pages 2, 42, 43

[29] Nikos Fotiou, Pekka Nikander, Dirk Trossen, and George C Polyzos. Developing Information Networking Further: From PSIRP to PURSUIT. pages 6

[30] Massimo Gallo, Bruno Kauffmann, Luca Muscariello, Alain Simonian, and Christian Tanguy. Performance Evaluation of the Random Replacement Policy for Networks of Caches. pages 37

[31] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: seeing the forest for the trees. *the 10th ACM Workshop*, pages 1–6, 2011. pages 2, 13, 37, 42, 43

[32] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. YouTube Traffic Characterization: A View From the Edge. pages 39

[33] Guoqiang Zhang, Xiaohui Wang, Qian Gao, and Zhen Liu. A Hybrid ICN Cache Coordination Scheme Based on Role Division between Cache Nodes. *2015 IEEE Global Communications Conference (GLOBECOM). Proceedings*, 2015. pages 62

[34] V. Jacobson, D.K. Smetters, J.D. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking Named Content. *Communications of the ACM*, 55(1), 1 2012. pages 5, 13, 22, 35

[35] Jiang Xiaoke, Bi Jun, Nan Guoshun, and Li Zhaogeng. A survey on Information-centric networking: Rationales, designs and debates. *China Communications*, 12(7), 7 2015. pages 5, 13

[36] Kideok Cho, Munyoung Lee, Kunwoo Park, T.T. Kwon, Yanghee Choi, and Sangheon Pack. WAVE: popularity-based and collaborative in-network caching for content-oriented networks. *IEEE INFOCOM 2012 - IEEE Conference on Computer Communications Workshops*, 2012. pages 22, 35

[37] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A Data-Oriented (and Beyond) Network Architecture. pages 6, 13

[38] Dmitrij Lagutin, Kari Visala, and Sasu Tarkoma. Publish/Subscribe for Internet: PSIRP Perspective. *Towards the Future Internet*, pages 75–84, 2010. pages 6

[39] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. The LCD interconnection of LRU caches and its analysis. *Performance Evaluation*, 63(7):609–634, 7 2006. pages 12, 22, 26, 29, 35

[40] Nikolaos Laoutaris, Sofia Syntila, and Ioannis Stavrakakis. Meta algorithms for hierarchical web caches. *IEEE International Performance, Computing and Communications Conference, Proceedings*, 23:445–452, 2004. pages 22, 35

[41] Yang Li, Tao Lin, Hui Tang, and Peng Sun. A chunk caching location and searching scheme in Content Centric Networking. *IEEE International Conference on Communications*, pages 2655–2659, 2012. pages 22

[42] Zhe Li and Gwendal Simon. Time-shifted TV in content centric networks: The case for cooperative in-network caching. In *IEEE International Conference on Communications*, 2011. pages 22

[43] Diego Perino and Matteo Varvello. A reality check for content centric networking. *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, ICN'11*, pages 44–49, 2011. pages 10, 17, 28

[44] G. Piro, S. Signorello, M. R. Palattella, L. A. Grieco, G. Boggia, and T. Engel. Understanding the social impact of ICN: between myth and reality. *AI & SOCIETY*, 2 2016. pages 5, 13

[45] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. *ICN'12 - ACM Proceedings of the Information-Centric Networking Workshop*, pages 55–60, 2012. pages 22, 28, 35

[46] Jarno Rajahalme, Mikko Särelä, Pekka Nikander, and Sasu Tarkoma. Incentive-Compatible Caching and Peering in Data-Oriented Networks. pages 22

[47] E.J. Rosensweig and J. Kurose. Breadcrumbs: efficient, best-effort content location in cache networks. *2009 Proceedings IEEE INFOCOM. 28th International Conference on Computer Communications*, 2009. pages 22

[48] Dario Rossi and Giuseppe Rossini. Caching performance of content centric networks under multi-path routing (and more). pages 31, 32, 34, 37, 55

[49] Dario Rossi and Giuseppe Rossini. On sizing CCN content stores by exploiting topological information. pages 22, 38

[50] Giuseppe Rossini and Dario Rossi. Evaluating CCN multi-path interest forwarding strategies. pages 32, 38, 55

[51] Giuseppe Rossini and Dario Rossi. Coupling caching and forwarding: Benefits, analysis, and implementation. *ICN 2014 - Proceedings of the 1st International Conference on Information-Centric Networking*, pages 127–136, 9 2014. pages 32, 33, 35, 37, 42

[52] Lorenzo Saino, Ioannis Psaras, and George Pavlou. Icarus: a Caching Simulator for Information Centric Networking (ICN). *SIMUTools*, 2014. pages 31

[53] D. Saucez, S. Secci, and C. Barakat. On the incentives and incremental deployments of ICN technologies for OTT services. *IEEE Network*, 28(3), 5 2014. pages 10

[54] Vasilis Sourlas, Lazaros Gkatzikis, Paris Flegkas, and Leandros Tassiulas. Distributed cache management in information-centric networks. *IEEE Transactions on Network and Service Management*, 2013. pages 22

[55] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP Topologies with Rocketfuel. pages 34

[56] David G Thaler and Chinya V Ravishankar. Using Name-Based Mappings to Increase Hit Rates. pages 8

[57] M. Tortelli, D. Rossi, G. Boggia, and L.A. Grieco. ICN software tools: Survey and cross-comparison. *Simulation Modelling Practice and Theory*, 63:23–46, 4 2016. pages 31

[58] Michele Tortelli, Dario Rossi, and Emilio Leonardi. ModelGraft: Accurate, Scalable, and Flexible Performance Evaluation of General Cache Networks. pages 31

[59] Gareth Tyson, Nishanth Sastry, Ruben Cuevas, Ivica Rimac, and Andreas Mauthe. A survey of mobility in information-centric networks. *Communications of the ACM*, 56(12):90–98, 12 2013. pages 2, 10

[60] A.V. Vasilakos, Zhe Li, G. Simon, and Wei You. Information centric network: Research challenges and opportunities. *Journal of Network and Computer Applications*, 52, 6 2015. pages 5, 13

[61] Jason Min Wang, Jun Zhang, and Brahim Bensaou. Intra-AS Cooperative Caching for Content-Centric Networks. pages 22

[62] Meisong Wang, Prem Prakash Jayaraman, Rajiv Ranjan, Karan Mitra, Miranda Zhang, Eddie Li, Samee Khan, Mukkaddim Pathan, and Dimitrios Georgeakopoulos. An overview of cloud based content delivery networks: Research dimensions and state-of-the-Art. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9070:131–158, 2015. pages 1

[63] Wei Koong Chai, Diliang He, I. Psaras, and G. Pavlou. Cache less for more in information-centric networks. *NETWORKING 2012. Proceedings 11th International IFIP TC 6 Networking Conference*, 2012. pages 22

[64] George Xylomenos, Christopher N. Ververidis, Vasilios A. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros, and George C. Polyzos. A Survey of Information-Centric Networking Research. *IEEE Communications Surveys & Tutorials*, 16(2):1024–1049, 22 2014. pages 5, 13

[65] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D Thornton, Diana K Smetters, Beichuan Zhang, Gene Tsudik, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. Named Data Networking (NDN) Project. *NDN*, 2010. pages 13

[66] Meng Zhang, Hongbin Luo, and Hongke Zhang. A survey of caching mechanisms in information-centric networking. *IEEE Communications Surveys and Tutorials*, 17(3):1473–1499, 7 2015. pages 19, 22